

workiva

Utilities Workshop

Guide to Wdata & Chains Build

Presenter Name

APAC SA's

Design Philosophy

Operational Excellence



Reliability



Security



Performance



Empathy



Client's Case Story

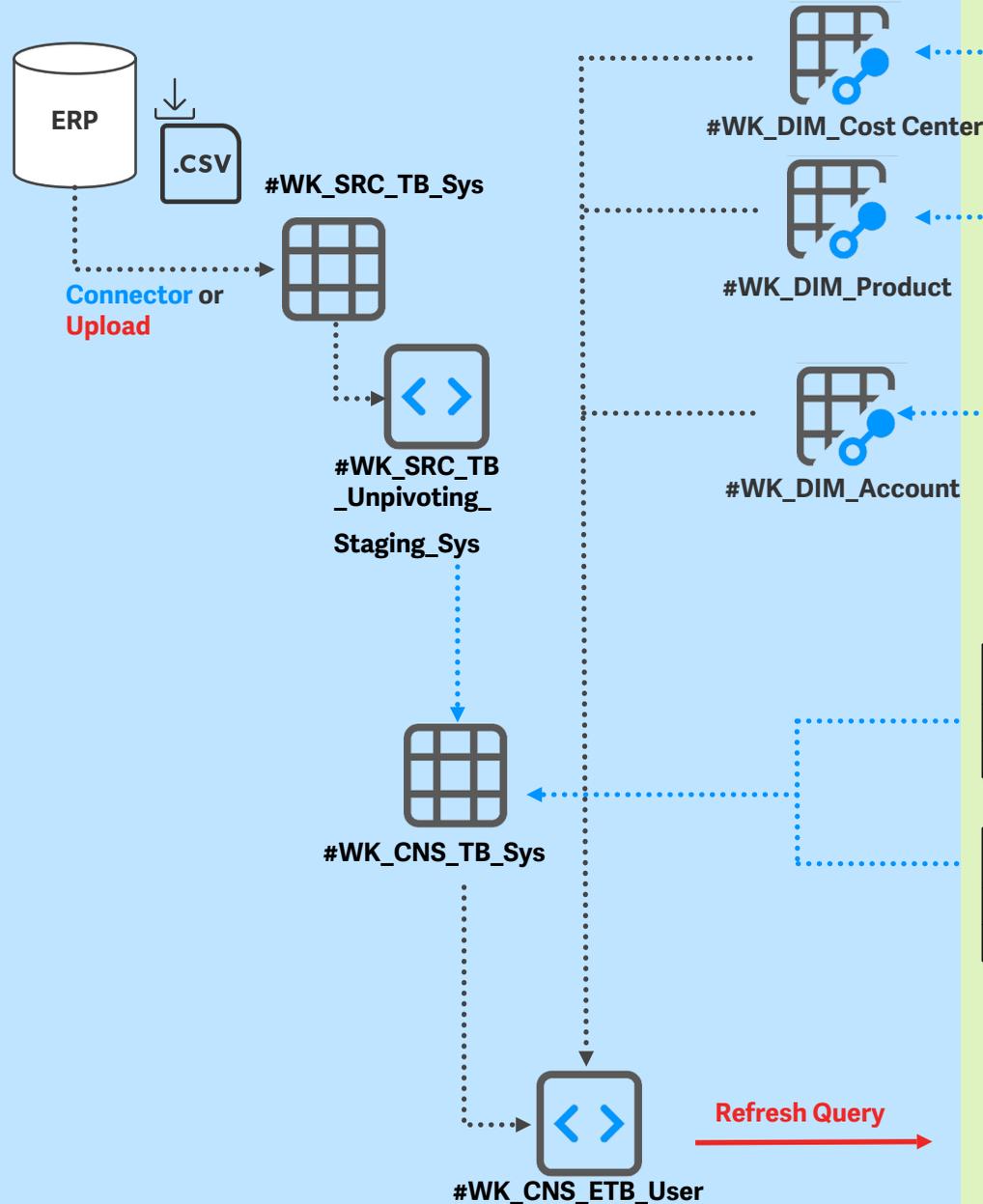
Our client, Dreamer Inc. has chosen Workiva to be their consultants in helping them improve their data workflow and processes. The end outcome that we agreed is for Workiva to generate a Master Spreadsheet with all the relevant data sets they need for their Financial Reports.

They have sent across their source files that flows to their Financial Reports that comprises of:

1. Trial Balance Actuals Dec'19, Nov'20, Dec'20 (.csv)
2. Adjustments & Budget Input Template Dec'20 (.csv)
3. Chart of Accounts (.csv)
4. Cost Center Mapping (.csv)
5. Product Mapping (.csv)

Workiva's objective is to setup a Workspace for Dreamer Inc. to design a model that automates their Monthly Financial Reporting process. Let us now take a look at the expected process of the data flow with Workiva.

- Automated by Wdata Queries
- Automated by Wdata Chains
- Manual User Call



Control Tower Sheet



- Controls the Year and Month to facilitate the Roll-forward process
- Feeds chain with spreadsheet IDs, query IDs, query parameters and data tags

Cost Centre Mapping

Account Code	Cost Centre Code	Cost Centre Name	ccru2	ccru3	ccru4
16-13-00-00-111	CC4337	Investment Solutions	Funds Management	Investment Solutions	
17-12-00-00-124	CC6173	Facilities Management	Support Units	Operations	Investment Operations

Product Mapping

productcode	productname	fmproductassetclass	fmproducttype	fmproductgeography	fmproductclass	productru
4062	Product - 284	Fixed Income	Fund	Australia	Institutional	Funds Management
4015	Product - 262	Fixed Income	Mandate	Australia	Institutional	Funds Management

Chart of Accounts / Account Hierarchy/ FS Mapping

GL Account	GL Description	FS	FS1	FS2
11-11-00-00-112	Intangible Assets - Software Licence	BS	Non current assets	Intangible assets
11-12-00-00-112	Intangible Assets Amortisation - Software Licence	BS	Non current assets	Intangible assets

Extended CoA

GL Account	GL Description	FS	FS1	FS2
BG001	Intangible assets	BS	Non current assets	Intangible assets
BG002	Bank loan - non current	BS	Non Current liability	Bank loan - non current

Adjustments

Current_FY	FS	Account_Code	Description	Adj_Jan	Adj_Feb	Adj_Mar	YTD_Adj_Jan	YTD_Adj_Feb	YTD_Adj_Mar
2020	BS	13-11-11-00-111	Share Capital	25,960.00	37,809.00	22,557.00		27,980.00	27,980.00	27,980.00
2020	BS	13-11-11-00-112	Share Premium	30,100.00	68,985.00	14,328.00		32,120.00	32,120.00	32,120.00

Budget

Current_FY	FS	Account_Code	Description	Bud_Jan	Bud_Feb	Bud_Mar	YTD_Bud_Jan	YTD_Bud_Feb	YTD_Bud_Mar
2020	BS	BG001	Intangible assets	8,671,304.80	1,143,195.83	10,999,007.36		8,673,324.80	8,673,324.80	8,673,324.80
2020	BS	BG002	Bank loan - non current	(0.12)	11,677.70	4,137,598.02		2,019.88	2,019.88	2,019.88

Extended Trial Balance

Current_FY	GL Account	GL Description	FS	FS1	FS2	Cost Centre	Product Code	CY_YTD	CY_FTM	CY_MoM	PY_YTD	PY_FTM
2020	11-11-00-00-112	Software Licence	BS	Non current assets	Intangible assets	General	7405	2,583,341	2,583,341	(726,565)	1,988,920	1,808,109
2020	11-12-00-00-112	Intangible Assets Amortisation	BS	Non current assets	Intangible assets	Mgmt Team	4102	(1,077,919)	(1,077,919)	303,165	(984,024)	(894,568)

Wdesk Workspace Set-up

Objectives:

1. Import Reporting Package.tar.gz base spreadsheet template
2. Create Folders for organizing tables & queries

Points to note:

- Base template is purely for training purpose to provide the necessary resources & templates to build for this exercise. It does not serve to be used for client's projects
- Creating folders allows the separation and identification of Workiva's Resources & Client's Resources
 - We do not work in different environments (e.g. UAT, SIT, Production)

Creating Tags

Objectives:

1. Setting up Tag Names & Tag Values



TB_Dec_2019



TB_Nov_2020



TB_Dec_2020



Wdata Tables

Data (Fact) Table

* Fact Table

Enter Table Description

Table Type (?) **Data** Folder No Folder Selected (root) Add Columns from a .CSV or .TSV file Choose file... Browse Comma Delimiter **+ Add Column**

DISPLAY NAME	DESCRIPTION	COLUMN ID	TYPE (?)	IMPORT FORMAT (?)	SOURCE VALUE
* Enter New Display Name	Enter New Description	* Enter New ID	Text ”		

Dimension Table

* Dimension Table

Enter Table Description

Table Type (?) **Dimension** Folder No Folder Selected (root) Add Columns from a .CSV or .TSV file Choose file... Browse Comma Delimiter **+ Add Column**

DISPLAY NAME	DESCRIPTION	COLUMN ID	TYPE (?)	IMPORT FORMAT (?)	SOURCE VALUE	KEY (?)
* Enter New Display Name	Enter New Description	* Enter New ID	Text ”			<input type="checkbox"/>

Data (Fact) Table Vs Dimension Table

Data (Fact) Table

- Typically to store transactional data i.e. GL transactions
- Allows for duplicate transactions i.e. an account tied to a dimension in multiple rows
- Typically imports and stacks data files every reporting cycle

Dimension Table

- To store Dimension Data i.e. Data that defines and categorizes Fact Data
- Does not allow any duplicate data as a dimension data cannot define & categorize the same fact data differently i.e. an account cant be defined as 2 different product codes
- There is always a key found between the fact & dimension data to define and categorize the given transaction
- Typically replaces Dimension Data File when it is updated instead of stacking files

Relationship between Fact & Dimension

Data (Fact) Table

year	month	entity	account_code	description	forward_balance	debit	credit	ending_balance	product_code
2020	8	c5854	11-11-00-00-112	Intangible Assets - Software Licence	1789743.384	0	0	1789743.384	7405
2020	7	c5854	11-12-00-00-112	Intangible Assets Amortisation - Software Licence	-240686.556	0	497214.476	-737901.032	4102
2020	12	c2452	13-11-11-00-111	INV Subsidiary Share Capital	2.80E+07	0	0	2.80E+07	2010
2020	9	c2445	13-11-11-00-112	INV Subsidiary Share Premium	6135797.868	0	0	6135797.868	2095
2020	9	c5940	13-11-11-00-115	INV Other IC Share Capital	0	1.682208	0	1.682208	7898
2020	1	c2452	13-11-11-01-111	INV Subsidiary Share Capital FMV	3.18E+07	2.65E+07	1.32E+07	4.51E+07	2112
2020	10	c5854	15-11-00-00-113	Deposits Receivable - Other Deposit & Guarantee {1Y+}	50929.25749	164393.9293	100349.6494	114973.5374	7152
2020	8	c5854	15-11-00-00-113	Deposits Receivable - Other Deposit & Guarantee {1Y+}	45783.2232	147783.108	90210.0408	103356.2904	7152
2020	6	c5854	15-11-00-00-113	Deposits Receivable - Other Deposit & Guarantee {1Y+}	46253.02566	149299.5776	91135.72703	104416.8763	7152
2020	9	c5917	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	1.4256	0	0	1.4256	2038
2020	5	c5854	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	82557.25546	3604.176	40741.6055	45419.82595	2038
2020	4	c2445	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	1122996.487	215182.656	0	1338179.143	2038
2020	3	c2452	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	0.16848	0	0	0.16848	2038
2020	11	c5940	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	14536.8432	0	0	14536.8432	2038

Dimension Table

productcode	productname	fmproductassetclass	fmproducttype	fmproductgeography	fmproductclass	productru
2010	Product - 101	Real Estate	Fund	Australia	Institutional	Funds Management
2038	Product - 113	Equities	Mandate	Australia	Institutional	Funds Management
2095	Product - 117	Fixed Income	Mandate	Australia	Institutional	Funds Management
2112	Product - 126	Equities	Fund	Australia	Retail	Funds Management
4102	Product - 305	Fixed Income	Fund	Australia	Institutional	Funds Management
7152	Product - 486	Alternatives	Mandate	Australia	Institutional	Funds Management
7405	Product - 499	Fixed Income	Fund	Australia	Retail	Funds Management
7898	Product - 598	Fixed Income	Mandate	Australia	Institutional	Funds Management

Examples of Fact/Dimension Relationships

Data (Fact) Table

- Trial Balance
 - † Account Number
- General Ledger
 - † Product Code
- General Ledger
 - † Cost Center Abbreviation

Dimension Table

- Chart of Accounts
 - † Key - Account Number
- Product Listing/Table
 - † Key - Product Code
- Cost Center Listing/Table
 - † Cost Center Abbreviation

Creating Dimension Tables

Objectives:

1. Create Dimension Tables for Product Mapping, Cost Center Mapping & Accounts Mapping
2. Configuring Column Types
3. Indicating which Field is the "Key" in each Dimension Table by checking the box
4. Import the client's mapping source files into the respective tables
5. Import Account Dim Extension from Spreadsheet into Accounts Mapping Dimension Table

Points to note:

- Most fields are represented as "Text" for column type information other than amount values
- "Key" configuration is only specific to Dimension Tables to prevent duplicate information from being imported
- Account Dim Extension exists as an approach Workiva recommends to allow clients to post Budget & Adjustments values at an aggregated level instead of breaking down to an individual Account level

How do Clients Post their Data?

Account Level

Current_FY	Entity	FS	Account_Code	Description	Product_Code	Adj_Jan	Adj_Feb	Adj_Mar	Adj_Apr	Adj_May	Adj_Jun	Adj_Jul	Adj_Aug	Adj_Sep	Adj_Oct	Adj_Nov	Adj_Dec
2020	c2445	BS	13-11-11-00-111	INV Subsidiary Share Capital	2010	25,960	37,809	22,557	39,108	46,644	64,774	69,742	71,423	39,613	46,415	27,586	60,697
2020	c2445	BS	13-11-11-00-112	INV Subsidiary Share Premium	2095	30,100	68,985	14,328	45,312	23,312	34,287	37,612	32,860	16,002	66,763	28,497	12,145
2020	c2445	BS	13-11-11-01-111	INV Subsidiary Share Capital FMV	2112	13,145	37,001	75,120	22,842	55,305	70,156	73,254	69,435	16,729	11,260	52,675	20,679
2020	c2445	BS	13-14-11-00-117	Loan to (>50%) IFL {1Y+}	2071	64,458	44,440	79,962	24,543	35,337	54,071	62,014	64,926	32,788	79,876	56,120	68,998
2020	c2445	BS	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	2038	60,705	38,141	73,732	37,875	58,770	70,585	41,757	22,825	79,841	69,032	77,370	68,830
2020	c2445	BS	17-12-00-00-111	BAN General Accounts - Account 1	2000	65,529	22,553	10,002	32,600	63,133	23,984	40,719	64,643	32,525	70,598	47,909	51,762
2020	c2445	PL	61-11-00-00-111	Corporate Costs - Secretarial Services	2122	75,270	74,979	34,598	50,480	66,425	77,828	62,146	69,333	64,875	79,004	29,599	75,160
2020	c2445	PL	61-11-00-00-114	Corporate Costs - Domiciliation	2035	58,736	76,670	25,826	19,622	28,317	77,604	39,282	75,554	72,476	55,098	27,787	56,140
2020	c2445	PL	61-11-00-00-131	Corporate Costs - P-Codes	2110	26,123	49,738	63,155	40,596	52,465	54,669	16,833	56,935	42,606	64,408	53,113	33,125
2020	c2445	PL	61-17-00-00-111	BRE Management Fees Exp. - Accounting	2173	68,455	72,029	27,821	52,630	43,653	56,522	73,884	62,812	25,409	73,855	79,768	23,167
2020	c2445	PL	61-18-00-00-112	Professional Fees - Audit	2177	38,715	58,796	56,214	12,834	78,743	16,503	74,256	75,375	52,973	10,747	26,415	78,501
2020	c2445	PL	61-18-00-00-114	Professional Fees - Tax Compliance	2099	43,826	18,709	26,155	76,642	30,678	71,570	31,807	35,469	67,154	42,186	31,688	20,496
2020	c2445	PL	73-21-15-00-111	Revaluation Loss On Financial Assets At FVTPL	2312	43,826	18,709	26,155	76,642	30,678	71,570	31,807	35,469	67,154	42,186	31,688	20,496

Financial Statement Statutory Level

Current_FY	Entity	FS	Account_Code	Description	Bud_Jan	Bud_Feb	Bud_Mar	Bud_Apr	Bud_May	Bud_Jun	Bud_Jul	Bud_Aug	Bud_Sep	Bud_Oct	Bud_Nov	Bud_Dec
2020	c2445	BS	BG001	Intangible assets	8671304.8	1143195.83	10999007.36	100717.3	162315.27	928777.73	1222644.89	2899667.51	0	338942.38	0	2458195.2
2020	c2445	BS	BG002	Bank loan - non current	-0.12	11677.7	4137598.02	-11677.7	-3742858.86	11903	-78993022.55	4599499.89	-2087265.08	44343747.01	0	-1.59
2020	c2445	BS	BG003	Loan receivable - non current	1222644.89	2899667.51	0	338942.38	0	2458195.2	5592360.75	3705769.73	-2249745	-600566	-3342615.75	-315265
2020	c2445	BS	BG004	Other receivables - current	0.17	-0.12	11677.7	4137598.02	-11677.7	-3742858.86	11903	-78993022.55	4599499.89	-2087265.08	44343747.01	0
2020	c2445	BS	BG005	Loan receivable - current	8671304.8	1143195.83	10999007.36	100717.3	162315.27	928777.73	1222644.89	2899667.51	0	338942.38	0	2458195.2
2020	c2445	BS	BG006	Cash and cash equivalents	-0.12	11677.7	4137598.02	-11677.7	-3742858.86	11903	-78993022.55	4599499.89	-2087265.08	44343747.01	0	-1.59
2020	c2445	PL	BG007	Other income	1222644.89	2899667.51	0	338942.38	0	2458195.2	5592360.75	3705769.73	-2249745	-600566	-3342615.75	-315265
2020	c2445	PL	BG008	Loan Interest income	0.17	-0.12	11677.7	4137598.02	-11677.7	-3742858.86	11903	-78993022.55	4599499.89	-2087265.08	44343747.01	0
2020	c2445	PL	BG009	Dividend Income	8671304.8	1143195.83	10999007.36	100717.3	162315.27	928777.73	1222644.89	2899667.51	0	338942.38	0	2458195.2
2020	c2445	PL	BG010	Interest income	-0.12	11677.7	4137598.02	-11677.7	-3742858.86	11903	-78993022.55	4599499.89	-2087265.08	44343747.01	0	-1.59
2020	c2445	PL	BG011	Other expenses	1222644.89	2899667.51	0	338942.38	0	2458195.2	5592360.75	3705769.73	-2249745	-600566	-3342615.75	-315265
2020	c2445	PL	BG006	Income tax expense	8671304.8	1143195.83	10999007.36	100717.3	162315.27	928777.73	1222644.89	2899667.51	0	338942.38	0	2458195.2

Relationships between Accounts & FS Statutory

Account Level

Current_FY	Entity	FS	Account_Code	Description	Product_Code	Adj_Jan	Adj_Feb	Adj_Mar	Adj_Apr	Adj_May	Adj_Jun	Adj_Jul	Adj_Aug	Adj_Sep	Adj_Oct	Adj_Nov	Adj_Dec
2020	c2445	BS	13-11-11-00-111	INV Subsidiary Share Capital	2010	25,960	37,809	22,557	39,108	46,644	64,774	69,742	71,423	39,613	46,415	27,586	60,697
2020	c2445	BS	13-11-11-00-112	INV Subsidiary Share Premium	2095	30,100	68,985	14,328	45,312	23,312	34,287	37,612	32,860	16,002	66,763	28,497	12,145
2020	c2445	BS	13-11-11-01-111	INV Subsidiary Share Capital FMV	2112	13,145	37,001	75,120	22,842	55,305	70,156	73,254	69,435	16,729	11,260	52,675	20,679
2020	c2445	BS	13-14-11-00-117	Loan to (>50%) IFL {1Y+}	2071	64,458	44,440	79,962	24,543	35,337	54,071	62,014	64,926	32,788	79,876	56,120	68,998
2020	c2445	BS	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	2038	60,705	38,141	73,732	37,875	58,770	70,585	41,757	22,825	79,841	69,032	77,370	68,830
2020	c2445	BS	17-12-00-00-111	BAN General Accounts - Account 1	2000	65,529	22,553	10,002	32,600	63,133	23,984	40,719	64,643	32,525	70,598	47,909	51,762
2020	c2445	PL	61-11-00-00-111	Corporate Costs - Secretarial Services	2122	75,270	74,979	34,598	50,480	66,425	77,828	62,146	69,333	64,875	79,004	29,599	75,160
2020	c2445	PL	61-11-00-00-114	Corporate Costs - Domiciliation	2035	58,736	76,670	25,826	19,622	28,317	77,604	39,282	75,554	72,476	55,098	27,787	56,140
2020	c2445	PL	61-11-00-00-131	Corporate Costs - P-Codes	2110	26,123	49,738	63,155	40,596	52,465	54,669	16,833	56,935	42,606	64,408	53,113	33,125
2020	c2445	PL	61-17-00-00-111	BRE Management Fees Exp. - Accounting	2173	68,455	72,029	27,821	52,630	43,653	56,522	73,884	62,812	25,409	73,855	79,768	23,167
2020	c2445	PL	61-18-00-00-112	Professional Fees - Audit	2177	38,715	58,796	56,214	12,834	78,743	16,503	74,256	75,375	52,973	10,747	26,415	78,501
2020	c2445	PL	61-18-00-00-114	Professional Fees - Tax Compliance	2099	43,826	18,709	26,155	76,642	30,678	71,570	31,807	35,469	67,154	42,186	31,688	20,496
2020	c2445	PL	73-21-15-00-111	Revaluation Loss On Financial Assets At FVTPL	2312	43,826	18,709	26,155	76,642	30,678	71,570	31,807	35,469	67,154	42,186	31,688	20,496

Financial Statement Statutory Level

GL Account	GL Description	FS	FS1	FS2	FS3	FS4	FS5
11-11-00-00-112	Intangible Assets - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
11-12-00-00-112	Intangible Assets Amortisation - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-11-17-00-111	Deferred Financing Fees - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Costs incurred	Bank loan - non current	Costs incurred
12-11-21-00-111	Formation Expenses	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-12-17-00-111	Deferred Financing Fees Amortisation - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Accumulated amortisation	Bank loan - non current	Accumulated amortisation
13-11-11-00-111	INV Subsidiary Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-112	INV Subsidiary Share Premium	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-113	INV Associate Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss

Approach Examples

Using Existing Accounts

GL Account	GL Description	FS	FS1	FS2	FS3	FS4	FS5
11-11-00-00-112	Intangible Assets - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
11-12-00-00-112	Intangible Assets Amortisation - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-11-17-00-111	Deferred Financing Fees - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Costs incurred	Bank loan - non current	Costs incurred
12-11-21-00-111	Formation Expenses	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-12-17-00-111	Deferred Financing Fees Amortisation - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Accumulated amortisation	Bank loan - non current	Accumulated amortisation
13-11-11-00-111	INV Subsidiary Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-112	INV Subsidiary Share Premium	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-113	INV Associate Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss

Creating Dummy Accounts

gl_account	gl_description	fs	fs1	fs2	fs3	fs4	fs5
BG002	Bank loan - non current	BS	Non Current liability	Bank loan - non current			
BG003	Loan receivable - non current	BS	Non current assets	Loan receivable - non current			
BG009	Dividend Income	PL	Revenues	Dividend Income			
BG007	Other income	PL	Revenues	Other income			
BG011	Other expenses	PL	Expense	Other expenses			
BG006	Cash and cash equivalents	BS	Current assets	Cash and cash equivalents			
BG012	Income tax expense	PL	Expense	Income tax expense			
BG001	Intangible assets	BS	Non current assets	Intangible assets			
BG004	Other receivables - current	BS	Current assets	Other receivables - current			
BG010	Interest income	PL	Revenues	Interest income			
BG008	Loan Interest income	PL	Revenues	Loan Interest income			
BG005	Loan receivable - current	BS	Current assets	Loan receivable - current			

Creating Account Dimension Extended

Objectives:

1. Import the Account Dimension Extended from Spreadsheet into the Account Dimension Table created previously
2. Import is done through Wdata Connection through the spreadsheet

Points to note:

- Account Dimension Extended is created for the ease of training for defining the Budget & Adjustment Data
- Typically our Account Dimension (COA) is defined at the GL Account Level. However with Budget & Adjustments, clients typically do not post at such granular level, and instead post these values at a higher hierarchy level
- Hence, we created newly defined accounts specifically for the Budget & Adjustments Process which represents their mapping to be aligned with the main Account Dimensions Data
- Alternatively, another approach is for the client to use existing accounts with the same given mapping they want to post it to for the Budget & Adjustments Process

What Can Queries Do ?

Objectives:

1. Create Data Table for Trial Balance Import for Dec'19, Nov'20 & Dec'20
2. Configuring Column Types
3. Selecting the Tag Name & Tag Value for the file being imported
4. Adding New Datasets after the Data Table is created

Points to note:

- Most fields are represented as "Text" for column type information other than amount values
- Tags are an option to utilize when uploading more than a single file into a table. As shown earlier, tags help to "bookmark" each file uploaded with a specific tag name & value
- Tags can essentially be a way to parameterize the query results using the tag name and values in subsequent steps

Why Use Queries ?

Queries help to generate a more specific result output based on the user's business rules and methodology of calculations from a huge bank of data from one or many data sources.

Data (Fact) Table

year	month	entity	account_code	description	forward_balance	debit	credit	ending_balance	product_code
2020	8	c5854	11-11-00-00-112	Intangible Assets - Software Licence	1789743.384	0	0	1789743.384	7405
2020	7	c5854	11-12-00-00-112	Intangible Assets Amortisation - Software Licence	-240686.556	0	497214.476	-737901.032	4102
2020	12	c2452	13-11-11-00-111	INV Subsidiary Share Capital	2.80E+07	0	0	2.80E+07	2010
2020	9	c2445	13-11-11-00-111	INV Subsidiary Share Premium	6135797.868	0	0	6135797.868	2095
2020	9	c5940	13-11-11-00-111	INV Other IC Share Capital	0	1.682208	0	1.682208	7898
2020	1	c2452	13-11-11-01-111	INV Subsidiary Share Capital FMV	3.18E+07	2.65E+07	1.32E+07	4.51E+07	2112
2020	10	c5854	15-11-00-00-111	Deposits Receivable - Other Deposit & Guarantee {1Y+}	50929.25749	164393.9293	100349.6494	114973.5374	7152
2020	8	c5854	15-11-00-00-111	Deposits Receivable - Other Deposit & Guarantee {1Y+}	45783.2232	147783.108	90210.0408	103356.2904	7152
2020	6	c5854	15-11-00-00-111	Deposits Receivable - Other Deposit & Guarantee {1Y+}	46253.02566	149299.5776	91135.72703	104416.8763	7152
2020	9	c5917	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	1.4256	0	0	1.4256	2038
2020	5	c5854	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	82557.25546	3604.176	40741.6055	45419.82595	2038
2020	4	c2445	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	1122996.487	215182.656	0	1338179.143	2038
2020	3	c2452	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	0.16848	0	0	0.16848	2038
2020	11	c5940	16-13-00-00-111	ICoRec (>50%) Current Account {1Y-}	14536.8432	0	0	14536.8432	2038

Financial Statement Statutory Level

GL Account	GL Description	FS	FS1	FS2	FS3	FS4	FS5
11-11-00-00-112	Intangible Assets - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
11-12-00-00-112	Intangible Assets Amortisation - Software Licence	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-11-17-00-111	Deferred Financing Fees - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Costs incurred	Bank loan - non current	Costs incurred
12-11-21-00-111	Formation Expenses	BS	Non current assets	Intangible assets	Intangible assets	Intangible assets	Intangible assets
12-12-17-00-111	Deferred Financing Fees Amortisation - Loan Facility Fees	BS	Non Current liability	Bank loan - non current	Accumulated amortisation	Bank loan - non current	Accumulated amortisation
13-11-11-00-111	INV Subsidiary Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-112	INV Subsidiary Share Premium	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss
13-11-11-00-113	INV Associate Share Capital	BS	Non current assets	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss	Investments at fair value through profit or loss

Query Result

year	account_code	ending_balance	FS1	FS2
2020	11-11-00-00-112	1,789,743	Non current assets	Intangible assets
2020	13-11-11-00-111	1,808,109	Non current assets	Investments at fair value through profit or loss
2020	16-13-00-00-111	1,990,911	Current assets	Other receivables - current
2020	16-16-11-00-112	1,809,919	Current assets	Other receivables - current
2020	17-12-00-00-113	1,628,927	Current assets	Cash and cash equivalents

What Can Queries Do ?

Client Challenges:

- † Finance teams handle immense amounts of data on a regular basis
- † They export raw data from source systems to allow them to "massage" the data to the required state
- † This includes filtering, pivoting, formularizing, transposing and many more
- † This is repeated over & over again every reporting cycle

What clients do not realize is that such a process is based on business rules & logic that are used repeatedly every cycle

Solution:

1. Queries pull data from data sources
2. Queries can help to:
 - a. Filter Data
 - b. Transform Data
 - c. Formularize Data
 - d. Repeat Output consistently without manual intervention

Creating Current Year Query

Objectives:

1. Add Data Sources
2. Adding Fields from Data Sources into the Query
3. Creating a Relationship between Different Data Sources
4. Creating a Formulated Result for Monthly & Year-To-Date Fields
5. Creating a Filter with Tags
6. Sorting Data Result

Points to note:

- Query Results will follow the order of fields arranged in the query as you drag them from the data sources into the center panel
- Relationships must be defined if more than one source exist. A primary key or a similar field must exist between different sources to define the relationship.

Replicating Current Year Query for Prior Year

Objectives:

1. Copy an existing current year query for the prior year
2. Renaming the fields from current year to prior year
3. Setting the filters for prior year

Points to note:

- Filters help to pull specific data results, in this case, using the value of the year in our data table.
- Earlier on when the data table was created, "Year" field was defined as a text column
- Because of that, in the filter, we have to cast it as an integer first to minus 1 from the current year, before casting it back to text
- This would allow us to pull prior year's data 1 year earlier from the current year selected as the basis

Staging/Transforming Data

Original File Format:

year	month	entity	account_code	description	product_code	forward_balance	debit	mar	credit	feb	ending_balance	product_code
2020	1	c5854	11-11-00-00-112	Intangible Assets - Soft		1355866.2		0		0	1355866.2	7405
2020	2	c5854	11-11-00-00-112	Intangible Assets - Soft		1627039.44		0		0	1627039.44	7405
2020	3	c5854	11-11-00-00-112	Intangible Assets - Soft		1757202.595		0		0	1757202.595	7405
2020	4	c5854	11-11-00-00-112	Intangible Assets - Soft		1809918.673		0		0	1809918.673	7405
2020	5	c5854	11-11-00-00-112	Intangible Assets - Soft		1628926.806		0		0	1628926.806	7405
2020	6	c5854	11-11-00-00-112	Intangible Assets - Soft		1808108.754		0		0	1808108.754	7405
2020	7	c5854	11-11-00-00-112	Intangible Assets - Soft		1491452.82		0		0	1491452.82	7405
2020	8	c5854	11-11-00-00-112	Intangible Assets - Soft		1789743.384		0		0	1789743.384	7405
2020	9	c5854	11-11-00-00-112	Intangible Assets - Soft		1932922.855		0		0	1932922.855	7405
2020	10	c5854	11-11-00-00-112	Intangible Assets - Soft		1990910.54		0		0	1990910.54	7405
2020	11	c5854	11-11-00-00-112	Intangible Assets - Soft		1791819.486		0		0	1791819.486	7405
2020	12	c5854	11-11-00-00-112	Intangible Assets - Soft		1988919.63		0		0	1988919.63	7405
2020	1	c5854	11-12-00-00-112	Intangible Assets Amor		-218805.96		0	452013.16		-670819.12	4102
2020	2	c5854	11-12-00-00-112	Intangible Assets Amor		-262567.152		0	542415.792		-804982.944	4102
2020	3	c5854	11-12-00-00-112	Intangible Assets Amor		-283572.5242		0	585809.0554		-869381.5795	4102
2020	4	c5854	11-12-00-00-112	Intangible Assets Amor		-292079.6999		0	603383.327		-895463.0269	4102
2020	5	c5854	11-12-00-00-112	Intangible Assets Amor		-262871.7299		0	543044.9943		-805916.7242	4102
2020	6	c5854	11-12-00-00-112	Intangible Assets Amor		-291787.6202		0	602779.9437		-894567.5639	4102
2020	7	c5854	11-12-00-00-112	Intangible Assets Amor		-240686.556		0	497214.476		-737901.032	4102
2020	8	c5854	11-12-00-00-112	Intangible Assets Amor		-288823.8672		0	596657.3712		-885481.2384	4102
2020	9	c5854	11-12-00-00-112	Intangible Assets Amor		-311929.7766		0	644389.9609		-956319.7375	4102
2020	10	c5854	11-12-00-00-112	Intangible Assets Amor		-321287.6699		0	663721.6597		-985009.3296	4102
2020	11	c5854	11-12-00-00-112	Intangible Assets Amor		-289158.9029		0	597349.4938		-886508.3966	4102

ETB Query Result to achieve:

current_fy	entity	account_code	product_code	description	fs	jan	feb	mar	jan_ytd	feb_ytd	mar_ytd	py_jan	py_feb	py_mar	py_jan_ytd	py_feb_ytd	py_mar_ytd	carry_forward	adj_jan	adj_feb	adj_mar	ytd_adj_jan	ytd_adj_feb	ytd_adj_mar	bud_ja
2020	c2445	61-18-00-00-1	2099	Professional F PL	#####	#####	#####	#####	44094	97006	154152														
2020	c2452	47-21-11-00-1	2160	Other Op. Inc. PL		-7.61	-9.1	-9.86	-7.61	-16.742	-26.605														



Agenda

-  **1** Wdata Guide
-  **2** Wdata Chains Starters Guide
-  **3** Chain : Manage Redundant Files
-  **4** Chain : Run Query & Upload Files
-  **5** Chain : File Importer
-  **6** Chain : Creating Template Chain
-  **7** Chain : External to Source
-  **8** Chain : Source to Staging
-  **9** Chain : Budget & Adjustments
-  **10** Chain : Master Chain

Design Philosophy

Operational Excellence



Reliability



Security



Performance

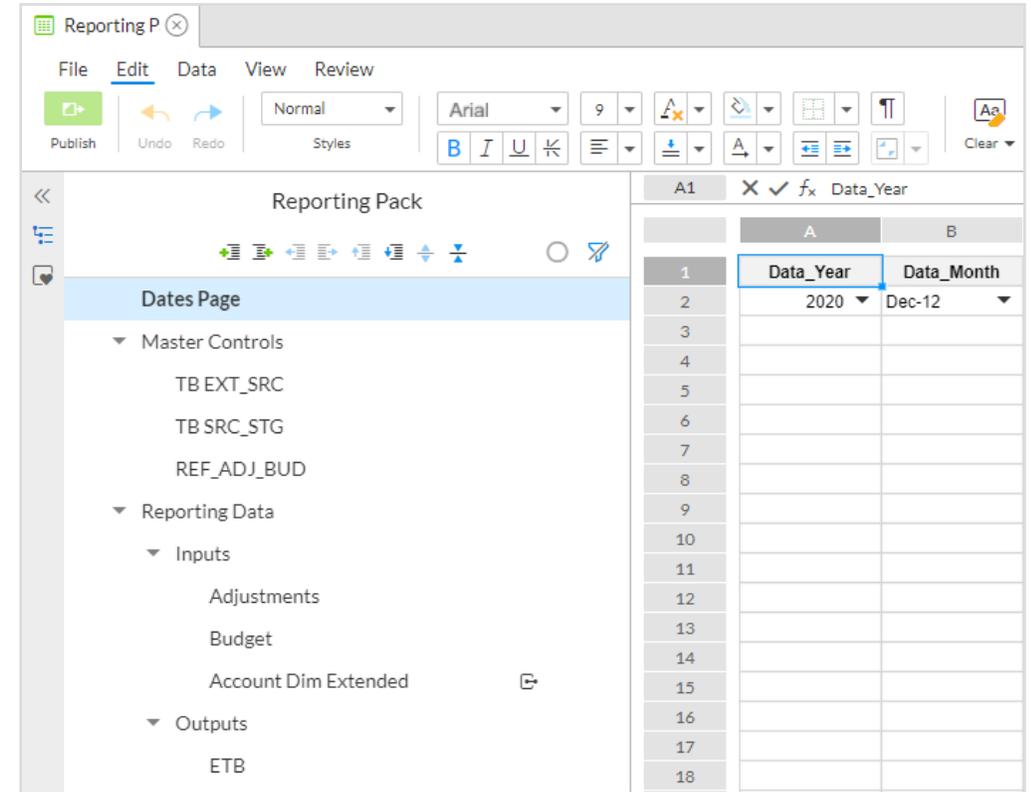
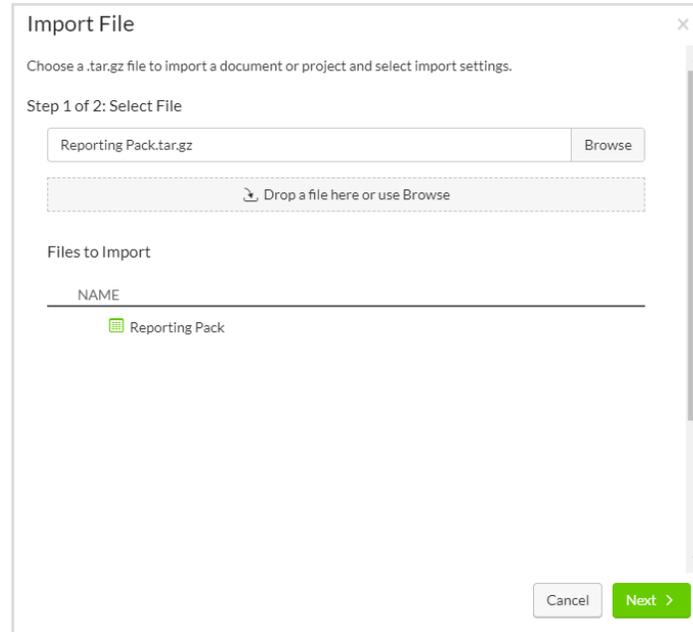
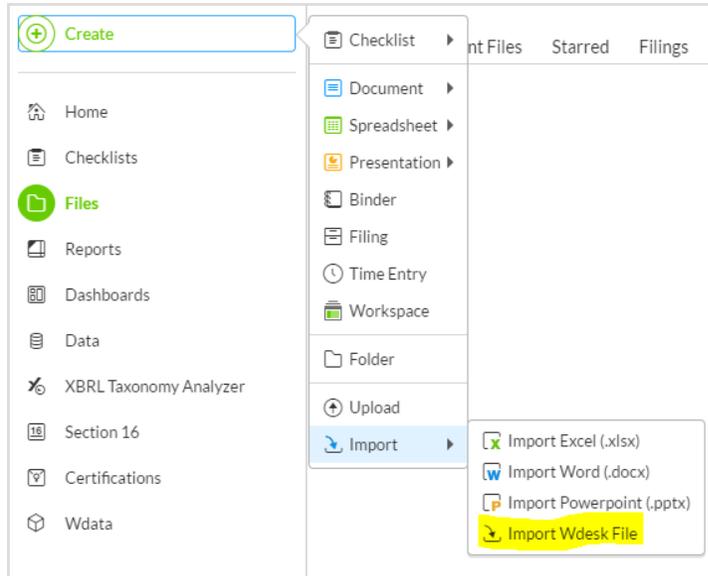


Empathy



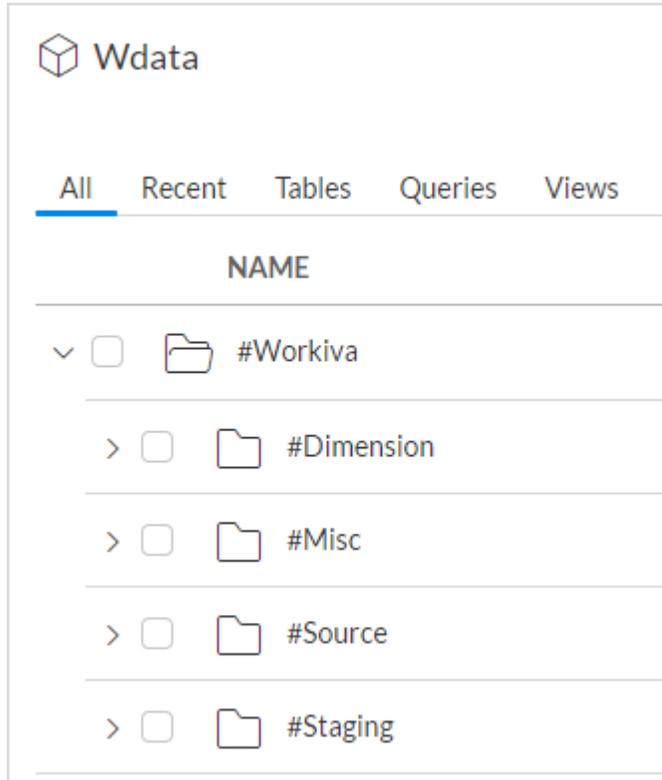
Wdata Tables, Queries

Wdesk Set-Up



To set-up Wdesk, import the "Reporting Pack.tar.gz" into Wdesk. This file would establish the required spreadsheet structure for this training.

Wdata Folder Structure



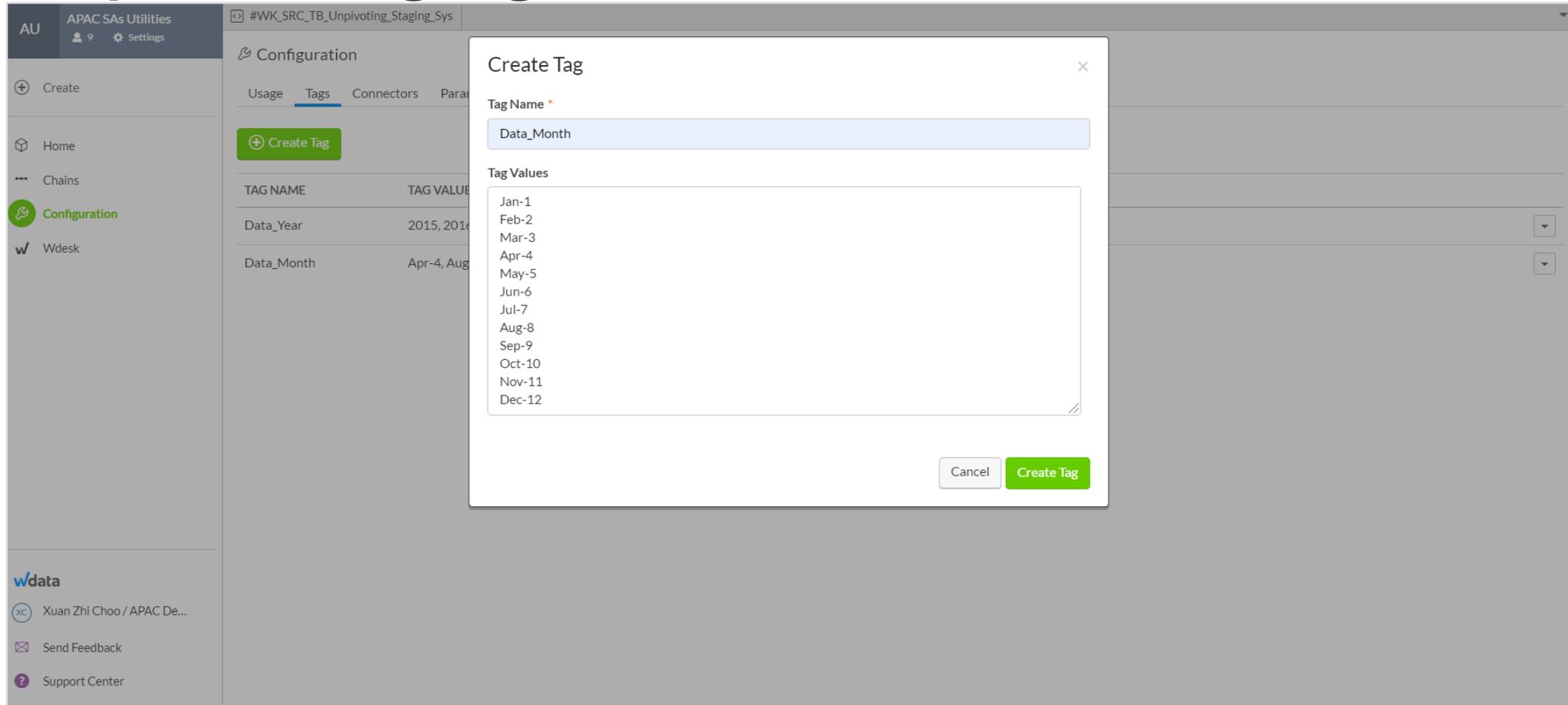
A screenshot of a file explorer window titled 'Wdata' showing a detailed view of the folder structure. The table below lists the contents of the '#Workiva' folder, including subfolders and tables with their creation and modification dates and authors.

NAME	CREATED	LAST MODIFIED
▼ #Workiva		
▼ #Dimension		
#WK_DIM_Cost Center	3/18/2021 2:33 PM	3/19/2021 12:44 PM Xuan Zhi Choo
#WK_DIM_Product	3/18/2021 4:39 PM	3/19/2021 12:44 PM Xuan Zhi Choo
#WK_DIM_Account	3/19/2021 10:27 AM	3/19/2021 12:44 PM Xuan Zhi Choo
#Misc		
▼ #Source		
#WK_SRC_TB_Sys	3/18/2021 2:42 PM	3/19/2021 12:44 PM Xuan Zhi Choo
▼ #Staging		
#WK_CNS_TB_Sys	3/26/2021 6:07 AM	3/30/2021 9:58 AM Xuan Zhi Choo
#WK_SRC_TB_Unpivoting_Staging_Sys	3/22/2021 1:13 PM	3/30/2021 9:58 AM Pankit Dhawan
#WK_CNS_ETB_User	3/26/2021 7:31 AM	3/30/2021 9:58 AM Xuan Zhi Choo

Wdata Folders are structured as follows:

- i. #Dimension - stores all the Dimension tables (i.e. COA, Product, Cost Center)
- ii. #Misc - stores miscellaneous (testing, troubleshooting) that is not required for the built
- iii. #Source - stores the Source table (TB GL Data)
- iv. #Staging - stores the staging query, consumption table, ETB query that drives the final figures

Step 4: Creating Tags



The screenshot shows the Workiva configuration interface. The main window is titled '#WK_SRC_TB_Unpivoting_Staging_Sys' and is in the 'Configuration' section, specifically the 'Tags' tab. A 'Create Tag' dialog box is open in the foreground. The dialog box has a title bar with a close button. It contains a 'Tag Name' field with the text 'Data_Month'. Below this is a 'Tag Values' field containing a list of months: 'Jan-1', 'Feb-2', 'Mar-3', 'Apr-4', 'May-5', 'Jun-6', 'Jul-7', 'Aug-8', 'Sep-9', 'Oct-10', 'Nov-11', and 'Dec-12'. At the bottom of the dialog box are two buttons: 'Cancel' and 'Create Tag' (which is highlighted in green). In the background, the 'Tags' configuration page is visible, showing a table with columns 'TAG NAME' and 'TAG VALUE'. The table has two rows: 'Data_Year' with value '2015, 2016' and 'Data_Month' with value 'Apr-4, Aug-8'. There is also a '+ Create Tag' button in the background.

TAG NAME	TAG VALUE
Data_Year	2015, 2016
Data_Month	Apr-4, Aug-8

Creating 'tags' allows tagging of data during import & creation of data tables. Create 2 tags, name 'Data_Year' & 'Data_Month'.

Step 5: Overview - Steps to Build

Steps	Description
1	First create Dimension tables using the 'COA, Product, Cost Center' data sets.
2	Next, create the TB source (SRC) table using the PY & CY Trial Balance Data.
3	Create a 'Staging' query ('#WK_SRC_TB_Unpivoting_Staging_Sys'), that Converts TB Fact Data with months Pivoted to 48 Columns (Jan to Dec movement + YTD Jan to YTD Dec) for Current & Prior Year.
4	The output from the 'Staging' query is used as an input into the consumption table ('#WK_CNS_TB_Sys').
5	Connect 'Budget' & 'Adjustments' spreadsheet to the consumption table ('#WK_CNS_TB_Sys').
6	Create the Consumption query (ETB) that drives the final results. This query would be connected to the 'Actuals ETB' sheet that would populate the final figures.
7	Set up 3 chains to automate the refresh of the Consumption table (#WK_CBS_TB_Sys) with the Staging query; i.e. step 3 & 4.
8	Glossary: i) DIM - Dimension tables (e.g. Mapping tables such as COA) ii) FACT - Actual data tables that stores financials data (e.g. Trial Balance)

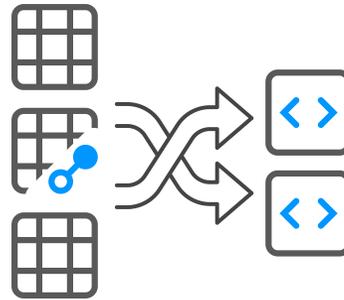
Wdata Tables & Queries



Data tables , also known as fact tables, contain information that builds up over time. For example, you can use data tables to curate trial balance data.



Dimension tables contain relational information, such as rollups by department or for mapping, like dates for fiscal year accounting.



Queries, slice-and-dice from multiple data tables, joins and curate information (financials) for analysis and reporting.

Wdata Tables - Data Transformation



Source Data from client's source systems.

- Trial Balance
- Adjustments information



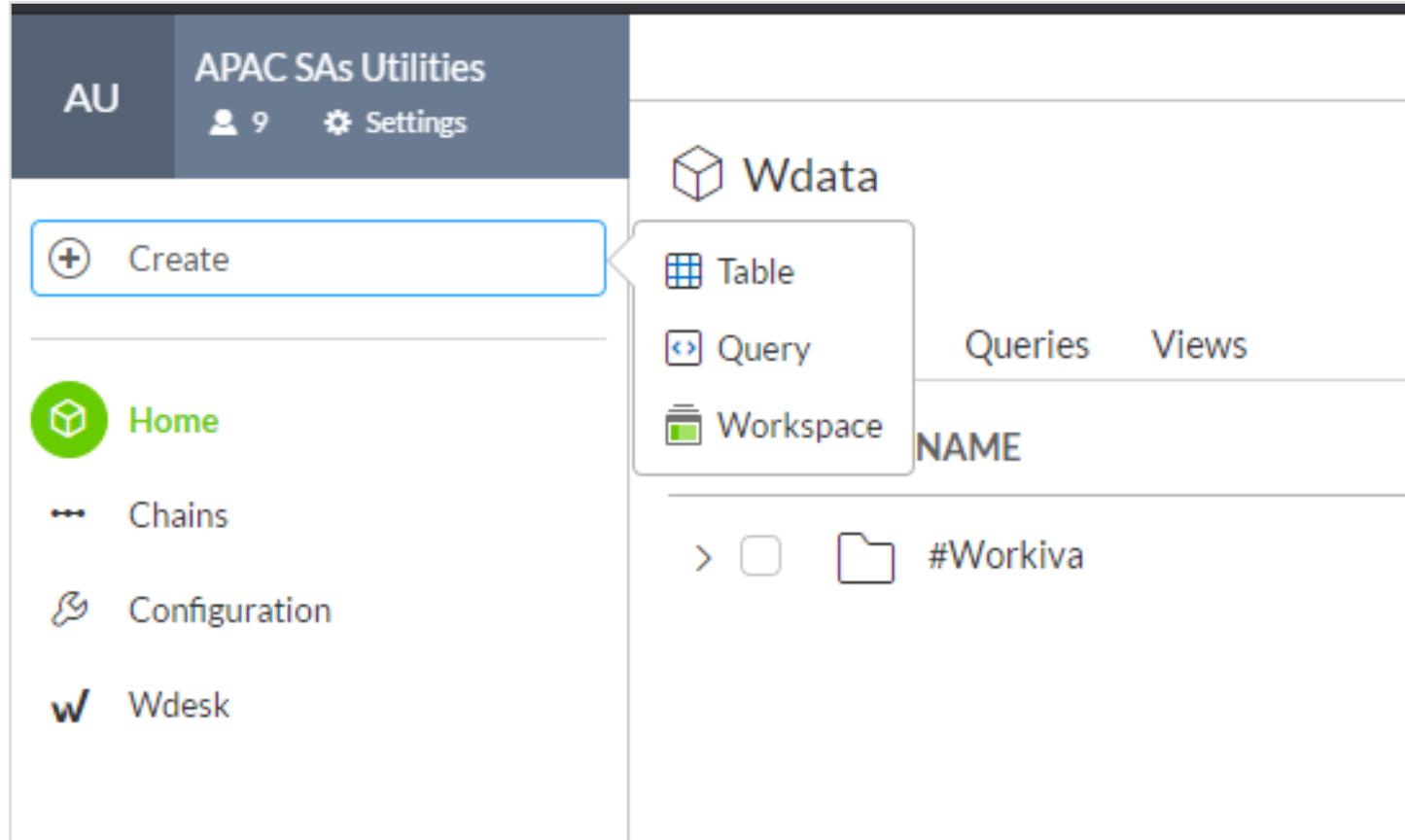
Queries are created to join both source and dimension data to transform



Dimension table - stores the main mapping data.

- Chart of Accounts (CoA)

Step 5.1: Creating COA Table



To create a table, click on 'Create', select 'Table' on the left panel

Step 5.2: Creating COA Table

New Table

#WK_DIM_Account

Enter Table Description

Table Type ? Folder Add Columns from a .CSV or .TSV file Delimiter

Dimension ? #Dimension Account Dimension.csv Browse Comma ? Add Column

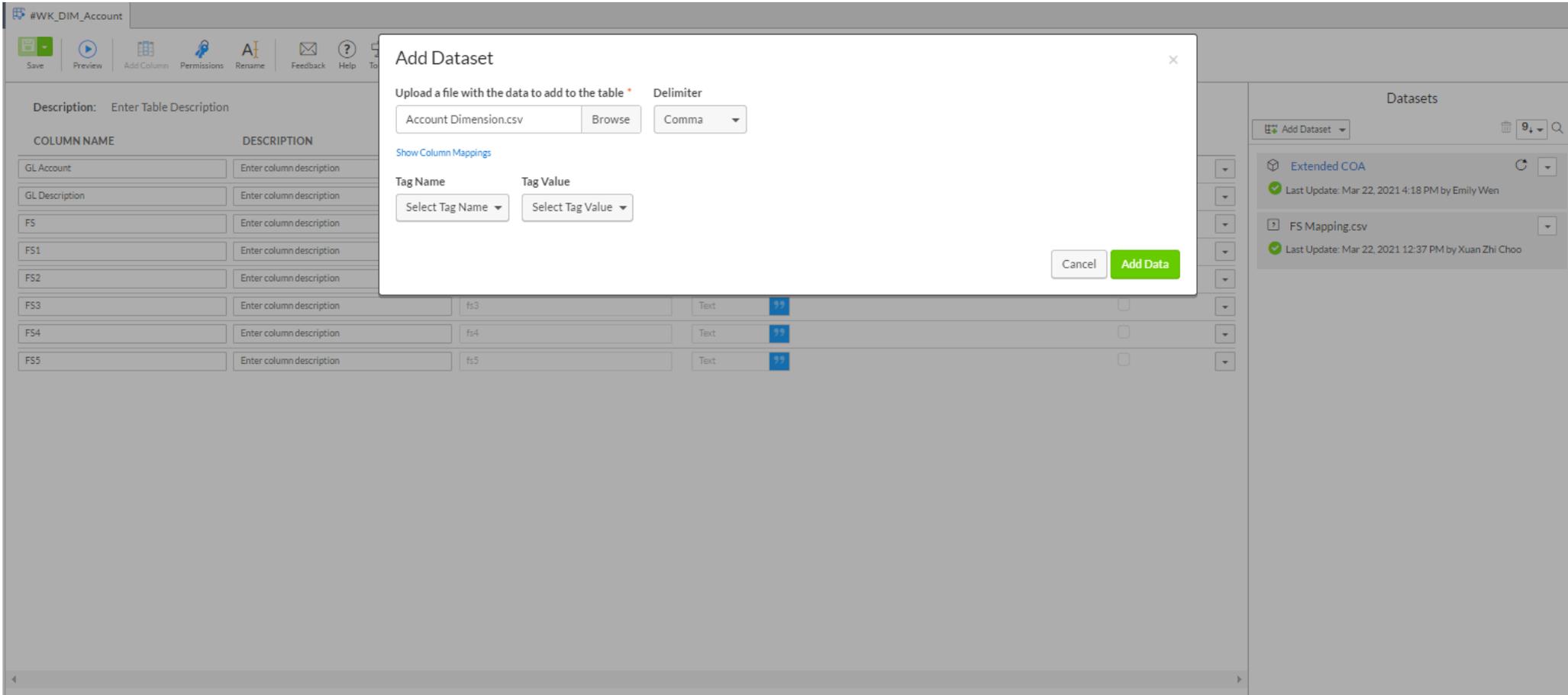
DISPLAY NAME	DESCRIPTION	COLUMN ID	TYPE ?	IMPORT FORMAT ?	SOURCE VALUE	KEY ?
GL Account	Enter New Description	GL_Account	Text ”		11-11-00-00-112	<input checked="" type="checkbox"/>
GL Description	Enter New Description	GL_Description	Text ”		Intangible Assets - Software Licence	<input type="checkbox"/>
FS	Enter New Description	FS	Text ”		BS	<input type="checkbox"/>
FS1	Enter New Description	FS1	Text ”		Non current assets	<input type="checkbox"/>
FS2	Enter New Description	FS2	Text ”		Intangible assets	<input type="checkbox"/>
FS3	Enter New Description	FS3	Text ”		Intangible assets	<input type="checkbox"/>
FS4	Enter New Description	FS4	Text ”		Intangible assets	<input type="checkbox"/>
FS5	Enter New Description	FS5	Text ”		Intangible assets	<input type="checkbox"/>

> File Preview

Tours Cancel Create Table

To create a table, we would need to import the Account Dimension.csv file that is provided via the 'Browse'. The first table that would be created is a COA/Account Dimension table that stores the Accounts data, setting the 'GL Account' as the primary key.

Step 5.3: Creating COA DIM Table P2



The screenshot displays the Workiva interface for creating a COA DIM Table P2. The main window shows a table configuration with columns for 'COLUMN NAME' and 'DESCRIPTION'. The table has rows for 'GL Account', 'GL Description', 'FS', 'FS1', 'FS2', 'FS3', 'FS4', and 'FS5'. A modal dialog titled 'Add Dataset' is open, allowing the user to upload a file. The dialog includes a file input field with 'Account Dimension.csv' and a 'Browse' button, a 'Delimiter' dropdown set to 'Comma', a 'Show Column Mappings' link, and 'Tag Name' and 'Tag Value' dropdowns. The 'Add Data' button is highlighted in green. The background interface includes a top navigation bar with icons for Save, Preview, Add Column, Permissions, Rename, Feedback, and Help. On the right, a 'Datasets' panel lists 'Extended COA' and 'FS Mapping.csv' with their last update dates and users.

COLUMN NAME	DESCRIPTION
GL Account	Enter column description
GL Description	Enter column description
FS	Enter column description
FS1	Enter column description
FS2	Enter column description
FS3	Enter column description
FS4	Enter column description
FS5	Enter column description

Add Dataset

Upload a file with the data to add to the table * Delimiter

Account Dimension.csv Comma

[Show Column Mappings](#)

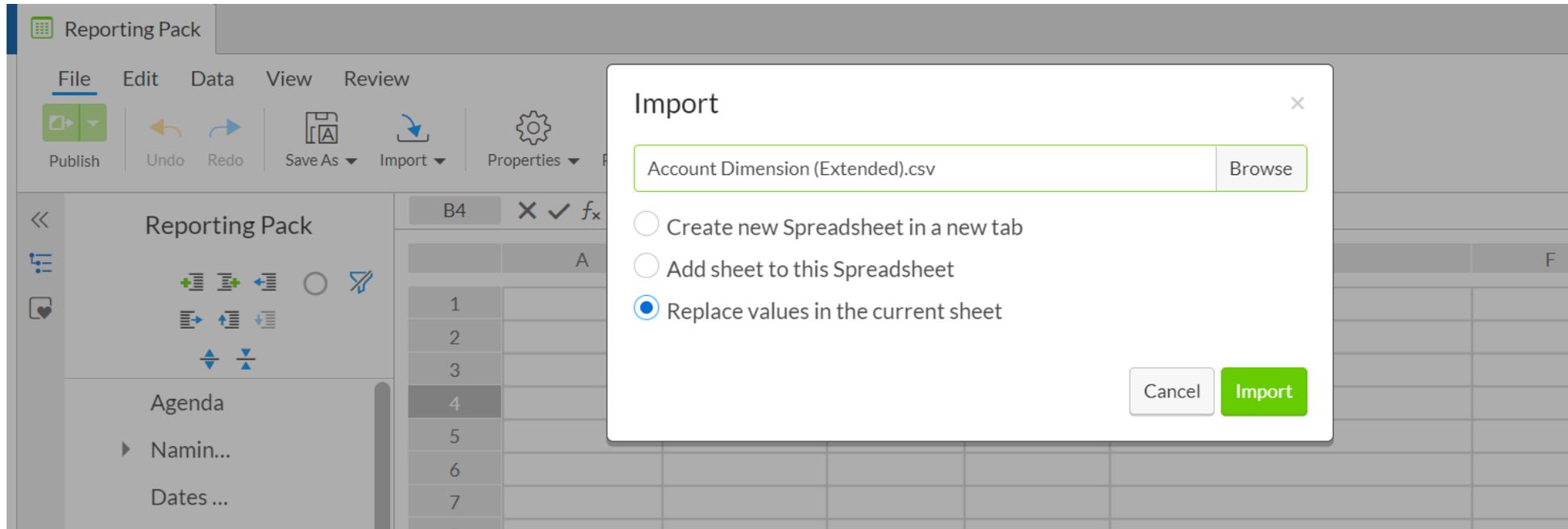
Tag Name Tag Value

Select Tag Name Select Tag Value

Cancel Add Data

Once the table is created, you would be directed to the page where you would be required to 'Add Dataset'. Simply add the .csv file that was uploaded previously.

Step 5.4: Creating the Extended COA P1



In order to create the staging query for the budget data, one would need to append the budget account codes in the COA DIM Table. To achieve this, import the file 'Account Dimension (Extended).csv' file into a blank Wdesk spreadsheet in the Reporting Pack. Name the sheet as "Extended COA". (Note: in most cases, the budget accounts are manually input by the users)

Step 5.5: Creating the Extended COA P2

The image illustrates the process of creating a Wdata connection in three sequential steps:

- Step 1:** In the 'Wdata Connections' panel, the 'Add Connection' button is highlighted with a red box.
- Step 2:** The 'Add Connection to Wdata Table' dialog is shown. The table '#WK_DIM_Account' is selected in the 'Tables' list, highlighted with a red box. The 'Next' button is also highlighted with a red box.
- Step 3:** The 'Add Connection to Wdata Table' dialog is shown again. The 'Dataset name' field is filled with 'Extended COA'. The 'Source' is 'Run_Sheet_SA Workspace / Extended COA'. The 'Finish' button is highlighted with a red box.

In the Spreadsheet, click the connection icon on the right panel and click add connection. Then, select table "#WK_DIM_Account" and click Next.

In the next page, click Finish and the Extended COA sheet is now appended to the COA DIM Table.

Step 5.6: Creating Product DIM Table

The screenshot shows a data management interface with a table being created. The table has the following columns:

COLUMN NAME	DESCRIPTION	COLUMN ID	COLUMN TYPE	IMPORT FORMAT	KEY
productcode	Enter column description	productcode	Text		<input checked="" type="checkbox"/>
productname	Enter column description	productname	Text		<input type="checkbox"/>
fmproductassetclass	Enter column description	fmproductassetclass	Text		<input type="checkbox"/>
fmproducttype	Enter column description	fmproducttype	Text		<input type="checkbox"/>
fmproductgeography	Enter column description	fmproductgeography	Text		<input type="checkbox"/>
fmproductclass	Enter column description	fmproductclass	Text		<input type="checkbox"/>
productru	Enter column description	productru	Text		<input type="checkbox"/>

The interface also shows a 'Description' field with the text 'Enter Table Description' and a 'Datasets' panel on the right with the file 'Product Dimension.csv' and a last update timestamp of 'Mar 18, 2021 4:40 PM by Xuan Zhi Choo'.

The next table that would be created is a Product Dimension table that stores the Product Mapping data, setting the 'Product Code' as the primary key.

Step 5.7: Creating Cost Center DIM Table

The screenshot shows a data management interface with the following components:

- Navigation:** #WK_DIM_Account, #WK_DIM_Product, #WK_DIM_Cost Center
- Actions:** Save, Preview, Add Column, Permissions, Rename, Feedback, Help, Tours
- Description:** Enter Table Description
- Table Structure:**

COLUMN NAME	DESCRIPTION	COLUMN ID ?	COLUMN TYPE ?	IMPORT FORMAT	KEY ?
accountcode	Enter column description	accountcode	Text		<input checked="" type="checkbox"/>
costcentercode	Enter column description	costcentercode	Text		<input type="checkbox"/>
costcentername	Enter column description	costcentername	Text		<input type="checkbox"/>
ccru2	Enter column description	ccru2	Text		<input type="checkbox"/>
ccru3	Enter column description	ccru3	Text		<input type="checkbox"/>
ccru4	Enter column description	ccru4	Text		<input type="checkbox"/>
ccru5	Enter column description	ccru5	Text		<input type="checkbox"/>

Datasets: Add Dataset, Cost Center Dimension.csv (Last Update: Mar 18, 2021 2:33 PM by Xuan Zhi Choo)

> Table Preview

The next table that would be created is a Cost Center Dimension table that stores the Cost Center data, setting the 'GL Account' as the primary key.

Step 6: Creating TB Source (SRC) Table

#WK_SRC_TB_Sys

Save Preview Add Column Permissions Rename Feedback Help Tours

Description: Enter Table Description

COLUMN NAME	DESCRIPTION	COLUMN ID ?	COLUMN TYPE ?	IMPORT FORMAT
year	Enter column description	year	Text 99	
month	Enter column description	month	Text 99	
entity	Enter column description	entity	Text 99	
account_code	Enter column description	account_code	Text 99	
description	Enter column description	description	Text 99	
forward_balance	Enter column description	forward_balance	Decimal 100.01	
debit	Enter column description	debit	Decimal 100.01	
credit	Enter column description	credit	Decimal 100.01	
ending_balance	Enter column description	ending_balance	Decimal 100.01	
product_code	Enter column description	product_code	Text 99	

> Table Preview

Datasets

Add Dataset

- TB_12_2019.csv
Last Update: Mar 19, 2021 1:52 PM by Pankit Dhawan
Tags: Data_Month: Dec-12, Data_Year: 2019
- TB_11_2020.csv
Last Update: Mar 19, 2021 1:52 PM by Pankit Dhawan
Tags: Data_Month: Nov-11, Data_Year: 2020
- TB_12_2020.csv
Last Update: Mar 19, 2021 1:51 PM by Pankit Dhawan
Tags: Data_Month: Dec-12, Data_Year: 2020

Creating a 'Data' type TB Table which would store data from the client's TB data, importing PY & CY Trial Balance data.

Step 7: Objective for Creating Staging Query

↪ Consumption query helps us derive the month balances for both the Profit & Loss & Balance Sheet

P&L

- Always represented in number incurred for the month.
- For our example data, ending balance represents the closing balance for the months.
- Calc Representation Example:

FS Mapping	Jan_YTD	Feb_YTD	Mar_YTD	Apr_YTD	May_YTD
PL	Jan	Jan + Feb	Jan + Feb + Mar	Jan + Feb + Mar + Apr	...

BS

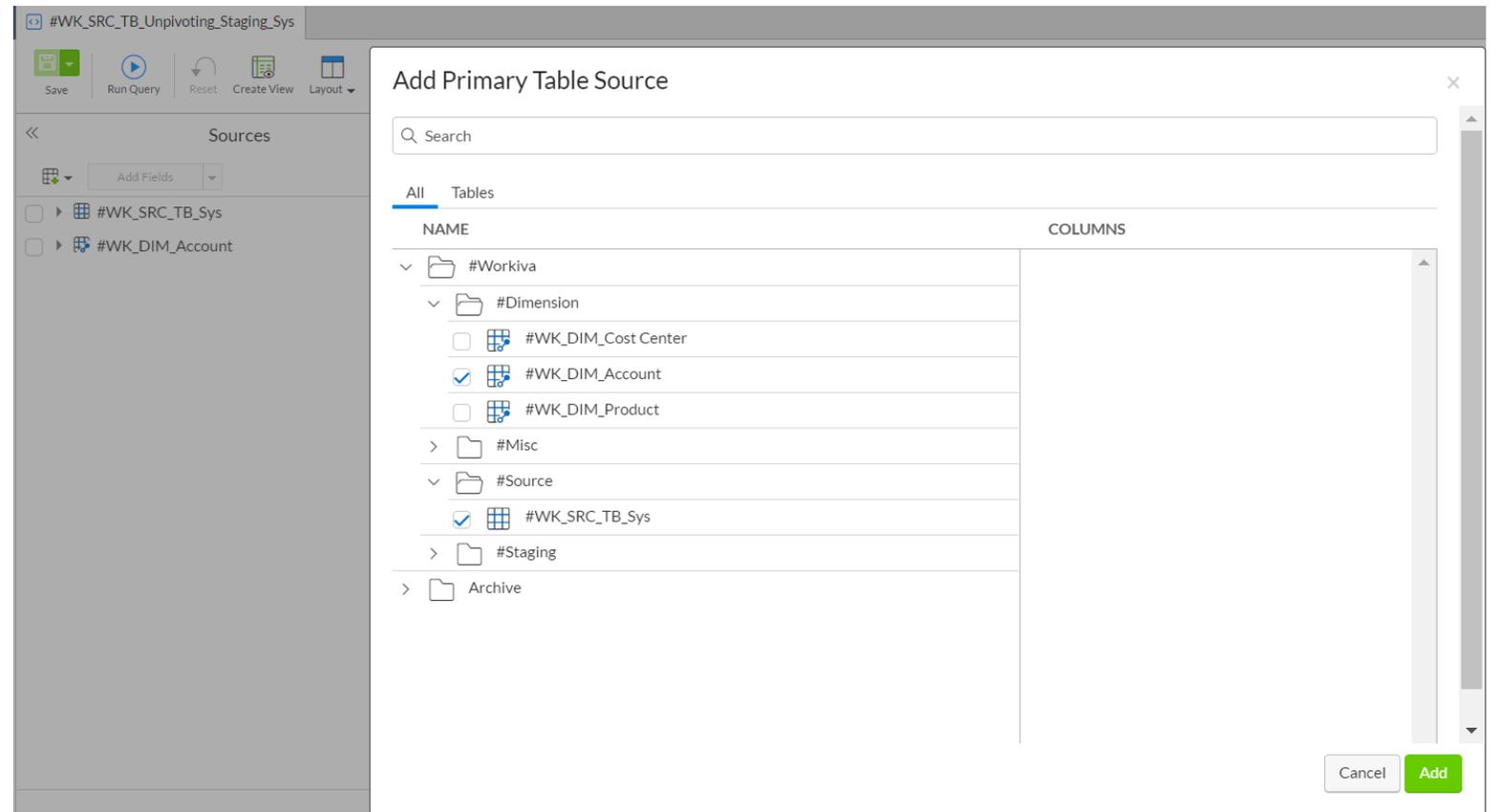
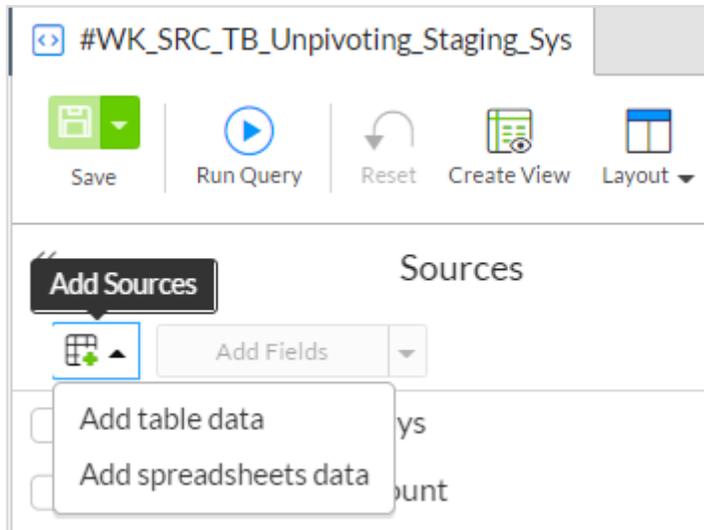
- Always represented in closing balance numbers.
- For our example data, ending balance represents the closing balance for the months.
- Calc Representation Example:

FS Mapping	Jan	Feb	Mar	Apr	May
BS	Jan	Feb	Mar	Apr	...

Step 7.1: Steps for creating the Staging Query

Steps	Description
1	First, build the query for Current Year.
2	Based on the Current Year query, replicate the query to retrieve Prior Year's data.
3	Do a 'Full Outer Join' for both Current & Prior Year's query.
4	As sometimes current year would not have data, we would use "Advance SQL" mode to get rid of the metadata to make it more presentable.
5	The following steps would be illustrated in the slides below.
6	Naming Convention: i) Queries & Tables with '_Sys' at the back of the name are system based, and should not be editable by users ii) Queries & Tables with '_User' at the back of the name are user type query, and should be editable by users

Step 7.2: Creating TB Staging Query - Adding table data source



To create a new query, click on the 'Create' -> 'Query'. Upon creation of a new query from the left panel, add all the relevant data tables that would be used in the query. Name the query as "#WK_SRC_TB_Unpivoting_Staging_Sys". Note: Always hit 'Save'!

Step 8: Establishing data table relationship

The screenshot shows a data tool interface with a 'Sources' panel on the left and a 'Relationships' panel on the right. The 'Sources' panel lists two tables: '#WK_SRC_TB_Sys' and '#WK_DIM_Account'. The 'Relationships' panel shows a relationship being established between the 'account_code' column of '#WK_SRC_TB_Sys' and the 'gl_account' column of '#WK_DIM_Account'. The relationship type is set to 'LEFT JOIN'. Below this, there are two dashed boxes with the text 'Drop column to add to relationship' and a 'LEFT JOIN' dropdown menu, indicating that the relationship is not yet fully defined.

As there are 2 data tables, a relationship would need to be established between the "#WK_SRC_TB_Sys" & "#WK_Dim_Account" tables via a common identifier. The common identifier is the 'GL Account' code

Step 9: Building Fields for the Query

#WK_SRC_TB_Unpivoting_Staging_Sys0

Save Run Query Reset Create View Layout Export Permissions Properties Feedback Help Tours

Sources

Add to Fields

fx Calculation

#WK_SRC_TB_Sys

#WK_DIM_Account

Fields Filters Sort Relationships

Builder SQL

HEADER	SOURCE	SOURCE COLUMN
year	#WK_SRC_TB_Sys	year
entity	#WK_SRC_TB_Sys	entity
account_code	#WK_SRC_TB_Sys	account_code
description	#WK_SRC_TB_Sys	description
Jan	#WK_SRC_TB_Sys	Multiple
Feb	#WK_SRC_TB_Sys	Multiple
Mar	#WK_SRC_TB_Sys	Multiple
Apr	#WK_SRC_TB_Sys	Multiple
May	#WK_SRC_TB_Sys	Multiple
Jun	#WK_SRC_TB_Sys	Multiple
Jul	#WK_SRC_TB_Sys	Multiple

Drop column to include in query results

Query Syntax is Good

To insert fields in the query, tick the intended columns and drag it to the space under 'Fields'. In this step, the query would only be extracting Current Year's values. The following steps will illustrate how the Prior Year fields will be setup.

Step 9.1: Inserting Calculation Fields & Month Formula P1

The screenshot shows the interface for creating a calculation field. The 'Sources' panel on the left lists various fields, including 'Calculation', 'YEAR', 'MONTH', 'ENTITY', 'ACCOUNT_CODE', 'DESCRIPTION', 'FORWARD_BALANCE', 'DEBIT', 'CREDIT', 'ENDING_BALANCE', 'PRODUCT_CODE', 'TAGS', 'FILENAME', 'TIMESTAMP', 'USERID', and '#WK_DIM_Account'. The 'Fields' panel in the center displays a table with columns 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The 'Field Properties' panel on the right shows the configuration for the 'Calculation' field, including the SQL code and the 'Included Columns' section. The 'Included Columns' section is highlighted with a red box and contains the following fields: 1. month / #WK_SRC_TB_Sys, 2. ending_balance / #WK_SRC_TB_Sys, 3. Drop columns to include here. The 'Apply' button is visible at the bottom right of the Field Properties panel.

To calculate for 'Jan' monthly amount field, drag 'Calculation' into the Fields and name it 'Jan'.

Drag the " month, ending_balance" into the 'Included Columns'. With an SUM CASE formula, we instruct the code to identify the particular month (i.e. January) and display the balance for that month.

Step 9.2: Inserting Calculation Fields & Month Formula P2

The screenshot displays a data tool interface with a table of months and their corresponding calculation fields. The table has columns for 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The 'SOURCE COLUMN' column contains the text 'Multiple' and a formula icon 'fx'. The 'SOURCE' column contains the text '#WK_SRC_TB_Sys'. The 'HEADER' column contains the months from Feb to Dec. A 'Field Properties' panel on the right shows the calculation formula: `SUM(CASE WHEN {1} = '12' THEN {2} ELSE 0 END)`. A second 'Field Properties' panel on the far right shows the same formula with '12' highlighted in yellow, indicating a change to '2' for February.

HEADER	SOURCE	SOURCE COLUMN
Feb	#WK_SRC_TB_Sys	Multiple fx
Mar	#WK_SRC_TB_Sys	Multiple fx
Apr	#WK_SRC_TB_Sys	Multiple fx
May	#WK_SRC_TB_Sys	Multiple fx
Jun	#WK_SRC_TB_Sys	Multiple fx
Jul	#WK_SRC_TB_Sys	Multiple fx
Aug	#WK_SRC_TB_Sys	Multiple fx
Sep	#WK_SRC_TB_Sys	Multiple fx
Oct	#WK_SRC_TB_Sys	Multiple fx
Nov	#WK_SRC_TB_Sys	Multiple fx
Dec	#WK_SRC_TB_Sys	Multiple fx

Field Properties (Feb):
Header: Feb
Calculation: `SUM(CASE WHEN {1} = '2' THEN {2} ELSE 0 END)`

Field Properties (Dec):
Header: Dec
Calculation: `SUM(CASE WHEN {1} = '12' THEN {2} ELSE 0 END)`

To calculate monthly amount fields for the remaining months, duplicate the "Jan" column and change the part of the calculation as highlighted above. (i.e. if we are creating the field for Feb, change from 1 to 2) Create the monthly amount fields for 12 months.

Step 10: Inserting Calculation Fields & YTD Formula P1

The screenshot shows the Workiva Query Builder interface. On the left, the 'Sources' panel lists two data sources: '#WK_SRC_TB_Sys' and '#WK_DIM_Account'. The '#WK_DIM_Account' source is expanded, showing fields like 'GLACCOUNT', 'GLDESCRIPTION', 'FS', and 'FS1'. The 'Fields' panel in the center displays a list of calculation fields for each month from 'Dec' to 'Dec_YTD'. The 'Jan_YTD' field is selected, and its 'Field Properties' are shown on the right. The 'Header' is 'Jan_YTD'. The 'Calculation' field is defined with the following SQL code:

```
1 IF ({1} = 'PL', SUM(CASE
2   WHEN {2} = '1' THEN {3}
3   ELSE 0
4   END
5   ), SUM(CASE
6   WHEN {2} = '1' THEN {3}
7   ELSE 0
8   END
9   ))
```

The 'Included Columns' section on the right lists three columns:

- fs / #WK_DIM_Account
- month / #WK_SRC_TB_Sys
- ending_balance / #WK_SRC_TB_Sys

Red arrows point from the 'Calculation' field in the Fields panel to the 'Field Properties' panel. Blue arrows point from the 'fs', 'month', and 'ending_balance' fields in the Sources panel to their respective entries in the 'Included Columns' list.

```
IF({1} = 'PL',
SUM(CASE
  WHEN {2} = '1'
  THEN {3}
  ELSE 0
END
), SUM(CASE
  WHEN {2} = '1'
  THEN {3}
  ELSE 0
END
))
```

To calculate for 'Jan_YTD', drag 'Calculation' into the Fields and name it 'Jan_YTD'. Drag the "fs, month, ending_balance" into the 'Included Columns'. With an IF formula, we instruct the code to identify whether it is a PL/BS item, and if the month is January, display the balance for January.

Step 10.1: Inserting Calculation Fields & YTD Formula P2

The screenshot shows the Workiva query builder interface. The 'Fields' tab is active, displaying a list of fields with columns for 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The 'Feb_YTD' field is selected. The 'Calculation' pane shows the following SQL formula:

```
1 IF({1} = 'PL', SUM(CASE
2   WHEN {2} = '1' OR
3     {2} = '2' THEN {3}
4   ELSE 0
5   END
6   ), SUM(CASE
7     WHEN {2} = '2' THEN {3}
8     ELSE 0
9     END
10  ))
```

The 'Field Properties' pane for 'Dec_YTD' shows the following SQL formula:

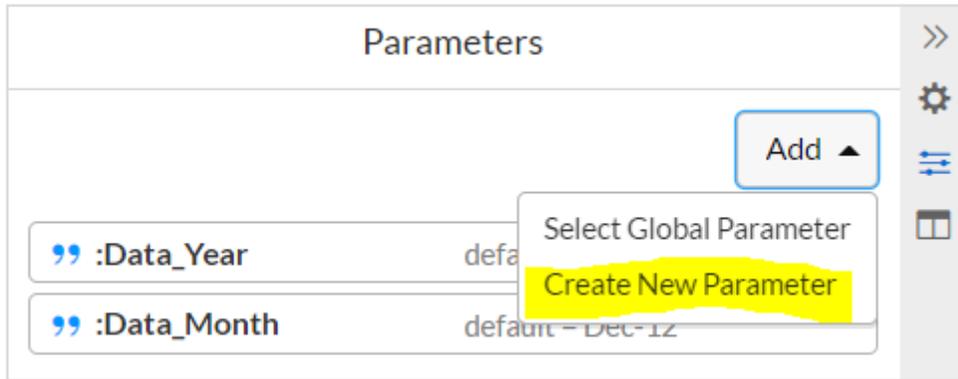
```
1 IF({1} = 'PL', SUM(CASE
2   WHEN {2} = '1' OR
3     {2} = '3' OR
4     {2} = '4' OR
5     {2} = '5' OR
6     {2} = '6' OR
7     {2} = '7' OR
8     {2} = '8' OR
9     {2} = '9' OR
10    {2} = '10' OR
11    {2} = '11' OR
12    {2} = '12' THEN {3}
13   ELSE 0
14   END
15   ), SUM(CASE
16     WHEN {2} = '12' THEN {3}
17     ELSE 0
18     END
19  ))
```

The 'Included Columns' section lists:

- fs / #WK_DIM_Account
- month / #WK_SRC_TB_Sys
- ending_balance / #WK_SRC_TB_Sys

To calculate monthly YTD amount fields for the remaining months, duplicate the "Jan_YTD" column and change the part of the calculation as highlighted above. (i.e. if we are creating the field for Feb, add in another OR statement) Create the monthly YTD amount fields for 12 months.

Step 11: Creating Parameters



Parameters

< Edit Parameter

Name

Data_Year

SQL Name: Data_Year

Type

Text

Pick List

Multi-Select

List Options ?

2018
2019
2020
2021

Default Value

2020

Cancel Apply

Click on the Parameters icon at the top right of the screen, create 2 Parameters ('Data_Year' & 'Data_Month'). Ensure that the parameter names matches the Tags name which was created in Step 0.

Step 11.1: Creating Filters

The screenshot displays a data tool interface with the following components:

- Top Bar:** Includes icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours.
- Sources Panel:** Lists various data sources and fields. The 'TAGS' field is highlighted with a red arrow.
- Fields, Filters, Sort, Relationships:** The 'Filters' tab is active, showing two filter rows:
 - Row 1: #WK_SRC_TB_Sys, TAGS [Data_Year], =, None
 - Row 2: #WK_SRC_TB_Sys, TAGS [Data_Month], =, None
 - Row 3: Drop column to filter query results by, Equal To
- Filter Properties Panel:** Shows the configuration for the first filter. The 'Column' is set to '_tags'. The 'Calculation' dropdown is open, showing options: ':Data_Year', 'Data_Year', 'Data_Month', 'Create New Parameter', and 'Select Global Parameter'. A red arrow points to the ':Data_Year' option.
- Status Bar:** Displays 'Query Syntax is Good'.

Create 2 filters ('Data_Year' & 'Data_Month'). Drag "Tags" into the 'Filters' and assign Filter values where 'Data_Year' = ":Data_Year" ; 'Data_Month' = ":Data_Month"

Step 12: Replicating for Prior Year's Data P1

The screenshot shows the Workiva query builder interface. The top navigation bar includes tabs for '#WK_SRC_TB_Unpivot_Staging_Sys', 'CY', and 'PY'. The main workspace is divided into three sections: Sources, Fields, and Query Properties.

Sources: A list of data sources is shown on the left, including 'Calculation', '#WK_SRC_TB_Sys', and '#WK_DIM_Account'.

Fields: A table of selected fields is displayed in the center. The table has three columns: 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The fields are:

HEADER	SOURCE	SOURCE COLUMN
PY_year	#WK_SRC_TB_Sys	year
PY_entity	#WK_SRC_TB_Sys	entity
PY_account_code	#WK_SRC_TB_Sys	account_code
PY_description	#WK_SRC_TB_Sys	description
PY_product_code	#WK_SRC_TB_Sys	product_code
PY_FS	#WK_DIM_Account	fs
PY_Jan	#WK_SRC_TB_Sys	Multiple
PY_Feb	#WK_SRC_TB_Sys	Multiple
PY_Mar	#WK_SRC_TB_Sys	Multiple
PY_Apr	#WK_SRC_TB_Sys	Multiple
PY_May	#WK_SRC_TB_Sys	Multiple

Below the table, there is a prompt: "Drop column to include in query results".

Query Properties: The right-hand panel shows the configuration for the query named 'PY'. It includes fields for 'Query name' (PY), 'Query description' (Enter query description), and 'Limit' (1000000). There is also a checkbox for 'Show only distinct rows' which is currently unchecked. At the bottom, it shows 'Last Run' (4/8/2021 10:17 AM), 'Total Records Returned' (200), 'Data Scanned' (237.71 KiB), and 'Query Run Time' (1.243 seconds).

A status bar at the bottom of the interface indicates "Query Syntax is Good".

After creating all the necessary columns, save this query as "CY". Copy the CY query and name it as "PY". In the PY query, change the front naming convention to begin with 'PY'; e.g. "PY_year".

Step 12.1: Replicating for Prior Year's Data P2

The screenshot displays a data tool interface with a top toolbar containing icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours. The main workspace is divided into several panels:

- Sources:** Lists available data sources: Calculation, #WK_SRC_TB_Sys, and #WK_DIM_Account.
- Fields, Filters, Sort, Relationships:** The 'Filters' tab is active, showing a filter expression: `{1} AND {2}`. Below this, a table lists filter conditions:

Filter ID	Field	Operator	Value
1	#WK_SRC_TB_Sys TAGS [Data_Year]	=	None
2	#WK_SRC_TB_Sys TAGS [Data_Mon...]	Equal To	'Dec-12'
3	Drop column to filter query results by	Equal To	
- Filter Properties:** Shows the selected filter's configuration:
 - Source: #WK_SRC_TB_Sys
 - Column: *_tags
 - Calculated: Yes
 - Calculation: `CAST(CAST(:Data_Year AS INTEGER) - 1 AS VARCHAR)`

A status bar at the bottom indicates "Query Syntax is Good" with a green checkmark. An "Apply" button is located at the bottom right of the Filter Properties panel.

In the PY query, go to Filters and change the Filter Value for 'Data_Year' with the calculation highlighted above - 'CAST (CAST(:Data_Year AS INTEGER) -1 AS VARCHAR)'. Set the filter value for 'Data_Month' to always be as "Dec-12".

In here, we are essentially adding on the "WHERE" clause for the Prior Year's query to cater for prior year.

Step 13.1: Joining Current & Prior Year Query P2

SQL Editor ?

Builder SQL

```
546 "#WK_SRC_TB_Sys"."month" = '2' or
547 "#WK_SRC_TB_Sys"."month" = '3' or
548 "#WK_SRC_TB_Sys"."month" = '4' or
549 "#WK_SRC_TB_Sys"."month" = '5' or
550 "#WK_SRC_TB_Sys"."month" = '6' or
551 "#WK_SRC_TB_Sys"."month" = '7' or
552 "#WK_SRC_TB_Sys"."month" = '8' or
553 "#WK_SRC_TB_Sys"."month" = '9' or
554 "#WK_SRC_TB_Sys"."month" = '10' or
555 "#WK_SRC_TB_Sys"."month" = '11' or
556 "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
557 ELSE 0
558 END
559 ), SUM(CASE
560 WHEN "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
561 ELSE 0
562 END
563 )) AS "PY_Dec_YTD"
564
565 FROM
566 "QWNjb3VudB8xOTA1MjA5MjE1"."e36418c3a92846debcbbb3f3708a555" AS "#WK_SRC_TB_Sys"
567 LEFT JOIN
568 "QWNjb3VudB8xOTA1MjA5MjE1"."817b61205a154bd98c835c294cd33240" AS "#WK_DIM_Account"
569 ON "#WK_SRC_TB_Sys"."account_code" = "#WK_DIM_Account"."gl_account"
570
571 WHERE
572 "#WK_SRC_TB_Sys"._tags['Data_Year'] = CAST(CAST(:Data_Year AS INTEGER) - 1 AS VARCHAR) AND
573 "#WK_SRC_TB_Sys"._tags['Data_Month'] = 'Dec-12'
574
575 GROUP BY "#WK_SRC_TB_Sys"."year", "#WK_SRC_TB_Sys"."entity", "#WK_SRC_TB_Sys"."account_code",
576 "#WK_SRC_TB_Sys"."description", "#WK_SRC_TB_Sys"."product_code", "#WK_DIM_Account"."fs",
577 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
578 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
579 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs"
580
581 ORDER BY "PY_account_code" DESC ) PY
582
583 ON CONCAT (CY."entity" , CY."account_code" , CY."product_code") = CONCAT( PY."PY_entity" , PY
584 ."PY_account_code" , PY."PY_product_code")
```

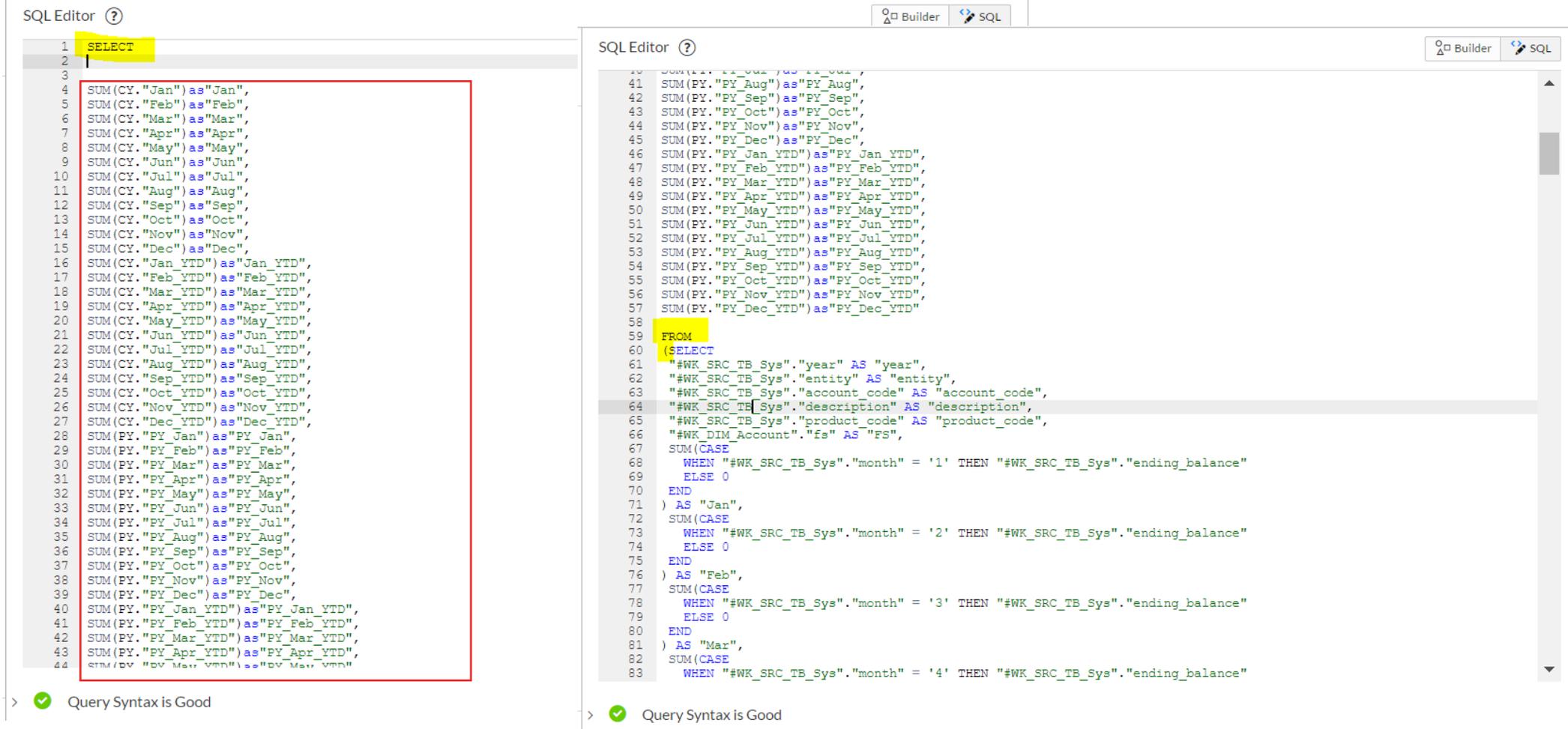
>  Query Syntax is Good

Go to the bottom of the combined query and delete the last row "LIMIT 1000000".

Add in the code highlighted on the left.

In here, we are joining the two sets of codes based on the concatenation of 'entity' & 'GL Account Code'.

Step 14: Unpivoting Months Columns



The image shows two screenshots of a SQL Editor interface. The left screenshot displays a long list of SQL queries, with the first 48 lines highlighted in a red box. These lines represent the SUM statements for each month and YTD for both Current Year (CY) and Prior Year (PY). The right screenshot shows the same SQL Editor with a FROM clause and a subquery that selects columns from a source table and uses CASE statements to pivot the data by month.

```
SQL Editor ?
```

```
1 SELECT
2
3
4 SUM(CY."Jan") as "Jan",
5 SUM(CY."Feb") as "Feb",
6 SUM(CY."Mar") as "Mar",
7 SUM(CY."Apr") as "Apr",
8 SUM(CY."May") as "May",
9 SUM(CY."Jun") as "Jun",
10 SUM(CY."Jul") as "Jul",
11 SUM(CY."Aug") as "Aug",
12 SUM(CY."Sep") as "Sep",
13 SUM(CY."Oct") as "Oct",
14 SUM(CY."Nov") as "Nov",
15 SUM(CY."Dec") as "Dec",
16 SUM(CY."Jan_YTD") as "Jan_YTD",
17 SUM(CY."Feb_YTD") as "Feb_YTD",
18 SUM(CY."Mar_YTD") as "Mar_YTD",
19 SUM(CY."Apr_YTD") as "Apr_YTD",
20 SUM(CY."May_YTD") as "May_YTD",
21 SUM(CY."Jun_YTD") as "Jun_YTD",
22 SUM(CY."Jul_YTD") as "Jul_YTD",
23 SUM(CY."Aug_YTD") as "Aug_YTD",
24 SUM(CY."Sep_YTD") as "Sep_YTD",
25 SUM(CY."Oct_YTD") as "Oct_YTD",
26 SUM(CY."Nov_YTD") as "Nov_YTD",
27 SUM(CY."Dec_YTD") as "Dec_YTD",
28 SUM(PY."PY_Jan") as "PY_Jan",
29 SUM(PY."PY_Feb") as "PY_Feb",
30 SUM(PY."PY_Mar") as "PY_Mar",
31 SUM(PY."PY_Apr") as "PY_Apr",
32 SUM(PY."PY_May") as "PY_May",
33 SUM(PY."PY_Jun") as "PY_Jun",
34 SUM(PY."PY_Jul") as "PY_Jul",
35 SUM(PY."PY_Aug") as "PY_Aug",
36 SUM(PY."PY_Sep") as "PY_Sep",
37 SUM(PY."PY_Oct") as "PY_Oct",
38 SUM(PY."PY_Nov") as "PY_Nov",
39 SUM(PY."PY_Dec") as "PY_Dec",
40 SUM(PY."PY_Jan_YTD") as "PY_Jan_YTD",
41 SUM(PY."PY_Feb_YTD") as "PY_Feb_YTD",
42 SUM(PY."PY_Mar_YTD") as "PY_Mar_YTD",
43 SUM(PY."PY_Apr_YTD") as "PY_Apr_YTD",
44 SUM(PY."PY_May_YTD") as "PY_May_YTD",
45 SUM(PY."PY_Jun_YTD") as "PY_Jun_YTD",
46 SUM(PY."PY_Jul_YTD") as "PY_Jul_YTD",
47 SUM(PY."PY_Aug_YTD") as "PY_Aug_YTD",
48 SUM(PY."PY_Sep_YTD") as "PY_Sep_YTD",
49 SUM(PY."PY_Oct_YTD") as "PY_Oct_YTD",
50 SUM(PY."PY_Nov_YTD") as "PY_Nov_YTD",
51 SUM(PY."PY_Dec_YTD") as "PY_Dec_YTD",
52
53 FROM
54 (SELECT
55   "#WK_SRC_TB_Sys"."year" AS "year",
56   "#WK_SRC_TB_Sys"."entity" AS "entity",
57   "#WK_SRC_TB_Sys"."account code" AS "account code",
58   "#WK_SRC_TB_Sys"."description" AS "description",
59   "#WK_SRC_TB_Sys"."product code" AS "product code",
60   "#WK_DIM_Account"."Es" AS "FS",
61   SUM(CASE
62     WHEN "#WK_SRC_TB_Sys"."month" = '1' THEN "#WK_SRC_TB_Sys"."ending_balance"
63     ELSE 0
64   ) AS "Jan",
65   SUM(CASE
66     WHEN "#WK_SRC_TB_Sys"."month" = '2' THEN "#WK_SRC_TB_Sys"."ending_balance"
67     ELSE 0
68   ) AS "Feb",
69   SUM(CASE
70     WHEN "#WK_SRC_TB_Sys"."month" = '3' THEN "#WK_SRC_TB_Sys"."ending_balance"
71     ELSE 0
72   ) AS "Mar",
73   SUM(CASE
74     WHEN "#WK_SRC_TB_Sys"."month" = '4' THEN "#WK_SRC_TB_Sys"."ending_balance"
```

Query Syntax is Good

Then, at the top, we add in the SUM statements as shown above. Also add in the code as highlighted. What we are trying to achieve here is to unpivot and sum the 48 monthly columns (Jan to Dec movement + YTD Jan to YTD Dec) for Current & Prior Year.

Step 15: Tips for Efficient Query Building

A	B	C	D	E
CY.	"	Jan)	as "Jan", SUM(CY."Jan") as "Jan",
CY.	"	Feb)	as "Feb", SUM(CY."Feb") as "Feb",
CY.	"	Mar)	as "Mar", SUM(CY."Mar") as "Mar",
CY.	"	Apr)	as "Apr", SUM(CY."Apr") as "Apr",
CY.	"	May)	as "May", SUM(CY."May") as "May",
CY.	"	Jun)	as "Jun", SUM(CY."Jun") as "Jun",
CY.	"	Jul)	as "Jul", SUM(CY."Jul") as "Jul",
CY.	"	Aug)	as "Aug", SUM(CY."Aug") as "Aug",
CY.	"	Sep)	as "Sep", SUM(CY."Sep") as "Sep",
CY.	"	Oct)	as "Oct", SUM(CY."Oct") as "Oct",
CY.	"	Nov)	as "Nov", SUM(CY."Nov") as "Nov",
CY.	"	Dec)	as "Dec", SUM(CY."Dec") as "Dec",
CY.	"	Jan_YTD)	as "Jan_YTD", SUM(CY."Jan_YTD") as "Jan_YTD",
CY.	"	Feb_YTD)	as "Feb_YTD", SUM(CY."Feb_YTD") as "Feb_YTD",
CY.	"	Mar_YTD)	as "Mar_YTD", SUM(CY."Mar_YTD") as "Mar_YTD",
CY.	"	Apr_YTD)	as "Apr_YTD", SUM(CY."Apr_YTD") as "Apr_YTD",
CY.	"	May_YTD)	as "May_YTD", SUM(CY."May_YTD") as "May_YTD",
CY.	"	Jun_YTD)	as "Jun_YTD", SUM(CY."Jun_YTD") as "Jun_YTD",
CY.	"	Jul_YTD)	as "Jul_YTD", SUM(CY."Jul_YTD") as "Jul_YTD",
CY.	"	Aug_YTD)	as "Aug_YTD", SUM(CY."Aug_YTD") as "Aug_YTD",
CY.	"	Sep_YTD)	as "Sep_YTD", SUM(CY."Sep_YTD") as "Sep_YTD",

```

Select
Case when CY."year" is NULL then cast (Cast(PY."PY_year" as INTEGER) + 1 as VARCHAR) else CY."year" end as "Current_Year",
Case when CY."entity" is NULL then PY."PY_entity" else CY."entity" end as "Entity",
Case when CY."account_code" is NULL then PY."PY_account_code" else CY."account_code" end as "Account_Code",
Case when CY."description" is NULL then PY."PY_description" else CY."description" end as "Description",
Case when CY."FS" is NULL then PY."PY_FS" else CY."FS" end as "FS",

SUM(CY."Jan") as "Jan",
SUM(CY."Feb") as "Feb",
SUM(CY."Mar") as "Mar",
SUM(CY."Apr") as "Apr",
SUM(CY."May") as "May",
SUM(CY."Jun") as "Jun",
SUM(CY."Jul") as "Jul",
SUM(CY."Aug") as "Aug",
SUM(CY."Sep") as "Sep",
SUM(CY."Oct") as "Oct",
SUM(CY."Nov") as "Nov",
SUM(CY."Dec") as "Dec",
SUM(CY."Jan_YTD") as "Jan_YTD",
SUM(CY."Feb_YTD") as "Feb_YTD",
SUM(CY."Mar_YTD") as "Mar_YTD",
SUM(CY."Apr_YTD") as "Apr_YTD",
SUM(CY."May_YTD") as "May_YTD",
SUM(CY."Jun_YTD") as "Jun_YTD",
SUM(CY."Jul_YTD") as "Jul_YTD",
SUM(CY."Aug_YTD") as "Aug_YTD",
SUM(CY."Sep_YTD") as "Sep_YTD",
SUM(CY."Oct_YTD") as "Oct_YTD",
SUM(CY."Nov_YTD") as "Nov_YTD",
SUM(CY."Dec_YTD") as "Dec_YTD",
SUM(PY."PY_Jan") as "PY_Jan",
SUM(PY."PY_Feb") as "PY_Feb",
SUM(PY."PY_Mar") as "PY_Mar",
SUM(PY."PY_Apr") as "PY_Apr",
SUM(PY."PY_May") as "PY_May",
SUM(PY."PY_Jun") as "PY_Jun",
SUM(PY."PY_Jul") as "PY_Jul",
SUM(PY."PY_Aug") as "PY_Aug",
SUM(PY."PY_Sep") as "PY_Sep",
SUM(PY."PY_Oct") as "PY_Oct",
SUM(PY."PY_Nov") as "PY_Nov",
SUM(PY."PY_Dec") as "PY_Dec",

```

Excel formulas can be leveraged when building & structuring queries. As illustrated above, we utilised excel formulas to build the SUM for CY & PY's representation.

Step 16: Grouping - Group By

CY query

```
290 #WK_SRC_TB_Sys"."month" = '9' OR
291 "#WK_SRC_TB_Sys"."month" = '10' OR
292 "#WK_SRC_TB_Sys"."month" = '11' OR
293 "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
294 ELSE 0
295 END
296 ), SUM(CASE
297 WHEN "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
298 ELSE 0
299 END
300 )) AS "Dec_YTD"
301
302 FROM
303 "QWNjb3VudB8xODkzNjM5MjMw"."6f6e4085dc6942c5b9543cb85a5ad085" AS "#WK_SRC_TB_Sys"
304 LEFT JOIN
305 "QWNjb3VudB8xODkzNjM5MjMw"."e450aa655ad1401280f8c9f804834433" AS "#WK_DIM_Account"
306 ON "#WK_SRC_TB_Sys"."account_code" = "#WK_DIM_Account"."gl_account"
307
308 WHERE
309 "#WK_SRC_TB_Sys"._tags['Data_Year'] = :Data_Year AND
310 "#WK_SRC_TB_Sys"._tags['Data_Month'] = :Data_Month
311
312 GROUP BY "#WK_SRC_TB_Sys"."year", "#WK_SRC_TB_Sys"."entity", "#WK_SRC_TB_Sys"."account_code", "#WK_SRC_TB_Sys"
313 ".description", "#WK_SRC_TB_Sys"."product_code", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
314 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
315 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
316 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs"
317
318 ORDER BY "account_code" ASC, "FS" ASC ) CY
319
320 FULL OUTER JOIN
321 (SELECT
322 "#WK_SRC_TB_Sys"."year" AS "PY_year",
323 "#WK_SRC_TB_Sys"."entity" AS "PY_entity",
324 "#WK_SRC_TB_Sys"."account_code" AS "PY_account_code",
```

PY query

```
539 IF("#WK_DIM_Account"."fs" = 'PL', SUM(CASE
540 WHEN "#WK_SRC_TB_Sys"."month" = '1' OR
541 "#WK_SRC_TB_Sys"."month" = '2' OR
542 "#WK_SRC_TB_Sys"."month" = '3' OR
543 "#WK_SRC_TB_Sys"."month" = '4' OR
544 "#WK_SRC_TB_Sys"."month" = '5' OR
545 "#WK_SRC_TB_Sys"."month" = '6' OR
546 "#WK_SRC_TB_Sys"."month" = '7' OR
547 "#WK_SRC_TB_Sys"."month" = '8' OR
548 "#WK_SRC_TB_Sys"."month" = '9' OR
549 "#WK_SRC_TB_Sys"."month" = '10' OR
550 "#WK_SRC_TB_Sys"."month" = '11' OR
551 "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
552 ELSE 0
553 END
554 ), SUM(CASE
555 WHEN "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
556 ELSE 0
557 END
558 )) AS "PY_Dec_YTD"
559
560 FROM
561 "QWNjb3VudB8xODkzNjM5MjMw"."6f6e4085dc6942c5b9543cb85a5ad085" AS "#WK_SRC_TB_Sys"
562 LEFT JOIN
563 "QWNjb3VudB8xODkzNjM5MjMw"."e450aa655ad1401280f8c9f804834433" AS "#WK_DIM_Account"
564 ON "#WK_SRC_TB_Sys"."account_code" = "#WK_DIM_Account"."gl_account"
565
566 WHERE
567 "#WK_SRC_TB_Sys"._tags['Data_Year'] = CAST(CAST(:Data_Year AS INTEGER) - 1 AS VARCHAR) AND
568 "#WK_SRC_TB_Sys"._tags['Data_Month'] = :Data_Month
569
570 GROUP BY "#WK_SRC_TB_Sys"."year", "#WK_SRC_TB_Sys"."entity", "#WK_SRC_TB_Sys"."account_code", "#WK_SRC_TB_Sys"
571 ".description", "#WK_SRC_TB_Sys"."product_code", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
572 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
573 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs",
574 "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs", "#WK_DIM_Account"."fs"
575
576 ORDER BY "PY_account_code" ASC, "PY_FS" ASC ) PY
577
578 ON CONCAT(CY."entity",CY."account_code",CY."product_code")= CONCAT(PY."PY_entity",PY."PY_account_code",PY
579 ".PY_product_code")
```

For both Current & Prior Year's query, at the end of the query we would group them by the following:

- i. Year
- ii. Entity
- iii. Account_Code
- iv. Description
- v. Product_Code

The GROUP BY statements are indicated above.

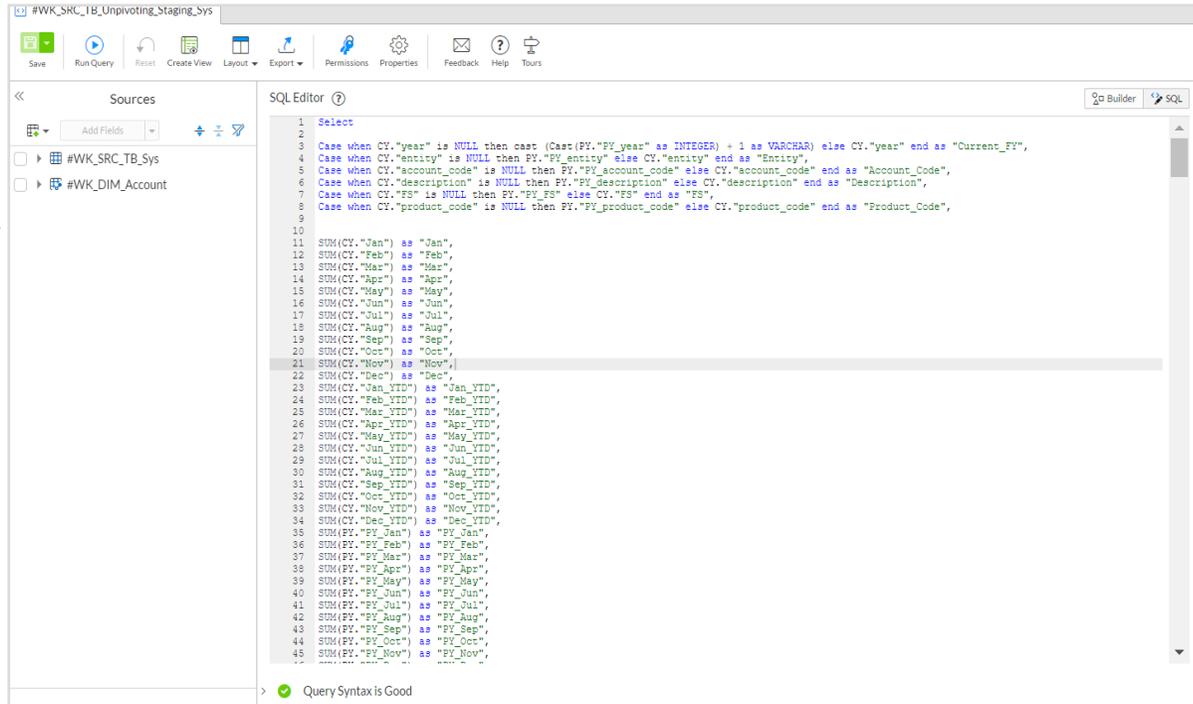
Step 16.1: Optimizing the Fields

```
SQL Editor ? Builder SQL
1 Select
2
3 Case when CY."year" is NULL then cast (Cast(PY."PY_year" as INTEGER) + 1 as VARCHAR) else CY."year" end as
  "Current_FY",
4 Case when CY."entity" is NULL then PY."PY_entity" else CY."entity" end as "Entity",
5 Case when CY."account_code" is NULL then PY."PY_account_code" else CY."account_code" end as "Account_Code",
6 Case when CY."description" is NULL then PY."PY_description" else CY."description" end as "Description",
7 Case when CY."FS" is NULL then PY."PY_FS" else CY."FS" end as "FS",
8 Case when CY."product_code" is NULL then PY."PY_product_code" else CY."product_code" end as "Product_Code",
9
```

As for rolling period, sometimes the current year's data would not be made available and this would result in 'blank / NULL output result fields. To cater for 'blank / NULL fields' of the results, we would include the following code snippet at the beginning.

This code snippet simply means, if any of the Current Year's columns is 'blank / NULL', we would append & display the Prior Year's data instead

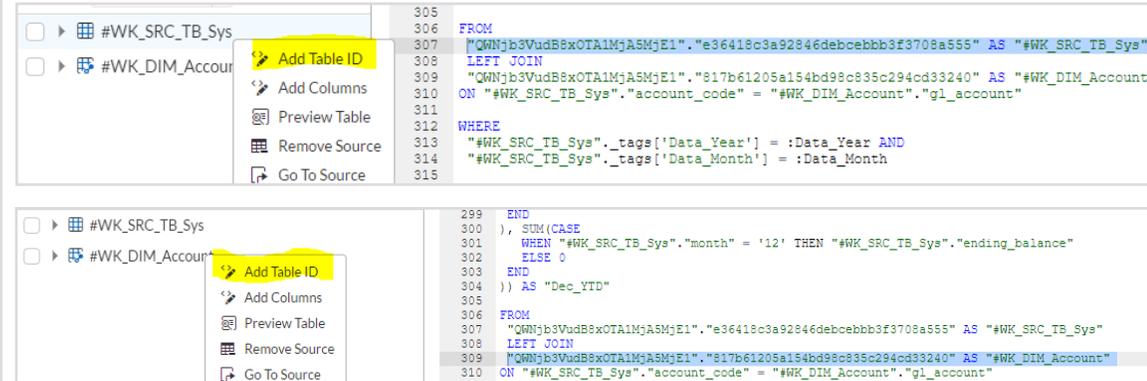
Step 16.2: Quick-Hack- #WK_SRC_TB_Unpivoting_Staging_Sys



The screenshot shows the SQL Editor interface with a query in the SQL Editor pane. The query is a complex SELECT statement with multiple CASE WHEN clauses and a series of SUM functions for each month of the year. A table ID is highlighted in the query, and a context menu is open over it, showing the 'Add Table ID' option selected.

```
1 Select
2
3 Case when CY."year" is NULL then cast (Cast(PY."PY_year" as INTEGER) + 1 as VARCHAR) else CY."year" end as "Current_FY",
4 Case when CY."entity" is NULL then PY."PY_entity" else CY."entity" end as "Entity",
5 Case when CY."account_code" is NULL then PY."PY_account_code" else CY."account_code" end as "Account_Code",
6 Case when CY."description" is NULL then PY."PY_description" else CY."description" end as "Description",
7 Case when CY."FS" is NULL then PY."PY_FS" else CY."FS" end as "FS",
8 Case when CY."product_code" is NULL then PY."PY_product_code" else CY."product_code" end as "Product_Code",
9
10
11 SUM(CY."Jan") as "Jan",
12 SUM(CY."Feb") as "Feb",
13 SUM(CY."Mar") as "Mar",
14 SUM(CY."Apr") as "Apr",
15 SUM(CY."May") as "May",
16 SUM(CY."Jun") as "Jun",
17 SUM(CY."Jul") as "Jul",
18 SUM(CY."Aug") as "Aug",
19 SUM(CY."Sep") as "Sep",
20 SUM(CY."Oct") as "Oct",
21 SUM(CY."Nov") as "Nov",
22 SUM(CY."Dec") as "Dec",
23 SUM(CY."Jan_YTD") as "Jan_YTD",
24 SUM(CY."Feb_YTD") as "Feb_YTD",
25 SUM(CY."Mar_YTD") as "Mar_YTD",
26 SUM(CY."Apr_YTD") as "Apr_YTD",
27 SUM(CY."May_YTD") as "May_YTD",
28 SUM(CY."Jun_YTD") as "Jun_YTD",
29 SUM(CY."Jul_YTD") as "Jul_YTD",
30 SUM(CY."Aug_YTD") as "Aug_YTD",
31 SUM(CY."Sep_YTD") as "Sep_YTD",
32 SUM(CY."Oct_YTD") as "Oct_YTD",
33 SUM(CY."Nov_YTD") as "Nov_YTD",
34 SUM(CY."Dec_YTD") as "Dec_YTD",
35 SUM(PY."PY_Jan") as "PY_Jan",
36 SUM(PY."PY_Feb") as "PY_Feb",
37 SUM(PY."PY_Mar") as "PY_Mar",
38 SUM(PY."PY_Apr") as "PY_Apr",
39 SUM(PY."PY_May") as "PY_May",
40 SUM(PY."PY_Jun") as "PY_Jun",
41 SUM(PY."PY_Jul") as "PY_Jul",
42 SUM(PY."PY_Aug") as "PY_Aug",
43 SUM(PY."PY_Sep") as "PY_Sep",
44 SUM(PY."PY_Oct") as "PY_Oct",
45 SUM(PY."PY_Nov") as "PY_Nov",
```

Replace Table IDs:



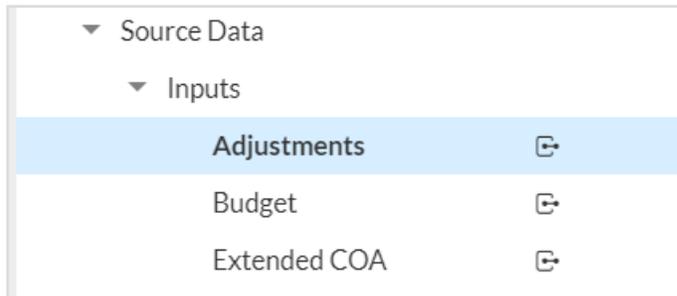
The first screenshot shows the context menu for the table ID '#WK_SRC_TB_Sys' in the query, with 'Add Table ID' selected. The second screenshot shows the context menu for the table ID '#WK_DIM_Account' in the query, with 'Add Table ID' selected.

```
305 FROM
306
307 ["QWNjB3VudB8xOTA1MjA5MjE1"."e36418c3a92846debocebbb3f3708a555" AS "#WK_SRC_TB_Sys"
308 LEFT JOIN
309 ["QWNjB3VudB8xOTA1MjA5MjE1"."817b61205a154bd98c835c294cd33240" AS "#WK_DIM_Account"
310 ON "#WK_SRC_TB_Sys"."account_code" = "#WK_DIM_Account"."gl_account"
311
312 WHERE
313 "#WK_SRC_TB_Sys"."_tags['Data_Year'] = :Data_Year AND
314 "#WK_SRC_TB_Sys"."_tags['Data_Month'] = :Data_Month
315
299 END
300 ), SUM(CASE
301 WHEN "#WK_SRC_TB_Sys"."month" = '12' THEN "#WK_SRC_TB_Sys"."ending_balance"
302 ELSE 0
303 END
304 ) AS "Dec_YTD"
305
306 FROM
307 ["QWNjB3VudB8xOTA1MjA5MjE1"."e36418c3a92846debocebbb3f3708a555" AS "#WK_SRC_TB_Sys"
308 LEFT JOIN
309 ["QWNjB3VudB8xOTA1MjA5MjE1"."817b61205a154bd98c835c294cd33240" AS "#WK_DIM_Account"
310 ON "#WK_SRC_TB_Sys"."account_code" = "#WK_DIM_Account"."gl_account
```

If your query doesn't work, copy paste the provided script into the 'SQL Editor', but remember to replace the 'Wdata Table ID at Line 307, 309, 566, 568. To replace the table ID, first highlight the entire line, right click on the '#WK_SRC_TB_Sys' table, and select "Add Table ID". Repeat for '#WK_DIM_Account'

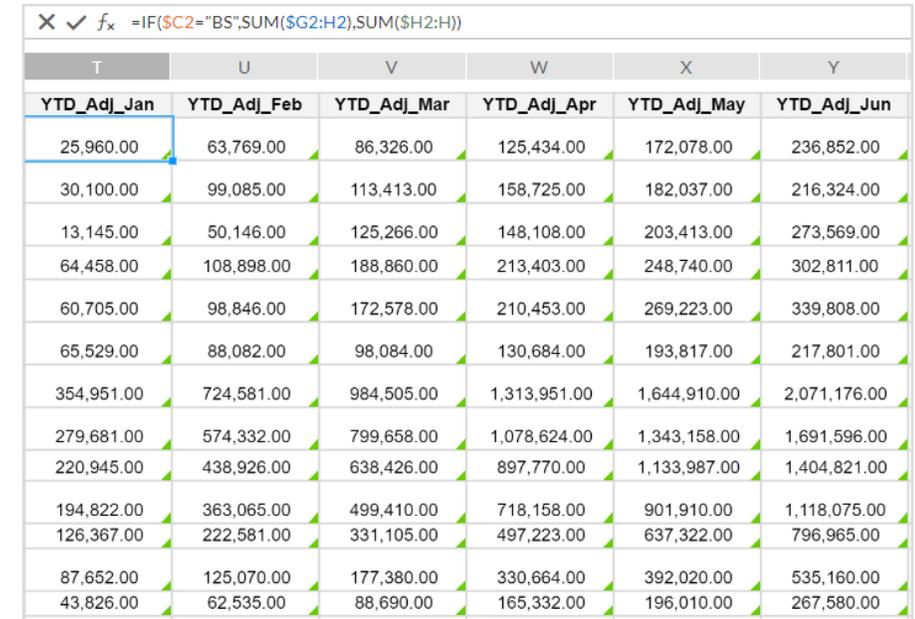
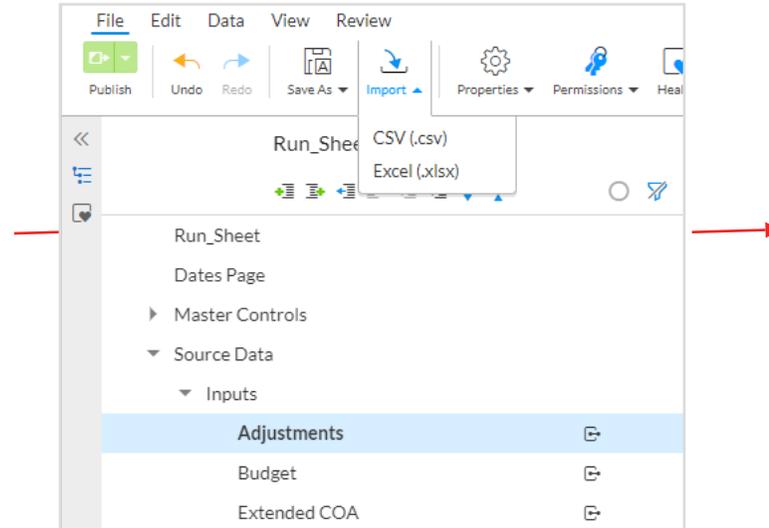
Step 17: Objective of the 'Adjustment' Sheet

↪ For users to populate late adjustments, these adjustment figures would be factored together with the TB amounts



Source Data

- Inputs
 - Adjustments
 - Budget
 - Extended COA



X ✓ f_x =IF(\$C2="BS",SUM(\$G2:H2),SUM(\$H2:H2))

T	U	V	W	X	Y
YTD_Adj_Jan	YTD_Adj_Feb	YTD_Adj_Mar	YTD_Adj_Apr	YTD_Adj_May	YTD_Adj_Jun
25,960.00	63,769.00	86,326.00	125,434.00	172,078.00	236,852.00
30,100.00	99,085.00	113,413.00	158,725.00	182,037.00	216,324.00
13,145.00	50,146.00	125,266.00	148,108.00	203,413.00	273,569.00
64,458.00	108,898.00	188,860.00	213,403.00	248,740.00	302,811.00
60,705.00	98,846.00	172,578.00	210,453.00	269,223.00	339,808.00
65,529.00	88,082.00	98,084.00	130,684.00	193,817.00	217,801.00
354,951.00	724,581.00	984,505.00	1,313,951.00	1,644,910.00	2,071,176.00
279,681.00	574,332.00	799,658.00	1,078,624.00	1,343,158.00	1,691,596.00
220,945.00	438,926.00	638,426.00	897,770.00	1,133,987.00	1,404,821.00
194,822.00	363,065.00	499,410.00	718,158.00	901,910.00	1,118,075.00
126,367.00	222,581.00	331,105.00	497,223.00	637,322.00	796,965.00
87,652.00	125,070.00	177,380.00	330,664.00	392,020.00	535,160.00
43,826.00	62,535.00	88,690.00	165,332.00	196,010.00	267,580.00

1: Create the following spreadsheet tabs

2: Import the Adjustment CSV data that is provided

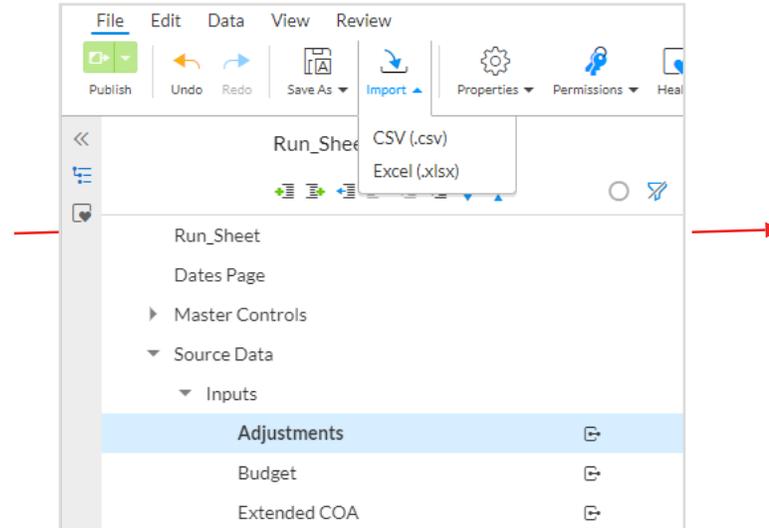
3: As the imported Adjustment spreadsheet only has up to 'Adj_Dec', we would need to create the 'YTD_Adj' calculations.

Sample Formula for 'YTD_Adj_Jan':
=IF(\$C2="BS",SUM(\$G2:H2),SUM(\$H2:H2))

Step 17.1: Objective of the 'Budget' Sheet

↪ For users to populate budget amounts, these budget figures would be combined in the TB table

Source Data
Inputs
Adjustments
Budget
Extended COA



	R	S	T	U
Dec	YTD_Bud_Jan	YTD_Bud_Feb	YTD_Bud_Mar	YTD_Bud_Apr
,195.20	8,671,304.80	9,814,500.63	20,813,507.99	20,914,225.29
(1.59)	(0.12)	11,677.58	4,149,275.60	4,137,597.90
,265.00)	1,222,644.89	4,122,312.40	4,122,312.40	4,461,254.78
—	0.17	0.05	11,677.75	4,149,275.77
,195.20	8,671,304.80	9,814,500.63	20,813,507.99	20,914,225.29
(1.59)	(0.12)	11,677.58	4,149,275.60	4,137,597.90
,265.00)	1,222,644.89	4,122,312.40	4,122,312.40	4,461,254.78
,195.20	8,671,304.80	9,814,500.63	20,813,507.99	20,914,225.29

1: Create the following spreadsheet tabs

2: Import the Budget CSV data that is provided

3: As the imported Budget spreadsheet only has up to 'Bud_Dec', we would need to create the 'YTD_Bud_Jan' calculations.

Sample Formula for 'YTD_Bud_Jan':
=SUM(\$F2:F2)

Step 17.2: Adjustment' Connected Sheet P1

The screenshot shows the 'Add Dataset' dialog box in Wdesk. The dialog has a search bar at the top and a list of datasets below. The 'Run_Sheet_SA Workspace' dataset is selected, and the 'Adjustments' sheet is highlighted under the 'Inputs' category. The background shows a table with columns for 'COLUMN NAME' and 'DESCRIPTION'.

COLUMN NAME	DESCRIPTION
Current_FY	Enter column de
Entity	Enter column de
Account_Code	Enter column de
Product_Code	Enter column de
Description	Enter column de
FS	Enter column de
jan	Enter column de
feb	Enter column de
mar	Enter column de
apr	Enter column de
may	Enter column de
jun	Enter column de
jul	Enter column de
aug	Enter column de
sep	Enter column de

In this step, we illustrate how to connect the 'Adjustment' spreadsheet on Wdesk to the Consumption table ("#WK_CNS_TB_Sys"). We click 'Add Dataset', and choose 'Spreadsheet', add the 'Adjustment' spreadsheet from Wdesk.

Step 17.3: Adjustment' Connected Sheet P2

Add Dataset

Select the sheet with the data to add to the table

Dataset name Source Run_Sheet_SA Workspace / Adjustments

Connect to sheet

EXISTING COLUMNS	COLUMNS FROM FILE	STATUS	FORMAT
current_fy	current_fy	✔ Mapped with same IDs	
entity	entity	✔ Mapped with same IDs	
account_code	account_code	✔ Mapped with same IDs	
product_code	-	❌ Not mapped	
description	description	✔ Mapped with same IDs	
fs	fs	✔ Mapped with same IDs	
1.00 .01 jan	-	❌ Not mapped	Period
1.00 .01 feb	-	❌ Not mapped	Period
1.00 .01 mar	-	❌ Not mapped	Period
1.00 .01 apr	-	❌ Not mapped	Period
1.00 .01 may	-	❌ Not mapped	Period
1.00 .01 jun	-	❌ Not mapped	Period
1.00 .01 jul	-	❌ Not mapped	Period
1.00 .01 aug	-	❌ Not mapped	Period

Add Dataset

1.00 .01 ytd_bud_jan	-	❌ Not mapped	Period
1.00 .01 ytd_bud_feb	-	❌ Not mapped	Period
1.00 .01 ytd_bud_mar	-	❌ Not mapped	Period
1.00 .01 ytd_bud_apr	-	❌ Not mapped	Period
1.00 .01 ytd_bud_may	-	❌ Not mapped	Period
1.00 .01 ytd_bud_jun	-	❌ Not mapped	Period
1.00 .01 ytd_bud_jul	-	❌ Not mapped	Period
1.00 .01 ytd_bud_aug	-	❌ Not mapped	Period
1.00 .01 ytd_bud_sep	-	❌ Not mapped	Period
1.00 .01 ytd_bud_oct	-	❌ Not mapped	Period
1.00 .01 ytd_bud_nov	-	❌ Not mapped	Period
1.00 .01 ytd_bud_dec	-	❌ Not mapped	Period

❌ 0 unmapped columns

[Hide Column Mappings](#)

Tag Name Tag Value

When 'Next' is clicked, tick the 'Connect to sheet' checkbox. Scroll down and add 'Data_Year' & 'Data_Month' as tags. Click 'Finish' to create the connection.

Step 17.4: Adjustment' Connected Sheet P3

The screenshot shows a data management interface for a table named '#WK_CNS_TB_Sys'. The table configuration is as follows:

COLUMN NAME	DESCRIPTION	COLUMN ID	COLUMN TYPE	IMPORT FORMAT
Current_FY	Enter column description	current_fy	Text	
Entity	Enter column description	entity	Text	
Account_Code	Enter column description	account_code	Text	
Product_Code	Enter column description	product_code	Text	
Description	Enter column description	description	Text	
FS	Enter column description	fs	Text	
jan	Enter column description	jan	Decimal	Period
feb	Enter column description	feb	Decimal	Period
mar	Enter column description	mar	Decimal	Period
apr	Enter column description	apr	Decimal	Period
may	Enter column description	may	Decimal	Period
jun	Enter column description	jun	Decimal	Period
jul	Enter column description	jul	Decimal	Period
aug	Enter column description	aug	Decimal	Period
sep	Enter column description	sep	Decimal	Period

The 'Datasets' panel on the right shows two connected datasets:

- Adjustments**: Last Update: Mar 26, 2021 11:49 AM by Xuan Zhi Choo. Tags: Data_Month: Dec-12, Data_Year: 2020.
- WK_SRC_TB_Unpivoting_Staging_Sys.csv**: Last Update: Mar 26, 2021 6:12 AM by Pankit Dhawan. Tags: Data_Month: Dec-12, Data_Year: 2020.

The 'Adjustment' spreadsheet is now connected to the Consumption table. Whenever the spreadsheet is refreshed, this table gets updated as well. Repeat the same steps to connect the **'Budget'** spreadsheet to this table as well. In the next step, we would be mixing all the tables that was created to form the Extended Trial Balance (ETB) Query.

Step 18: Creating the Budget Sheet

- 1: Repeat the steps for Adjustments but using Budget data.
- 2: Formulas for Budget will be different, the formula will be =SUM(\$F2:F2)

Step 19: Creating the Consumption Table P1 - WK_CNS_TB

The screenshot shows a data tool interface with a query editor and a results table. The query editor contains the following SQL code:

```
1 Select
2
3 Case when CY."year" is NULL then cast (Cast(PY."PY_year" as INTEGER) + 1 as VARCHAR) else CY."year"
4 end as "Current_Year",
5 Case when CY."entity" is NULL then PY."PY_entity" else CY."entity" end as "Entity",
6 Case when CY."account_code" is NULL then PY."PY_account_code" else CY."account_code" end as
7 "Account_Code",
8 Case when CY."description" is NULL then PY."PY_description" else CY."description" end as
9 "Description",
10 Case when CY."FS" is NULL then PY."PY_FS" else CY."FS" end as "FS",
11
12 SUM(CY."Jan") as "Jan",
13 SUM(CY."Feb") as "Feb",
14 SUM(CY."Mar") as "Mar",
15 SUM(CY."Apr") as "Apr",
16 SUM(CY."May") as "May",
17 SUM(CY."Jun") as "Jun",
18 SUM(CY."Jul") as "Jul",
```

The results table shows the following data:

	1	2	3	4	5	6	7	8	9
1	CURRENT_YEAR	ENTITY	ACCOUNT_CODE	DESCRIPTION	FS	JAN	FEB	MAR	APR
2	2020	c2452	61-18-00-00-114	Professional Fees - Tax Compliance	PL	2420	1936	2662	183
3	2020	c2452	61-11-00-00-124	Corporate Costs - Insurance	PL	538.38	430.7	592.22	409
4	2020	c2445	33-14-12-00-111	Int. on Loan from (>50%) PPL {1Y+}	BS	11700000	9360000	12900000	8892
5	2020	c2445	36-13-00-00-111	ICoPay (>50%) Current Account {1Y-}	BS	-155362	-124289.6	-170898.2	-118075
6	2020	c2452	61-18-00-00-115	Professional Fees - Legal	PL	345617.6	276494.08	380179.36	262669
7	2020	c2452	53-21-16-00-111	Financial Derivatives - Change in FMV (Gain)	PL	-64350.03	-51480.02	-70785.03	-48906
8	2020	c5854	67-17-26-00-111	HR Security Costs - ICORPI Soc. Sec. Cont. - Employer	PL	151330.43	361064.34	496463.17	343011

After the Staging query has been built & successfully ran, we export the results to 'CSV'.

Step 19.1: Building the 'WK_CNS_TB' Table Schema

From the Adjustments spreadsheet, copy the columns from 'Carry Forward', all the way to the end, and append to the query export csv file.

	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Carry-Forward	Adj_Jan	Adj_Feb	Adj_Mar	Adj_Apr	Adj_May	Adj_Jun	Adj_Jul	Adj_Aug	Adj_Sep	Adj_Oct	Adj_Nov	Adj_Dec	YTD_Adj_Jan
2		25,960.00	37,809.00	22,557.00	39,108.00	46,644.00	64,774.00	69,742.00	71,423.00	39,613.00	46,415.00	27,586.00	60,697.00	25,960.00
3		30,100.00	68,985.00	14,328.00	45,312.00	23,312.00	34,287.00	37,612.00	32,860.00	16,002.00	66,763.00	28,497.00	12,145.00	30,100.00
4		13,145.00	37,001.00	75,120.00	22,842.00	55,305.00	70,156.00	73,254.00	69,435.00	16,729.00	11,260.00	52,675.00	20,679.00	13,145.00
5		64,458.00	44,440.00	79,962.00	24,543.00	35,337.00	54,071.00	62,014.00	64,926.00	32,788.00	79,876.00	56,120.00	68,998.00	64,458.00
6		60,705.00	38,141.00	73,732.00	37,875.00	58,770.00	70,585.00	41,757.00	22,825.00	79,841.00	69,032.00	77,370.00	68,830.00	60,705.00
7		65,529.00	22,553.00	10,002.00	32,600.00	63,133.00	23,984.00	40,719.00	64,643.00	32,525.00	70,598.00	47,909.00	51,762.00	65,529.00
8		75,270.00	74,979.00	34,598.00	50,480.00	66,425.00	77,828.00	62,146.00	69,333.00	64,875.00	79,004.00	29,599.00	75,160.00	75,270.00
9		58,736.00	76,670.00	25,826.00	19,622.00	28,317.00	77,604.00	39,282.00	75,554.00	72,476.00	55,098.00	27,787.00	56,140.00	58,736.00

From the Budget spreadsheet, copy the columns from 'Bud_Jan', all the way to the end, and append to the query export csv file.

	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Bud_Jan	Bud_Feb	Bud_Mar	Bud_Apr	Bud_May	Bud_Jun	Bud_Jul	Bud_Aug	Bud_Sep	Bud_Oct	Bud_Nov	Bud_Dec	YTD_Bud_Jan	YTD_Bud_Feb
2	8,671,304.80	1,143,195.83	10,999,007.36	100,717.30	162,315.27	928,777.73	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	8,671,304.80	9,814,500.63
3	(0.12)	11,677.70	4,137,598.02	(11,677.70)	(3,742,858.86)	11,903.00	(78,993,022.55)	4,599,499.89	(2,087,265.08)	44,343,747.01	—	(1.59)	(0.12)	11,677.58
4	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	5,592,360.75	3,705,769.73	(2,249,745.00)	(600,566.00)	(3,342,615.75)	(315,265.00)	1,222,644.89	4,122,312.40
5	0.17	(0.12)	11,677.70	4,137,598.02	(11,677.70)	(3,742,858.86)	11,903.00	(78,993,022.55)	4,599,499.89	(2,087,265.08)	44,343,747.01	—	0.17	0.05
6	8,671,304.80	1,143,195.83	10,999,007.36	100,717.30	162,315.27	928,777.73	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	8,671,304.80	9,814,500.63
7	(0.12)	11,677.70	4,137,598.02	(11,677.70)	(3,742,858.86)	11,903.00	(78,993,022.55)	4,599,499.89	(2,087,265.08)	44,343,747.01	—	(1.59)	(0.12)	11,677.58
8	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	5,592,360.75	3,705,769.73	(2,249,745.00)	(600,566.00)	(3,342,615.75)	(315,265.00)	1,222,644.89	4,122,312.40
9	0.17	(0.12)	11,677.70	4,137,598.02	(11,677.70)	(3,742,858.86)	11,903.00	(78,993,022.55)	4,599,499.89	(2,087,265.08)	44,343,747.01	—	0.17	0.05
10	8,671,304.80	1,143,195.83	10,999,007.36	100,717.30	162,315.27	928,777.73	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	8,671,304.80	9,814,500.63
11	(0.12)	11,677.70	4,137,598.02	(11,677.70)	(3,742,858.86)	11,903.00	(78,993,022.55)	4,599,499.89	(2,087,265.08)	44,343,747.01	—	(1.59)	(0.12)	11,677.58
12	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	5,592,360.75	3,705,769.73	(2,249,745.00)	(600,566.00)	(3,342,615.75)	(315,265.00)	1,222,644.89	4,122,312.40
13	8,671,304.80	1,143,195.83	10,999,007.36	100,717.30	162,315.27	928,777.73	1,222,644.89	2,899,667.51	—	338,942.38	—	2,458,195.20	8,671,304.80	9,814,500.63

The consolidated schema in the CSV file:

AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ
Y_Oct_YTD	PY_Nov_YTD	PY_Dec_YTD	Carry-Forward	Adj_Jan	Adj_Feb	Adj_Mar	Adj_Apr	Adj_May	Adj_Jun	Adj_Jul	Adj_Aug	Adj_Sep	Adj_Oct	Adj_Nov	Adj_Dec	YTD_Adj_Jan	YTD_Adj_Feb
			25960	37809	22557	39108	46644	64774	69742	71423	39613	46415	27586	60697		25960	63769

The 'WK_CNS_TB' table schema, is a combination of Staging query export, Adjustment, and Budget columns. From the previous step, after the query was exported, copy the columns from the Adjustments and Budget Spreadsheet and append to the columns behind to build the schema for the 'WK_CNS_TB' wdata table.

Step 19.2: Creating the Consumption Table P2 - WK_CNS_TB

#WK_CNS_TB_Sys

Save Preview Add Column Permissions Rename Feedback Help Tours

Description: Enter Table Description

COLUMN NAME	DESCRIPTION	COLUMN ID	COLUMN TYPE	IMPORT FORMAT
Current_Year	Enter column description	current_year	Text	
Entity	Enter column description	entity	Text	
Account_Code	Enter column description	account_code	Text	
Product_Code	Enter column description	product_code	Text	
Description	Enter column description	description	Text	
FS	Enter column description	fs	Text	
Jan	Enter column description	jan	Decimal	
Feb	Enter column description	feb	Decimal	
Mar	Enter column description	mar	Decimal	
Apr	Enter column description	apr	Decimal	
May	Enter column description	may	Decimal	
Jun	Enter column description	jun	Decimal	
Jul	Enter column description	jul	Decimal	
Aug	Enter column description	aug	Decimal	
Sep	Enter column description	sep	Decimal	

Datasets

Add Dataset

WK_SRC_TB_Unpivoting_Staging_Sys.csv

Last Update: Mar 23, 2021 3:31 PM by Xuan Zhi Choo

Tags: Data_Month: Dec-12, Data_Year: 2020

> Table Preview

The updated file from previous step is used to create the consumption table "#WK_CNS_TB_Sys". The schema of consumption table would encapsulate the 'Pivoted Actual data', 'Adjustments' and 'Budget' data fields.

Step 20: Uploading Data into Consumption Table (Adv Query, Adj, Bud).

Uploaded the transformed TB data, filename:

1. Act_Dec-12_2020.csv
2. Adj_Dec-12_2020.csv
3. Bud_Dec-12_2020.csv

Step 21: Setting up Consumption Query

Steps	Description
0	Based on the previous steps, in general: <ul style="list-style-type: none">i. Actual Accounts can be used for Adjustmentsii. Actual Accounts can be used for Budgetiii. Specially created Budget Accounts should never be used for Adjustmentsiv. Specially created Adjustment Accounts should never be used for Budgetv. (THE ABOVE 2 have to be mutually exclusive) This relates to the extended COA.
1	We first import all the tables that was created previously, namely: <ul style="list-style-type: none">i. #WK_CNS_TB_Sysii. #WK_DIM_Cost Centeriii. #WK_DIM_Productiv. #WK_DIM_Account
2	Establish relationships between the tables: <ul style="list-style-type: none">i. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Account on account_codeii. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Cost Center on account_codeiii. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Product on product_code
3	Create calculation fields to display Quarter, YTD data which we would illustrate in the next few steps.

Step 21.1: Importing the Tables

The screenshot displays a software interface for adding table sources. A central dialog box titled "Add Primary Table Source" is open, featuring a search bar and a table of available sources. The table has columns for "NAME" and "COLUMNS". Under the "#Workiva" folder, the following tables are listed and checked:

NAME	COLUMNS
#WK_DIM_Cost Center	
#WK_DIM_Account	
#WK_DIM_Product	
#WK_CNS_TB_Sys	

Other folders visible include #Dimension, #Misc, #Source, #Staging, TB Data, and Archive. The background interface includes a "Sources" panel on the left with a tree view of the same folders, and a "Query Properties" panel on the right showing query details like "3/26/2021 10:56 AM", "Records Returned: 260", and "Time: 2.022 seconds".

Create a new query, import the following tables:

- i. #WK_DIM_Cost Center
- ii. #WK_DIM_Account
- iii. #WK_DIM_Product
- iv. #WK_CNS_TB_Sys

Step 21.2: Establish Table Relationships

The screenshot shows a data tool interface with a top navigation bar containing icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours. Below the navigation bar, there are two tabs: #WK_CNS_TB_Sys and #WK_CNS_ETB_User. The main interface is divided into two panels. The left panel, titled 'Sources', contains a list of data sources: Calculation, #WK_DIM_Cost Center, #WK_DIM_Product, #WK_CNS_TB_Sys, and #WK_DIM_Account. The right panel, titled 'Relationships', contains a table with columns for source, field, join type, and target. The table is as follows:

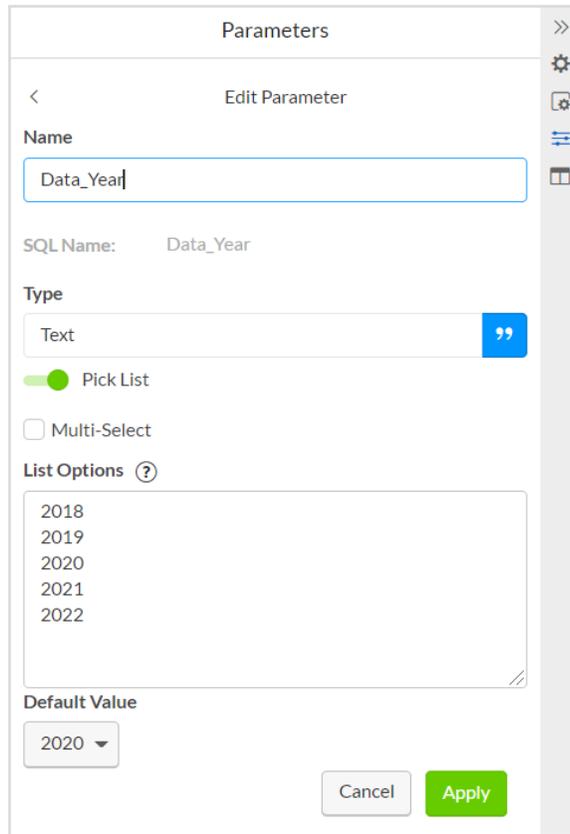
Source	Field	Join Type	Target
#WK_CNS_TB_Sys	account_code	LEFT JOIN	#WK_DIM_Cost C... accountcode
#WK_CNS_TB_Sys	account_code	LEFT JOIN	#WK_DIM_Account gl_account
#WK_CNS_TB_Sys	product_code	LEFT JOIN	#WK_DIM_Product productcode
Drop column to add to relationship		LEFT JOIN	Drop column to add to relationship

At the bottom of the interface, there is a green checkmark and the text 'Query Syntax is Good'.

Establish relationships between:

- i. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Cost Center on Account Code
- ii. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Account on GL Account
- iii. #WK_CNS_TB_Sys LEFT JOIN #WK_DIM_Product on Product Code

Step 21.3: Create Parameters



Parameters

Edit Parameter

Name
Data_Year

SQL Name: Data_Year

Type
Text

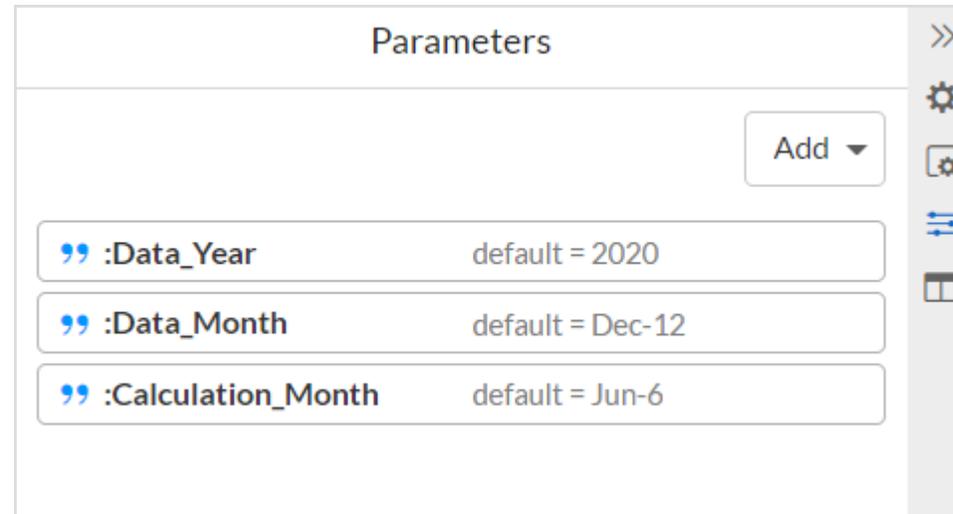
Pick List

Multi-Select

List Options ?
2018
2019
2020
2021
2022

Default Value
2020

Cancel Apply



Parameters

Add

☞ :Data_Year default = 2020

☞ :Data_Month default = Dec-12

☞ :Calculation_Month default = Jun-6

Create the following Parameters:

- i. Data_Year - Data for the year that was imported
- ii. Data_Month - Data for the month that was imported
- iii. Calculation_Month - Derive the YTD data (selected by user). (Note that the calculation month needs to be equal or earlier than Data_Month)

Step 21.4: Create Filters

The screenshot shows the Workiva query builder interface. At the top, there are two tabs: #WK_CNS_TB_Sys and #WK_CNS_ETB_User. Below the tabs is a toolbar with icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours. The main interface is divided into two main sections. On the left is the 'Sources' panel, which contains a list of data sources: Calculation, #WK_DIM_Cost Center, #WK_DIM_Product, #WK_CNS_TB_Sys, and #WK_DIM_Account. On the right is the 'Filters' tab, which shows a filter rule: {1} AND {2}. Below this, there are three filter rows. Row 1: #WK_CNS_TB_Sys TAGS [Data_Year] Equal To :Data_Year (default: '2020'). Row 2: #WK_CNS_TB_Sys TAGS [Data_Mon...] Equal To :Data_Month (default: 'Dec-12'). Row 3: Drop column to filter query results by Equal To. The interface also has tabs for Fields, Sort, and Relationships, and buttons for Builder and SQL.

Create the following filters:

- i. #WK_CNS_TB_Sys [Data_Year] = 'Data_Year' parameter
- ii. #WK_CNS_TB_Sys [Data_Month] = 'Data_Month' parameter

Step 22: Building Fields for the Query

The screenshot shows a query builder interface for a database query. The top navigation bar includes icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours. The main interface is divided into two main sections: 'Sources' on the left and 'Fields' on the right.

The 'Sources' section displays a list of data sources with checkboxes and a search bar. The list includes:

- ENTITY
- ACCOUNT_CODE
- PRODUCT_CODE
- DESCRIPTION
- FS
- JAN (1.00 .01)
- FEB (1.00 .01)
- MAR (1.00 .01)
- APR (1.00 .01)
- MAY (1.00 .01)
- JUN (1.00 .01)
- JUL (1.00 .01)
- AUG (1.00 .01)
- SEP (1.00 .01)
- OCT (1.00 .01)
- NOV (1.00 .01)
- DEC (1.00 .01)
- JAN_YTD (1.00 .01)

The 'Fields' section is currently active and shows a table with three columns: 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The table contains the following rows:

HEADER	SOURCE	SOURCE COLUMN
Current_FY	#WK_CNS_TB_Sys	current_fy
Entity	#WK_CNS_TB_Sys	entity
GL Account	#WK_DIM_Account	gl_account
GL Description	#WK_DIM_Account	gl_description
FS	#WK_DIM_Account	fs
FS1	#WK_DIM_Account	fs1
FS2	#WK_DIM_Account	fs2
FS3	#WK_DIM_Account	fs3
FS4	#WK_DIM_Account	fs4
FS5	#WK_DIM_Account	fs5
costcentercode	#WK_DIM_Cost Center	costcentercode

A red arrow points from the 'Sources' list to the 'Fields' table, indicating the process of dragging columns from the sources to the fields.

Drag the required columns to the 'Fields' space. Note: The required columns should mirror the columns in the 'ETB' spreadsheet in Wdesk.

Step 22.1: Current Year Quarter (Q) Fields

The screenshot displays a data tool interface with three main sections: Sources, Fields, and Field Properties.

- Sources:** A list of data sources is shown on the left. A 'Calculation' field is highlighted in yellow.
- Fields:** A table lists fields with their headers, sources, and source columns. The 'CY_Q1' field is highlighted in blue.
- Field Properties:** The 'Header' is set to 'CY_Q1'. The 'Calculation' field contains a SQL query. The 'Included Columns' section, highlighted in red, lists columns to be included in the query.

HEADER	SOURCE	SOURCE COLUMN
CY_Q1	#WK_CNS_TB_Sys	Multiple
CY_Q1_Adj	#WK_CNS_TB_Sys	Multiple
CY_Q1_TB	#WK_CNS_TB_Sys	Multiple
CY_Q2	#WK_CNS_TB_Sys	Multiple
CY_Q3	#WK_CNS_TB_Sys	Multiple
CY_Q4	#WK_CNS_TB_Sys	Multiple
PY_Q1	#WK_CNS_TB_Sys	100.01 py_mar_ytd
PY_Q2	#WK_CNS_TB_Sys	100.01 py_jun_ytd
PY_Q3	#WK_CNS_TB_Sys	100.01 py_sep_ytd
PY_Q4	#WK_CNS_TB_Sys	100.01 py_dec_ytd
CY_YTD	#WK_CNS_TB_Sys	Multiple

```
1 SUM(CASE
2   WHEN :Calculation_Month = 'Jan-1' THEN
3     COALESCE({1}, 0) + COALESCE({2}, 0)
4   WHEN :Calculation_Month = 'Feb-2' THEN
5     COALESCE({3}, 0) + COALESCE({4}, 0)
6   ELSE COALESCE({5}, 0) + COALESCE({6}, 0)
7 )
```

Included Columns:

- 100.01 jan_ytd / #WK_CNS_TB_Sys
- 100.01 ytd_adj_jan / #WK_CNS_TB_Sys
- 100.01 feb_ytd / #WK_CNS_TB_Sys
- 100.01 ytd_adj_feb / #WK_CNS_TB_Sys

Drop columns to include here

Apply

Query Syntax is Good

To calculate the Current Year Quarter, we would be summing the YTD numbers & Adjustments. First pull a 'Calculation' into the 'Fields' space, then include "ytd" & "ytd_adj" columns into the space highlighted in red, along with the SUM CASE formula. As this is for Q1, we would only be including up to Mar' numbers.

Step 22.2: Current Year Q Adjustment

The screenshot displays a query builder interface with the following components:

- Sources:** A list of data sources on the left, including 'Calculation', '#WK_DIM_Cost Center', '#WK_DIM_Product', '#WK_CNS_TB_Sys', and various fields like 'CURRENT_FY', 'ENTITY', 'ACCOUNT_CODE', 'PRODUCT_CODE', 'DESCRIPTION', 'FS', and months from 'JAN' to 'AUG'.
- Fields:** A central workspace with tabs for 'Fields', 'Filters', 'Sort', and 'Relationships'. It contains a table with columns: 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The table lists various quarterly and year-to-date (YTD) calculations, such as 'CY_Q1', 'CY_Q1_Adj', 'CY_Q1_TB', 'CY_Q2', 'CY_Q3', 'CY_Q4', 'PY_Q1', 'PY_Q2', 'PY_Q3', 'PY_Q4', and 'CY_YTD'. The 'CY_Q1_Adj' row is highlighted.
- Field Properties:** A panel on the right showing the 'Header' as 'CY_Q1_Adj' and a 'Calculation' formula:

```
1 SUM(CASE
2   WHEN :Calculation_Month = 'Jan-1' THEN {1}
3   WHEN :Calculation_Month = 'Feb-2' THEN {2}
4   ELSE {3}
5 END
6 )
```
- Included Columns:** A list of columns to be included in the query results, with the first three items highlighted in red:
 1. $\frac{100}{.01}$ ytd_adj_jan / #WK_CNS_TB_Sys
 2. $\frac{100}{.01}$ ytd_adj_feb / #WK_CNS_TB_Sys
 3. $\frac{100}{.01}$ ytd_adj_mar / #WK_CNS_TB_Sys
 4. Drop columns to include here
- Buttons:** 'Builder' and 'SQL' buttons are at the top right. An 'Apply' button is at the bottom right of the 'Included Columns' section.
- Status:** A green checkmark and the text 'Query Syntax is Good' are at the bottom left.

To display adjustments figures by quarter, pull a 'Calculation' into the 'Fields' space, then include 'ytd_adj' columns along with the SUM CASE formula. As this is for Q1, we would only be including up to Mar' numbers.

Step 22.3: Current Year Q TB

The screenshot shows a query builder interface with a top toolbar containing icons for Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, and Tours. The main workspace is divided into three panels: Sources, Fields, and Field Properties.

Sources Panel: Lists available data sources including #WK_DIM_Cost Center, #WK_DIM_Product, #WK_DIM_Account, and #WK_CNS_TB_Sys.

Fields Panel: A table with columns for HEADER, SOURCE, and SOURCE COLUMN. The 'CY_Q1_TB' field is selected, showing it is derived from the '#WK_CNS_TB_Sys' source and is a 'Multiple' type calculation.

HEADER	SOURCE	SOURCE COLUMN
costcentername	#WK_DIM_Cost Center	costcentername
fmproducttype	#WK_DIM_Product	fmproducttype
fmproductgeography	#WK_DIM_Product	fmproductgeography
CY_Q1	#WK_CNS_TB_Sys	Multiple
CY_Q1_Adj	#WK_CNS_TB_Sys	Multiple
CY_Q1_TB	#WK_CNS_TB_Sys	Multiple
CY_Q2	#WK_CNS_TB_Sys	Multiple
CY_Q3	#WK_CNS_TB_Sys	Multiple
CY_Q4	#WK_CNS_TB_Sys	Multiple
PY_Q1	#WK_CNS_TB_Sys	py_mar_ytd
PY_Q2	#WK_CNS_TB_Sys	py_iun_vtd

Field Properties Panel: Shows the 'Header' as 'CY_Q1_TB' and the 'Calculation' as a SUM CASE formula:

```
1 SUM(CASE
2   WHEN :Calculation_Month = 'Jan-1' THEN {1}
3   WHEN :Calculation_Month = 'Feb-2' THEN {2}
4   ELSE {3}
5 END
6 )
```

Included Columns: A list of columns to be included in the query results, highlighted with a red box:

1. 100.01 jan_ytd / #WK_CNS_TB_Sys
2. 100.01 feb_ytd / #WK_CNS_TB_Sys
3. 100.01 mar_ytd / #WK_CNS_TB_Sys
4. Drop columns to include here

An 'Apply' button is located below the included columns list.

At the bottom of the interface, a green checkmark indicates 'Query Syntax is Good'.

To display TB figures by quarter, pull a 'Calculation' into the 'Fields' space, then include '_ytd' columns along with the SUM CASE formula. As this is for Q1, we would only be including up to Mar' numbers.

Step 23: Current Year YTD

The screenshot displays a data tool interface for configuring a query. The main workspace is divided into three sections: Sources, Fields, and Field Properties.

- Sources:** A list of data sources is shown on the left, including '#WK_DIM_Cost Center', '#WK_DIM_Product', '#WK_CNS_TB_Sys', and various fiscal year and month sources like 'CURRENT_FY', 'ENTITY', 'ACCOUNT_CODE', 'PRODUCT_CODE', 'DESCRIPTION', 'FS', and months from 'JAN' to 'NOV'.
- Fields:** A table lists fields with their headers, sources, and source columns. The 'CY_YTD' field is highlighted. The table is as follows:

HEADER	SOURCE	SOURCE COLUMN
CY_Q4	#WK_CNS_TB_Sys	Multiple
CY_Q4_Adj	#WK_CNS_TB_Sys	Multiple
CY_Q4_TB	#WK_CNS_TB_Sys	Multiple
CY_YTD	#WK_CNS_TB_Sys	Multiple
CY_FTM	#WK_CNS_TB_Sys	Multiple
CY_MoM	#WK_CNS_TB_Sys	Multiple
CY_QoQ	#WK_CNS_TB_Sys	Multiple
PY_Q1	#WK_CNS_TB_Sys	py_mar_ytd
PY_Q2	#WK_CNS_TB_Sys	py_jun_ytd
PY_Q3	#WK_CNS_TB_Sys	py_sep_ytd
PY_Q4	#WK_CNS_TB_Sys	py_dec_ytd
PY_YTD	#WK_CNS_TB_Sys	py_dec_ytd
PY_FTM	#WK_CNS_TB_Sys	Multiple
- Field Properties:** The 'Header' is 'CY_YTD'. The 'Calculation' is a SUM CASE formula:

```
SUM(CASE
  1 WHEN :Calculation_Month = 'Jan-1' THEN COALESCE({1}, 0) + COALESCE({13}, 0)
  2 WHEN :Calculation_Month = 'Feb-2' THEN COALESCE({2}, 0) + COALESCE({14}, 0)
  3 WHEN :Calculation_Month = 'Mar-3' THEN COALESCE({3}, 0) + COALESCE({15}, 0)
  4 WHEN :Calculation_Month = 'Apr-4' THEN COALESCE({4}, 0) + COALESCE({16}, 0)
  5 WHEN :Calculation_Month = 'May-5' THEN COALESCE({5}, 0) + COALESCE({17}, 0)
  6 WHEN :Calculation_Month = 'Jun-6' THEN COALESCE({6}, 0) + COALESCE({18}, 0)
  7 WHEN :Calculation_Month = 'Jul-7' THEN COALESCE({7}, 0) + COALESCE({19}, 0)
  8 WHEN :Calculation_Month = 'Aug-8' THEN COALESCE({8}, 0) + COALESCE({20}, 0)
  9 WHEN :Calculation_Month = 'Sep-9' THEN COALESCE({9}, 0) + COALESCE({21}, 0)
  10 WHEN :Calculation_Month = 'Oct-10' THEN COALESCE({10}, 0) + COALESCE({22}, 0)
  11 WHEN :Calculation_Month = 'Nov-11' THEN COALESCE({11}, 0) + COALESCE({23}, 0)
  12 ELSE COALESCE({12}, 0) + COALESCE({24}, 0)
  13 END )
```

The 'Included Columns' list contains: 'sep_ytd / #WK_CNS_TB_Sys', 'oct_ytd / #WK_CNS_TB_Sys', 'nov_ytd / #WK_CNS_TB_Sys', 'dec_ytd / #WK_CNS_TB_Sys', and 'adj_ian / #WK_CNS_TB_Sys'. An 'Apply' button is at the bottom.

A status message at the bottom left indicates 'Query Syntax is Good'.

To display Current Year's YTD, pull a 'Calculation' into the 'Fields' space, then include '_ytd' columns, along with the YTD adjustments columns along with the SUM CASE formula. This would serve as an aggregated figures for the whole year.

Step 23.1: Current Year FTM (w/o adjustments)

The screenshot shows a query builder interface with the following components:

- Sources Panel:** A tree view on the left showing data sources. The selected source is `#WK_DIM_Account`, with a sub-selection for `100 MAY`.
- Fields Panel:** A table with columns: HEADER, SOURCE, and SOURCE COLUMN. It lists various fields like `Current_FY`, `Entity`, `GL Account`, etc., mapped to their respective source tables.
- Field Properties Panel:** Shows the header `CY_FTM (w/o Adj)`.
- Calculation Panel:** Contains a SQL query using a SUM CASE formula to calculate the current year's FTM for each month from Jan to Nov.
- Included Columns Panel:** Lists columns for each month: `100 jan / #WK_DIM_Account`, `100 feb / #WK_DIM_Account`, ..., `100 may / #WK_DIM_Account`.

```
1 SUM(CASE
2
3   WHEN :Calculation_Month = 'Jan-1' THEN {1}
4   WHEN :Calculation_Month = 'Feb-2' THEN {2}
5   WHEN :Calculation_Month = 'Mar-3' THEN {3}
6   WHEN :Calculation_Month = 'Apr-4' THEN {4}
7   WHEN :Calculation_Month = 'May-5' THEN {5}
8   WHEN :Calculation_Month = 'Jun-6' THEN {6}
9   WHEN :Calculation_Month = 'Jul-7' THEN {7}
10  WHEN :Calculation_Month = 'Aug-8' THEN {8}
11  WHEN :Calculation_Month = 'Sep-9' THEN {9}
12  WHEN :Calculation_Month = 'Oct-10' THEN {10}
13  WHEN :Calculation_Month = 'Nov-11' THEN {11}
14
15  ELSE {12} END)
```

To display Current Year's For the Month (FTM), pull a 'Calculation' into the 'Fields' space, then include 'Jan-Dec', along with the SUM CASE formula. Note that this query did not account for adjustments. Next slide illustrates an example which accounts for adjustments.

Step 23.2: Current Year FTM (with Adjustments)

The screenshot shows a data tool interface with the following components:

- Sources:** A list of data sources on the left, including ADJ_JUL, ADJ_AUG, ADJ_SEP, ADJ_OCT, ADJ_NOV, ADJ_DEC, YTD_ADJ_JAN, YTD_ADJ_FEB, YTD_ADJ_MAR, YTD_ADJ_APR, YTD_ADJ_MAY, YTD_ADJ_JUN, YTD_ADJ_JUL, YTD_ADJ_AUG, YTD_ADJ_SEP, YTD_ADJ_OCT, YTD_ADJ_NOV, YTD_ADJ_DEC, COMMENTS, BUD_JAN, and BUD_FEB.
- Fields:** A central table listing fields with their source and calculation type. The 'CY_FTM' field is highlighted.
- Field Properties:** A panel on the right showing the configuration for the 'CY_FTM' field, including a calculation formula and a list of included columns.

HEADER	SOURCE	SOURCE COLUMN
CY_Q4	#WK_CNS_TB_Sys	Multiple fx
CY_Q4_Adj	#WK_CNS_TB_Sys	Multiple fx
CY_Q4_TB	#WK_CNS_TB_Sys	Multiple fx
CY_YTD	#WK_CNS_TB_Sys	Multiple fx
CY_FTM	#WK_CNS_TB_Sys	Multiple fx
CY_MoM	#WK_CNS_TB_Sys	Multiple fx
CY_QoQ	#WK_CNS_TB_Sys	Multiple fx
PY_Q1	#WK_CNS_TB_Sys	100 py_mar_ytd Σ
PY_Q2	#WK_CNS_TB_Sys	100 py_jun_ytd Σ
PY_Q3	#WK_CNS_TB_Sys	100 py_sep_ytd Σ
PY_Q4	#WK_CNS_TB_Sys	100 py_dec_ytd Σ
PY_YTD	#WK_CNS_TB_Sys	100 py_dec_ytd Σ
PY_FTM	#WK_CNS_TB_Sys	Multiple fx

```
Header *
CY_FTM

Calculation Ⓞ
1 SUM(CASE
2 WHEN :Calculation_Month = 'Jan-1' THEN COALESCE({1}, 0) + COALESCE({13}, 0)
3 WHEN :Calculation_Month = 'Feb-2' THEN COALESCE({2}, 0) + COALESCE({14}, 0)
4 WHEN :Calculation_Month = 'Mar-3' THEN COALESCE({3}, 0) + COALESCE({15}, 0)
5 WHEN :Calculation_Month = 'Apr-4' THEN COALESCE({4}, 0) + COALESCE({16}, 0)
6 WHEN :Calculation_Month = 'May-5' THEN COALESCE({5}, 0) + COALESCE({17}, 0)
7 WHEN :Calculation_Month = 'Jun-6' THEN COALESCE({6}, 0) + COALESCE({18}, 0)
8 WHEN :Calculation_Month = 'Jul-7' THEN COALESCE({7}, 0) + COALESCE({19}, 0)
9 WHEN :Calculation_Month = 'Aug-8' THEN COALESCE({8}, 0) + COALESCE({20}, 0)
10 WHEN :Calculation_Month = 'Sep-9' THEN COALESCE({9}, 0) + COALESCE({21}, 0)
11 WHEN :Calculation_Month = 'Oct-10' THEN COALESCE({10}, 0) + COALESCE({22}, 0)
12 WHEN :Calculation_Month = 'Nov-11' THEN COALESCE({11}, 0) + COALESCE({23}, 0)
13 ELSE COALESCE({12}, 0) + COALESCE({24}, 0)
14 END )

Included Columns
1. 100 jan / #WK_CNS_TB_Sys
2. 100 feb / #WK_CNS_TB_Sys
3. 100 mar / #WK_CNS_TB_Sys
4. 100 apr / #WK_CNS_TB_Sys
5. 100 may / #WK_CNS_TB_Sys
25. Drop columns to include here
```

Apply

Query Syntax is Good

To display Current Year's For the Month (FTM), pull a 'Calculation' into the 'Fields' space, then include 'Jan-Dec', along with the adjustment columns along with the SUM CASE formula.

Step 23.3: Current Year MOM

The screenshot shows a data tool interface with the following components:

- Sources:** A list of data sources including ENTITY, ACCOUNT_CODE, PRODUCT_CODE, DESCRIPTION, FS, and months from JAN to DEC, along with JAN_YTD.
- Fields:** A table with columns HEADER, SOURCE, and SOURCE COLUMN. The 'CY_MoM' field is selected, showing it is derived from the '#WK_CNS_TB_Sys' source.
- Field Properties:** Shows the 'Header' as 'CY_MoM' and the 'Calculation' as a SUM CASE formula. The formula calculates the difference between the current month and the previous month, using COALESCE to handle null values.
- Included Columns:** A list of columns to be included in the query, including '100.01 jan / #WK_CNS_TB_Sys', '100.01 py_dec / #WK_CNS_TB_Sys', and '100.01 feb / #WK_CNS_TB_Sys'.
- Footer:** A green 'Apply' button and a status message 'Query Syntax is Good'.

```
1 SUM(CASE
2   WHEN :Calculation Month = 'Jan-1' THEN COALESCE
3     ({1}, 0) - COALESCE({2}, 0)
4   WHEN :Calculation Month = 'Feb-2' THEN COALESCE
5     ({3}, 0) - COALESCE({1}, 0)
6   WHEN :Calculation Month = 'Mar-3' THEN COALESCE
7     ({4}, 0) - COALESCE({3}, 0)
8   WHEN :Calculation Month = 'Apr-4' THEN COALESCE
9     ({5}, 0) - COALESCE({4}, 0)
10  WHEN :Calculation Month = 'May-5' THEN COALESCE
11    ({6}, 0) - COALESCE({5}, 0)
12  WHEN :Calculation Month = 'Jun-6' THEN COALESCE
13    ({7}, 0) - COALESCE({6}, 0)
14  WHEN :Calculation Month = 'Jul-7' THEN COALESCE
15    ({8}, 0) - COALESCE({7}, 0)
16  WHEN :Calculation Month = 'Aug-8' THEN COALESCE
17    ({9}, 0) - COALESCE({8}, 0)
18  WHEN :Calculation Month = 'Sep-9' THEN COALESCE
19    ({10}, 0) - COALESCE({9}, 0)
20  WHEN :Calculation Month = 'Oct-10' THEN COALESCE
21    ({11}, 0) - COALESCE({10}, 0)
22  WHEN :Calculation Month = 'Nov-11' THEN COALESCE
23    ({12}, 0) - COALESCE({11}, 0)
24  WHEN :Calculation Month = 'Dec-12' THEN COALESCE
25    ({13}, 0) - COALESCE({12}, 0)
26  ELSE 0
27 END)
```

To display Current Year's Month on Month (MOM), pull a 'Calculation' into the 'Fields' space, then with the SUM CASE formula, derive the difference in figures between months encapsulated with 'COALESCE'

Step 23.4: Current Year QOQ

The screenshot displays a data tool interface for configuring a query. The main area is divided into three sections:

- Sources:** A list of data sources on the left, including 'ENTITY', 'ACCOUNT_CODE', 'PRODUCT_CODE', 'DESCRIPTION', 'FS', and months from 'JAN' to 'DEC', along with 'JAN_YTD'.
- Fields:** A table with columns 'HEADER', 'SOURCE', and 'SOURCE COLUMN'. The 'CY_QoQ' field is highlighted. Other fields include 'CY_FTM', 'CY_MoM', 'PY_Q1' through 'PY_Q4', 'PY_YTD', 'Bud_CY_Q1', and 'Bud_CY_Q2'.
- Field Properties:** A panel for the selected 'CY_QoQ' field. It shows the 'Header' as 'CY_QoQ' and the 'Calculation' as a SQL formula:

```
1 SUM(CASE
2   WHEN :Calculation_Month = 'Mar-3' THEN COALESCE({1}, 0)
3   ) - COALESCE({2}, 0)
4   WHEN :Calculation_Month = 'Jun-6' THEN COALESCE({3}, 0)
5   ) - COALESCE({1}, 0)
6   WHEN :Calculation_Month = 'Sep-9' THEN COALESCE({4}, 0)
7   ) - COALESCE({3}, 0)
8   WHEN :Calculation_Month = 'Dec-12' THEN COALESCE({5},
9   0) - COALESCE({4}, 0)
10  ELSE 0
11  END
12 )
```

Below the formula, 'Included Columns' are listed: 'mar_ytd / #WK_CNS_TB_Sys', 'py_dec_ytd / #WK_CNS_TB_Sys', 'jun_ytd / #WK_CNS_TB_Sys', and 'sep_ytd / #WK_CNS_TB_Sys'. A green 'Apply' button is at the bottom.

A status message at the bottom left of the main area reads: "Query Syntax is Good".

To display Current Year's Quarter on Quarter (QOQ), pull a 'Calculation' into the 'Fields' space, then with the SUM CASE formula, derive the difference in figures between quarters encapsulated with 'COALESCE'. Replicate for Budget.

Step 23.5: Past Year's Quarter (Q) Fields

The screenshot shows a data tool interface with the following components:

- Top Bar:** #WK_CNS_ETB_User, Save, Run Query, Reset, Create View, Layout, Export, Permissions, Properties, Feedback, Help, Tours.
- Sources Panel:** A list of fields with checkboxes. 'PY_MAR_YTD' is selected.
- Fields Panel:** A table with columns: HEADER, SOURCE, SOURCE COLUMN, and an aggregation icon.
- Field Properties Panel:** Configuration for the selected field.

HEADER	SOURCE	SOURCE COLUMN	Aggregation
CY_Q4_TB	#WK_CNS_TB_Sys	Multiple	f _x
PY_Q1	#WK_CNS_TB_Sys	1.00 .01 py_mar_ytd	Σ
PY_Q2	#WK_CNS_TB_Sys	1.00 .01 py_jun_ytd	Σ
PY_Q3	#WK_CNS_TB_Sys	1.00 .01 py_sep_ytd	Σ
PY_Q4	#WK_CNS_TB_Sys	1.00 .01 py_dec_ytd	Σ
CY_YTD	#WK_CNS_TB_Sys	Multiple	f _x
PY_YTD	#WK_CNS_TB_Sys	1.00 .01 py_dec_ytd	Σ
CY_MoM	#WK_CNS_TB_Sys	Multiple	f _x
CY_QoQ	#WK_CNS_TB_Sys	Multiple	f _x
CY_FTM	#WK_CNS_TB_Sys	Multiple	f _x
PY_FTM	#WK_CNS_TB_Sys	Multiple	f _x

Field Properties:

- Source: #WK_CNS_TB_Sys
- Column: 1.00
.01 py_mar_ytd
- Header: PY_Q1
- Type: Decimal (1.00
.01)
- Aggregation: Sum of
- Use distinct aggregation

Apply

Query Syntax is Good

As Past Year's figures have already occurred, the Quarter (Q) Fields are simply a direct representation of the of the quarter's YTD, i.e. ('PY_MAR_YTD', 'PY_JUN_YTD', 'PY_SEP_YTD', 'PY_DEC_YTD')

Step 24: Current Year YTD

#WK_CNS_ETB_User

Save Run Query Reset Create View Layout Export Permissions Properties Feedback Help Tours

Sources

Add to Fields

100.01 PY_DEC_YTD

100.01 CARRY FORWARD

100.01 ADJ_JAN

100.01 ADJ_FEB

100.01 ADJ_MAR

100.01 ADJ_APR

100.01 ADJ_MAY

100.01 ADJ_JUN

100.01 ADJ_JUL

100.01 ADJ_AUG

100.01 ADJ_SEP

100.01 ADJ_OCT

100.01 ADJ_NOV

100.01 ADJ_DEC

100.01 YTD_ADJ_JAN

100.01 YTD_ADJ_FEB

100.01 YTD_ADJ_MAR

100.01 YTD_ADJ_APR

Fields Filters Sort Relationships Builder SQL

HEADER	SOURCE	SOURCE COLUMN
CY_YTD	#WK_CNS_TB_Sys	Multiple f_x
PY_Q1	#WK_CNS_TB_Sys	100.01 py_mar_ytd Σ
PY_Q2	#WK_CNS_TB_Sys	100.01 py_jun_ytd Σ
PY_Q3	#WK_CNS_TB_Sys	100.01 py_sep_ytd Σ
PY_Q4	#WK_CNS_TB_Sys	100.01 py_dec_ytd Σ
PY_YTD	#WK_CNS_TB_Sys	100.01 py_dec_ytd Σ
CY_MoM	#WK_CNS_TB_Sys	Multiple f_x
CY_QoQ	#WK_CNS_TB_Sys	Multiple f_x
CY_FTM	#WK_CNS_TB_Sys	Multiple f_x
PY_FTM	#WK_CNS_TB_Sys	Multiple f_x
Bud. CY O1	#WK_CNS_TB_Sys	Multiple f_x

Drop column to include in query results

Field Properties

Source #WK_CNS_TB_Sys

Column 100.01 py_dec_ytd

Header PY_YTD

Type Decimal 100.01

Aggregation Sum of

Use distinct aggregation

Apply

Query Syntax is Good

To display Past Year's YTD, 'PY_DEC_YTD' would be an accurate representation for the year's YTD figures.

Step 24.1: Past Year's FTM

The screenshot displays a data tool interface for configuring a query. The main workspace is divided into three panels:

- Sources:** A list of data sources on the left. 'FEB' is selected under the '100 .01' category.
- Fields:** A table of available fields. The 'PY_FTM' field is highlighted, showing it is a calculation from the '#WK_CNS_TB_Sys' source.
- Field Properties:** Configuration for the selected 'PY_FTM' field.
 - Header:** PY_FTM
 - Calculation:** A SUM CASE formula:

```
1 SUM(CASE
2 WHEN :Calculation_Month = 'Jan-1' THEN {1}
3 WHEN :Calculation_Month = 'Feb-2' THEN {2}
4 WHEN :Calculation_Month = 'Mar-3' THEN {3}
5 WHEN :Calculation_Month = 'Apr-4' THEN {4}
6 WHEN :Calculation_Month = 'May-5' THEN {5}
7 WHEN :Calculation_Month = 'Jun-6' THEN {6}
8 WHEN :Calculation_Month = 'Jul-7' THEN {7}
9 WHEN :Calculation_Month = 'Aug-8' THEN {8}
10 WHEN :Calculation_Month = 'Sep-9' THEN {9}
11 WHEN :Calculation_Month = 'Oct-10' THEN {10}
12 WHEN :Calculation_Month = 'Nov-11' THEN {11}
13 WHEN :Calculation_Month = 'Dec-12' THEN {12}
14 ELSE 0
15 END
```
 - Included Columns:** A list of columns to include in the query. The first four items are '100 .01 py_jan / #WK_CNS_TB_Sys', '100 .01 py_feb / #WK_CNS_TB_Sys', '100 .01 py_mar / #WK_CNS_TB_Sys', and '100 .01 py_apr / #WK_CNS_TB_Sys'. A 'Drop columns to include here' box is at the bottom.

A green 'Apply' button is located at the bottom right of the Field Properties panel. A status message at the bottom left of the Fields panel reads 'Query Syntax is Good'.

To display Past Year's FTM, pull a 'Calculation' into the 'Fields' space, then include 'PY_Jan-PY_Dec' columns along with the SUM CASE formula.

Step 24.2: Quick-Hack - #WK_CNS_TB_User

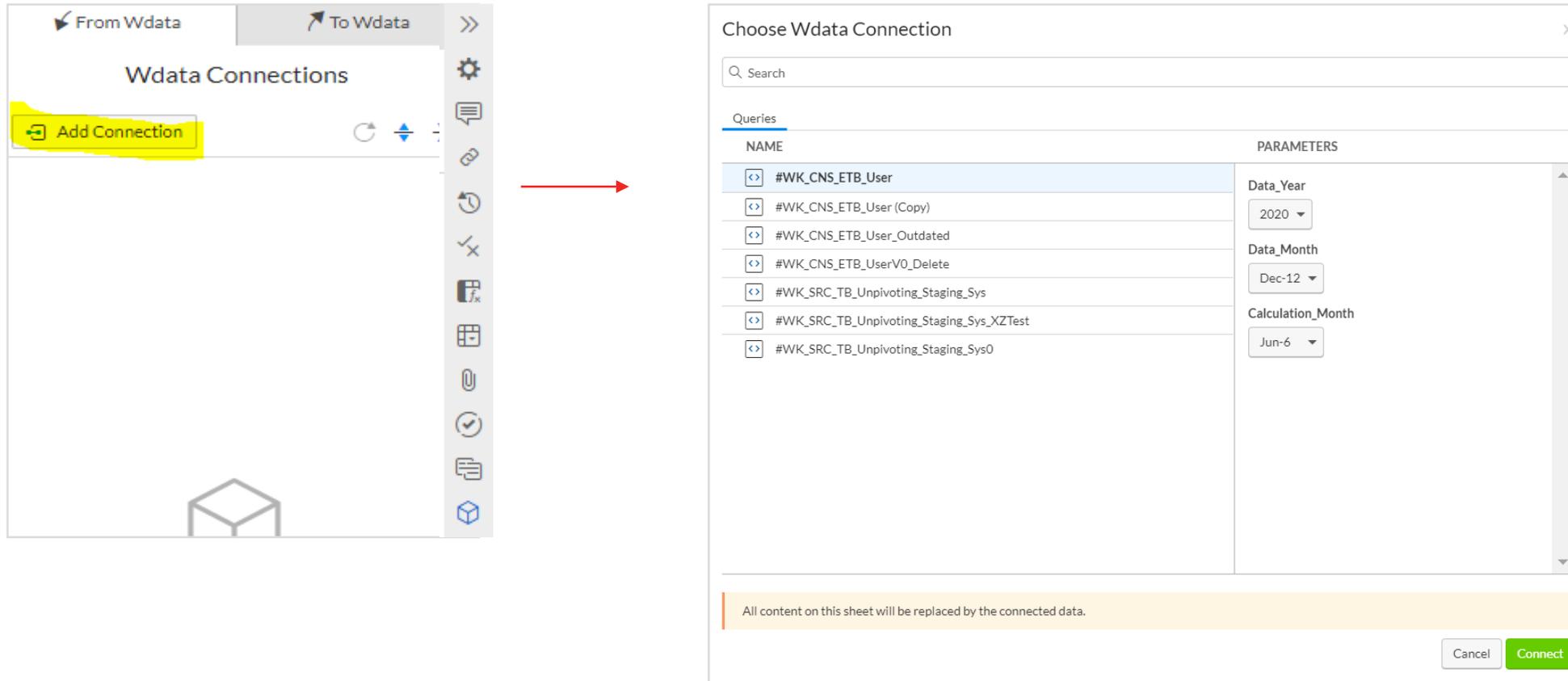
Replace Table IDs:

The screenshot shows the SQL Editor interface with a query in the SQL Editor pane. The query is a complex SQL statement with multiple CASE statements and COALESCE functions. The Sources panel on the left lists several tables: #WK_DIM_Cost Center, #WK_DIM_Product, #WK_CNS_TB_Sys, and #WK_DIM_Account. A red arrow points from a file named '#WK_CNS_TB_User.txt' to the Sources panel.

The screenshots show the process of replacing table IDs in the SQL Editor. The Sources panel on the left lists the tables: #WK_DIM_Cost Center, #WK_DIM_Product, #WK_CNS_TB_Sys, and #WK_DIM_Account. The SQL Editor pane shows the query with table IDs. The 'Add Table ID' button is highlighted in yellow, and the process is repeated for each table ID in the query.

If your query doesn't work, copy paste the provided script into the 'SQL Editor', but remember to replace the 'Wdata Table ID' at Line 244, 246, 249, 252. To replace the table ID, first highlight the entire line, right click on the '#WK_SRC_TB_Sys' table, and select "Add Table ID". Repeat for '#WK_DIM_Account', '#WK_DIM_Cost Center', '#WK_DIM_Account', '#WK_DIM_Product'.

Step 25: Connecting to ETB Spreadsheet



The image shows two screenshots from a software interface. The left screenshot shows the 'Wdata Connections' panel with the 'Add Connection' button highlighted in yellow. A red arrow points from this button to the right screenshot, which is a 'Choose Wdata Connection' dialog box. The dialog box has a search bar and a table of queries. The first query, '#WK_CNS_ETB_User', is selected. To the right of the table, there are three dropdown menus for parameters: 'Data_Year' (set to 2020), 'Data_Month' (set to Dec-12), and 'Calculation_Month' (set to Jun-6). At the bottom of the dialog, there is a warning message: 'All content on this sheet will be replaced by the connected data.' and two buttons: 'Cancel' and 'Connect'.

NAME	PARAMETERS
#WK_CNS_ETB_User	Data_Year: 2020
#WK_CNS_ETB_User (Copy)	Data_Month: Dec-12
#WK_CNS_ETB_User_Outdated	Calculation_Month: Jun-6
#WK_CNS_ETB_UserV0_Delete	
#WK_SRC_TB_Unpivoting_Staging_Sys	
#WK_SRC_TB_Unpivoting_Staging_Sys_XZTest	
#WK_SRC_TB_Unpivoting_Staging_Sys0	

In the 'ETB' spreadsheet, go to the top right panel and click on the 'Wdata Connection' icon. Add Connection and select the ETB query (#WK_CNS_ETB_User) to be connected to this spreadsheet.



Wdata Chains

Step 1: Starting Chains

The screenshot shows the Workiva user interface. On the left, a dark sidebar contains the user's name 'AU' and 'APAC SAs Utilities' with a 'Settings' icon. Below this are navigation links: 'Create', 'Home', 'Chains' (highlighted with a green circle), 'Configuration', and 'Wdesk'. At the bottom of the sidebar, there is a 'wdata' logo and user information for 'Xuan Zhi Choo / APAC De...', along with links for 'Send Feedback' and 'Support Center'. The main content area is titled 'Chains' and features a central graphic of three connected nodes. Below the graphic, text explains that chains are used to create workflows for connecting data. A prominent green button labeled 'Open Chain Builder' is centered below the text and is enclosed in a red rectangular box.

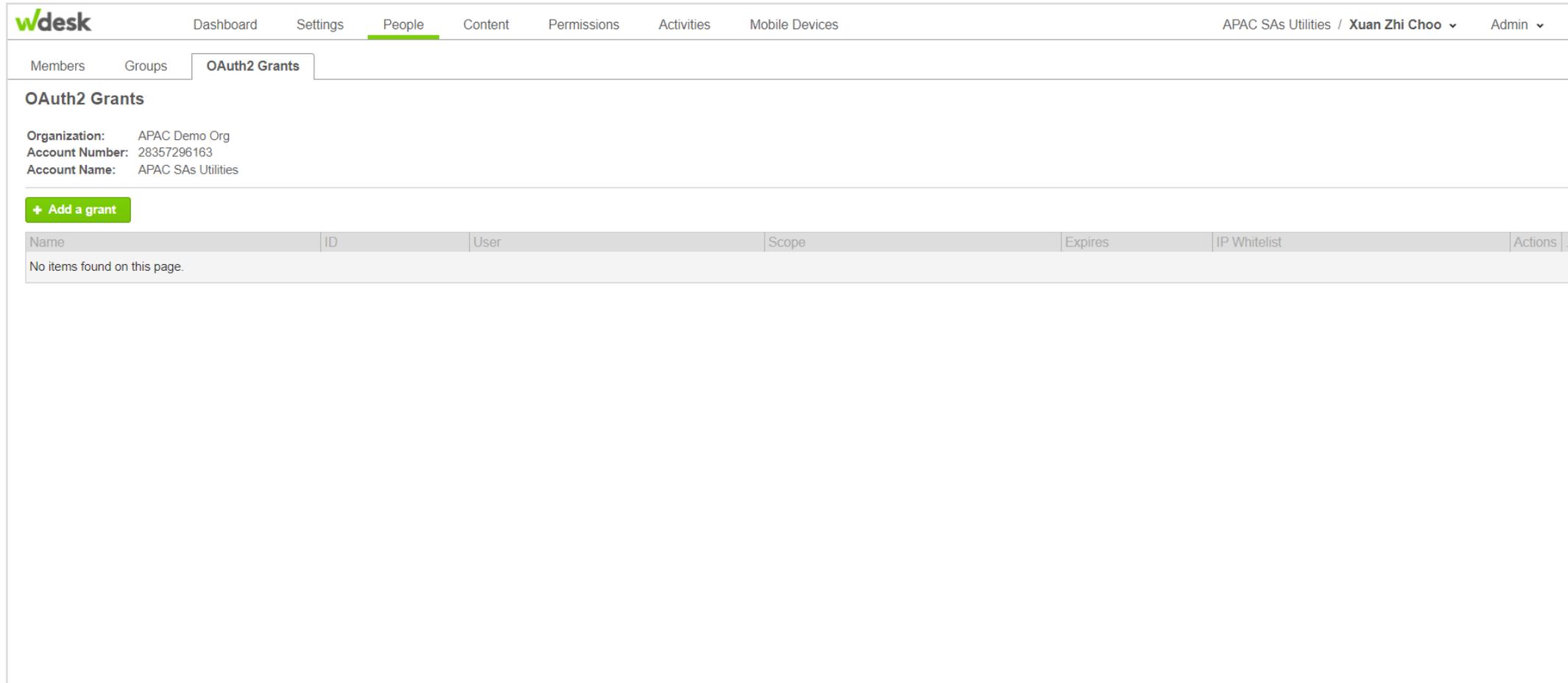
Start Chains by clicking 'Chains' on the left panel & click on 'Open Chain Builder'

Step 2.1: Create Workiva Connection

The screenshot shows the Wdata interface. On the left, there is a navigation sidebar with options: Create, Home, Chains, Configuration, and Wdesk. The main area displays a file explorer for 'Wdata' with tabs for All, Tables, Queries, and Views. A folder named '#Workiva' is visible. At the bottom left, a user menu is open, showing options: APAC DEMO ORG, Classic Profile, Switch Organizations, Binders Admin, Workspace Settings, Classic Wdesk (selected), and Sign Out. A tooltip for 'Classic Account Admin' is visible next to the 'Classic Wdesk' option. The right side of the interface shows a 'Details' panel with a database icon and the text 'Select an Item to View Details'.

At the Wdata homepage, click on your username on the bottom left and select 'Classic Account Admin'

Step 2.2: Create 'OAuth2 Grant' P1



wdesk Dashboard Settings **People** Content Permissions Activities Mobile Devices APAC SAs Utilities / Xuan Zhi Choo Admin

Members Groups **OAuth2 Grants**

OAuth2 Grants

Organization: APAC Demo Org
Account Number: 28357296163
Account Name: APAC SAs Utilities

[+ Add a grant](#)

Name	ID	User	Scope	Expires	IP Whitelist	Actions
No items found on this page.						

Creating an authentication grant by clicking 'Add a grant' here allows Wdata chains component to connect to Wdata & Wdesk via an API call.

Step 2.3: Create 'OAuth2 Grant' P2

The screenshot shows a web interface with a modal dialog titled "Add An OAuth2 Grant". The dialog contains the following fields and options:

- Grant Name:** Text input field containing "Workiva Integration".
- Username:** Dropdown menu showing "xuan.zhi@workiva.com". A red box highlights this field with an arrow pointing to it and the text: "Note: Username accounts (Service Acc) that has access to all tables & queries".
- Scope:** A list of checkboxes for various permissions, including "Spreadsheets (Read)", "Write Tasks", "Spreadsheets (Write)", "Read Audit API", "Write Audit API", "Graph API (Read)", "Graph API (Write)", "SCIM (Read)", "Data Entities (Read)", "Graph Admin Access", "Read Files", "Write Files", "Read Graph", "Write Graph", and "Read Tasks".
- Expiration:** Text input field containing "03/31/2030".
- IP Whitelist:** Text input field.

At the bottom of the dialog, there is a green "Create Grant" button and a grey "Cancel" button. A small asterisk and the word "required" are visible at the bottom left of the dialog.

Once 'Create Grant' is clicked, it generates a unique 'Client ID' & 'Client Secret' that would be inputted in the chains connection to establish the connection.

Step 2.4: Client ID & Client Secret

The screenshot shows the 'Edit OAuth2 Grant' modal window in the Workiva admin console. The modal is titled 'Edit OAuth2 Grant' and contains the following fields and options:

- Client ID:** cb20b0a7da744391a709d4189610e134
- Client Secret:** 39ec9472c34cc53c390e3675d6d4c9d8f9beb96f385613d3
- Grant Name:** Workiva Integration
- Username:** xuan.zhi@workiva.com
- Scope:** A list of permissions including: Spreadsheets (Read), Spreadsheets (Write), Read Audit API, Write Audit API, Graph API (Read), Graph API (Write), SCIM (Read), Data Entities (Read), Graph Admin Access, Read Files, Write Files, Read Graph, Write Graph, Read Tasks, and Write Tasks.
- Expiration:** 03/31/2031
- IP Whitelist:** (Empty field)

At the bottom of the modal, there are buttons for 'Save changes' and 'Cancel'. A '*required' label is visible at the bottom left of the modal area.

The background shows the 'wdesk' admin console with the 'OAuth2 Grants' tab selected. The left sidebar shows 'Members', 'Groups', and 'OAuth2 Grants'. The main content area shows a table with one entry: 'Workiva Integration' with ID 'cb20b0a7da744391a709d4189610e134'. The right sidebar shows 'APAC SAs Utilities / Xuan Zhi Choo' and 'Admin'.

Copy the 'Client ID' & 'Client Secret' - paste it under the 'Properties' section as shown in "Step 2.1"

Step 2.5: Creating Workiva Connection P1

The screenshot shows the 'Wdata chain builder' interface. On the left is a navigation sidebar with options: Home, Workspaces, Build, Templates, Monitor, Tasks, Schedules, and Connections (highlighted). The main area is titled 'Create Connection' and contains a 'BizApp Connection' section. Below this title is a list of BizApp connections: Tabular Transformation, Trintech, UltiPro, VTS Connector, Workday Connector, and Workiva. The 'Workiva' connection is selected and highlighted with a blue border. To the right of the list is a 'Runners' dropdown menu with 'CloudRunner' selected. Below the list is a 'Description' text input field. At the bottom right of the main area is a blue '+' button. Below the main area is a 'Properties' section with the text 'Input fields relevant to your connection'. It contains two input fields: 'ID' with the value 'cb20b0a7da744391a709d4189610e134' and a note 'The randomly generated ID in your OAuth2 Grant that is used to identify the integration user'; and 'Secret' with a masked value and a note 'The randomly generated Secret in your OAuth2 Grant that is used to authenticate the integration user'. At the top right of the main area are 'CANCEL' and 'SAVE' buttons. At the bottom left of the sidebar is an 'Admin' link with a gear icon.

Creating a BizApp connection enables 'Chains' to talk to the Wdata & Wdesk spreadsheets. Common BizApps connections include "Workiva", "JSON", "Tabular Transformation", "File Utils".

Step 2.6: Creating Workiva Connection P2

wdata chain builder

Update Connection CANCEL **SAVE** DELETE

`5Uaa1113U9ec4b4bb21ad86e3dddtd6 /`
The randomly generated ID in your OAuth2 Grant that is used to identify the integration user

Secret
.....
The randomly generated Secret in your OAuth2 Grant that is used to authenticate the integration user

Cerebral Host Override
`https://h.demo.wdesk.com/s/wdata/prep`
OPTIONAL: Use this field to override the URL host and base path for the Cerebral API when running in test environments

Spreadsheets Host Override
`https://api.demo.wdesk.com/spreadsheets/v1/`
OPTIONAL: Use this field to override the URL host and base path for the Spreadsheets API when running in test environments

IAM Host Override
`https://api.demo.wdesk.com/iam/v1/oauth2/token`
OPTIONAL: Use this field to override the URL host and base path for the IAM API when running in test environments

Admin Host Override
`https://api.demo.wdesk.com/platform/v1/users`
OPTIONAL: Use this field to override the URL host and base path for the Admin API when running in test environments

Graph DB Host Override
`https://api.demo.wdesk.com/graphdb/v1/`
OPTIONAL: Use this field to override the URL host and base path for the Graph DB API when running in test environments

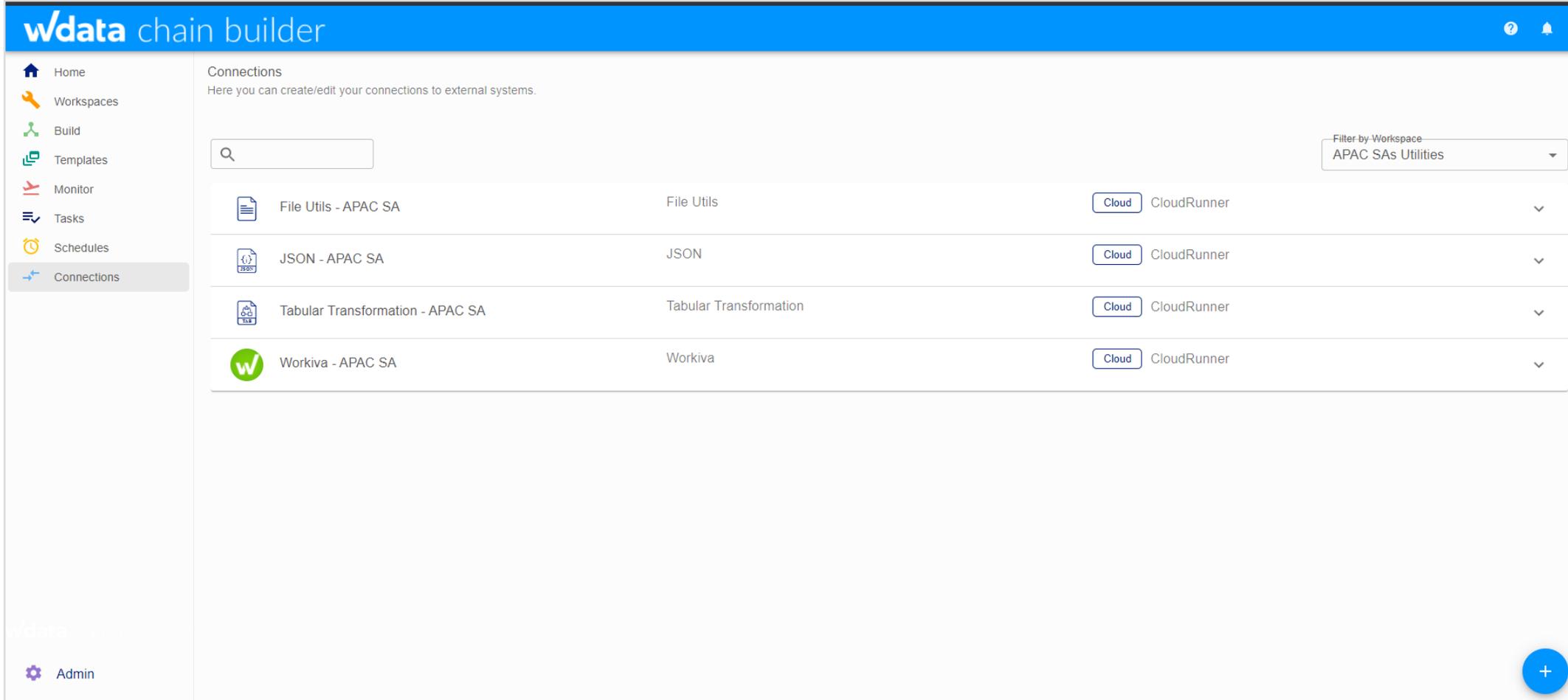
Environments Select All

Select workspaces and environments where your connection can be used

> Financial Reporting	<input type="checkbox"/>	> APAC SAs Utilities	<input checked="" type="checkbox"/>
		DEV	<input checked="" type="checkbox"/>

Lastly, enable all the relevant environments that the chain would be deployed to. Hit the 'Save' button on the top right and go on to create other BizApps.

Step 3: Setting up Bizapps



The screenshot shows the 'wdata chain builder' interface. On the left is a navigation sidebar with options: Home, Workspaces, Build, Templates, Monitor, Tasks, Schedules, and Connections (which is highlighted). Below the sidebar is an 'Admin' link. The main content area is titled 'Connections' and includes a search bar and a filter dropdown set to 'Filter by Workspace: APAC SAs Utilities'. A table lists four connections:

Icon	Name	Type	Cloud	Runner	Dropdown
	File Utils - APAC SA	File Utils	Cloud	CloudRunner	▼
	JSON - APAC SA	JSON	Cloud	CloudRunner	▼
	Tabular Transformation - APAC SA	Tabular Transformation	Cloud	CloudRunner	▼
	Workiva - APAC SA	Workiva	Cloud	CloudRunner	▼

A blue '+' button is located in the bottom right corner of the interface.

Common BizApps connections include "Workiva", "JSON", "Tabular Transformation", and "File Utils".



Wdata Utility Chains

Overview of control sheets

	A	B	C	D	E	F	G	H	I	J	K	L
1	Run_Chain	FileName	TableID	QueryID_OR_SQL	TagKey1	TagVal1	TagKey2	TagVal2	TagKey3	TagVal3	TagKey4	TagVal4
2	Yes	Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	a07950871751428a9df5f53811552e21	Data_Year	2020	Data_Month	Dec-12				
3	1		2	3	4	5	6	7	8	9	10	11
4												12

M	N	O	P	Q	R	S	T	U	V
ParamKey1	ParamVal1	ParamKey2	ParamVal2	ParamKey3	ParamVal3	ParamKey4	ParamVal4	Chain_Type	Last_Run
Data_Year	2020	Data_Month	Dec-12						
13	14	15	16	17	18	19	20		

Control sheets store the data variables that is required to run the chain (e.g. the file name of the files to be uploaded, ID of the table to which the output will be uploaded, ID of the query that should be executed, the tags and parameters used in tables and queries).

When starter chain (Chain 3) starts to run, it'll first identify the data variables in the control sheets. Note that we have 20 data variables in our example control sheets. To ease the burden of creating multiple starter chain which identifies 20 data variables, we usually create a chain template (illustrated in 'Chain 3.0: Creating Template Chain' section).

Creating Control Sheets for Training

For this training, the following Control Sheets would be created:

- i. Master Controls - Controls all the other control sheets below
- ii. TB EXT_SRC - Controls the External to Source chain
- iii. TB SRC_STG - Controls the Source to Staging chain
- iv. Refresh_Bud_Adj - Controls the Refresh Budget / Adjustment Chain

Master Controls

C2 X ✓ *f_x* =IF(B2="Manual","https://h.demo.wdesk.com/s/wdata/oc/app/apac-demo-org/workspace/654/environment/943/studio/chain/26795","No link, running Auto")

	A	B	C	D	E
1	Chain	Run_Mode	Manual_Chain_Link	Last_Run	
2	#WK_MASTER	Auto	No link, running Auto	On - 06/04/2021 At- 10:07 AM	
3					

Master Controls - Controls all the other control sheets below

TB EXT_SRC

A12 X ✓ f_x														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Run_Chain	FileName	TableID	QueryID_OR_SQL	TagKey1	TagVal1	TagKey2	TagVal2	TagKey3	TagVal3	TagKey4	TagVal4	ParamKey1	ParamVal1
2	Yes	Act_Dec-12_2020.csv	e36418c3a92846debcebbb3f3708a555		Data_Year	2020	Data_Month	Dec-12						
3	No	Act_Nov-11_2020.csv	e36418c3a92846debcebbb3f3708a555		Data_Year	2020	Data_Month	Nov-11						
4	No	Act_Dec-12_2019.csv	e36418c3a92846debcebbb3f3708a555		Data_Year	2019	Data_Month	Dec-12						

O	P	Q	R	S	T	U	V
ParamKey2	ParamVal2	ParamKey3	ParamVal3	ParamKey4	ParamVal4	Data_Type	Last_Run
						Act	
						Act	
						Act	

TB EXT_SRC - Controls the External to Source chain

TB SRC_STG

A2 X ✓ f_x Yes														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Run_Chain	FileName	TableID	QueryID_OR_SQL	TagKey1	TagVal1	TagKey2	TagVal2	TagKey3	TagVal3	TagKey4	TagVal4	ParamKey1	ParamVal1
2	Yes	Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	a07950871751428a9df5f53811552e21	Data_Year	2020	Data_Month	Dec-12					Data_Year	2020
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14

O	P	Q	R	S	T	U	V
ParamKey2	ParamVal2	ParamKey3	ParamVal3	ParamKey4	ParamVal4	Data_Type	Last_Run
Data_Month	Dec-12						On - 06/04/2021
15	16	17	18	19	20		

TB SRC_STG - Controls the Source to Staging chain

Refresh_Bud_Adj

B19 X ✓ fx														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Run_Chain	FileName	TableID	SheetID	TagKey1	TagVal1	TagKey2	TagVal2	TagKey3	TagVal3	TagKey4	TagVal4	ParamKey1	ParamVal1
2	Yes ▾	Bud_Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	e91d388e5e5b4f74924eb3345829c657	Data_Year	2020	Data_Month	Dec-12						
3	Yes ▾	Adj_Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	857b6c33458944b5b81c4277bcac5baa	Data_Year	2020	Data_Month	Dec-12						
4														

O	P	Q	R	S	T	U	V
ParamKey2	ParamVal2	ParamKey3	ParamVal3	ParamKey4	ParamVal4	Data_Type	Last_Run
						Bud	
						Adj	

Refresh_Bud_Adj - Controls the Refresh Budget / Adjustment Chain

Intro to Wdata Utility Chains P1



Identify Variables

- Reads Chains Control sheet and identifies the data variables:
 - Run Chain (Yes/No)
 - Wdata Table
 - Wdata Query
 - Tags
 - Parameters



Checks and Removes Redundant File(s)

- Un-import the file from the Wdata Source Table
- Deletes the outdated/ redundant file(s) which would be replaced with the new file with the same file name

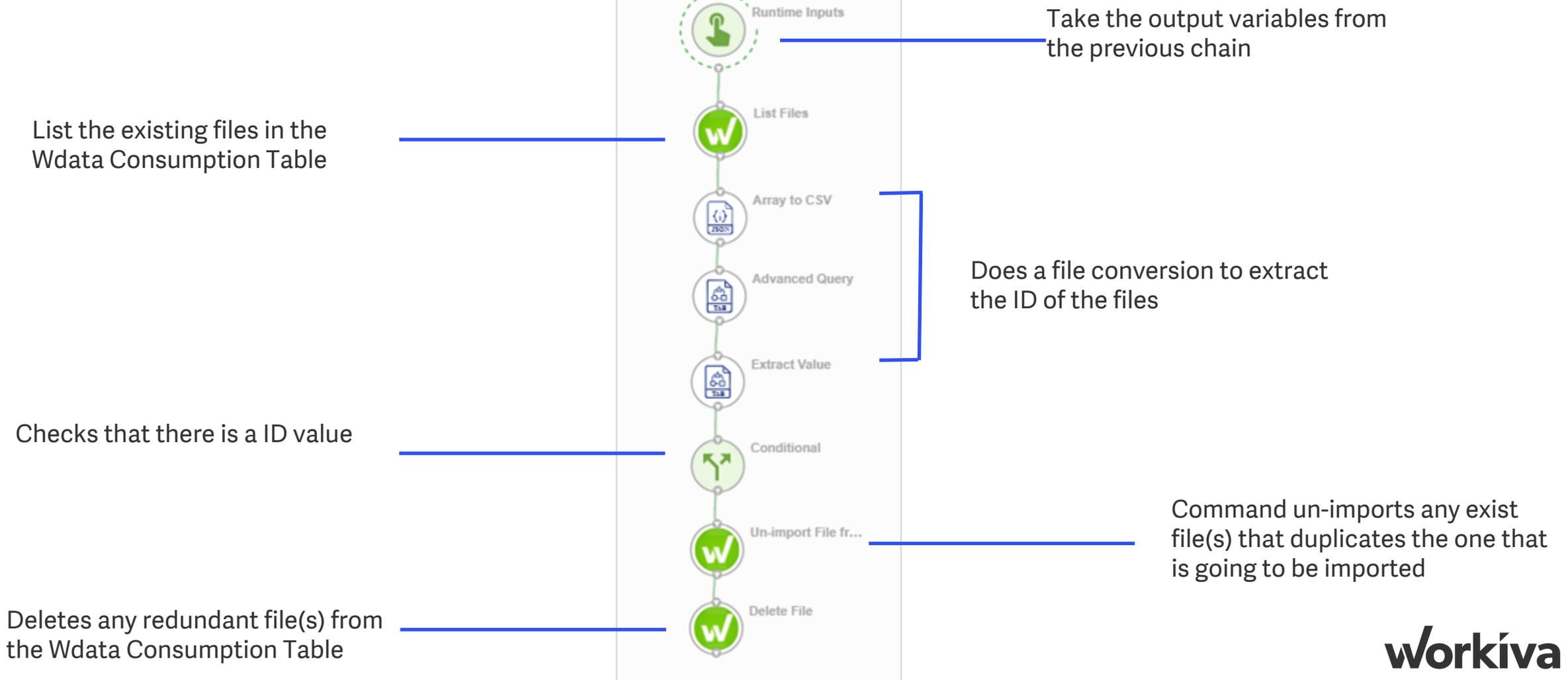


Run Query and Upload to the Wdata Table

- Runs the transformation query that transposes the data with the month data going down across to 12 columns
- Uploads to the results of the query to the Wdata Consumption table

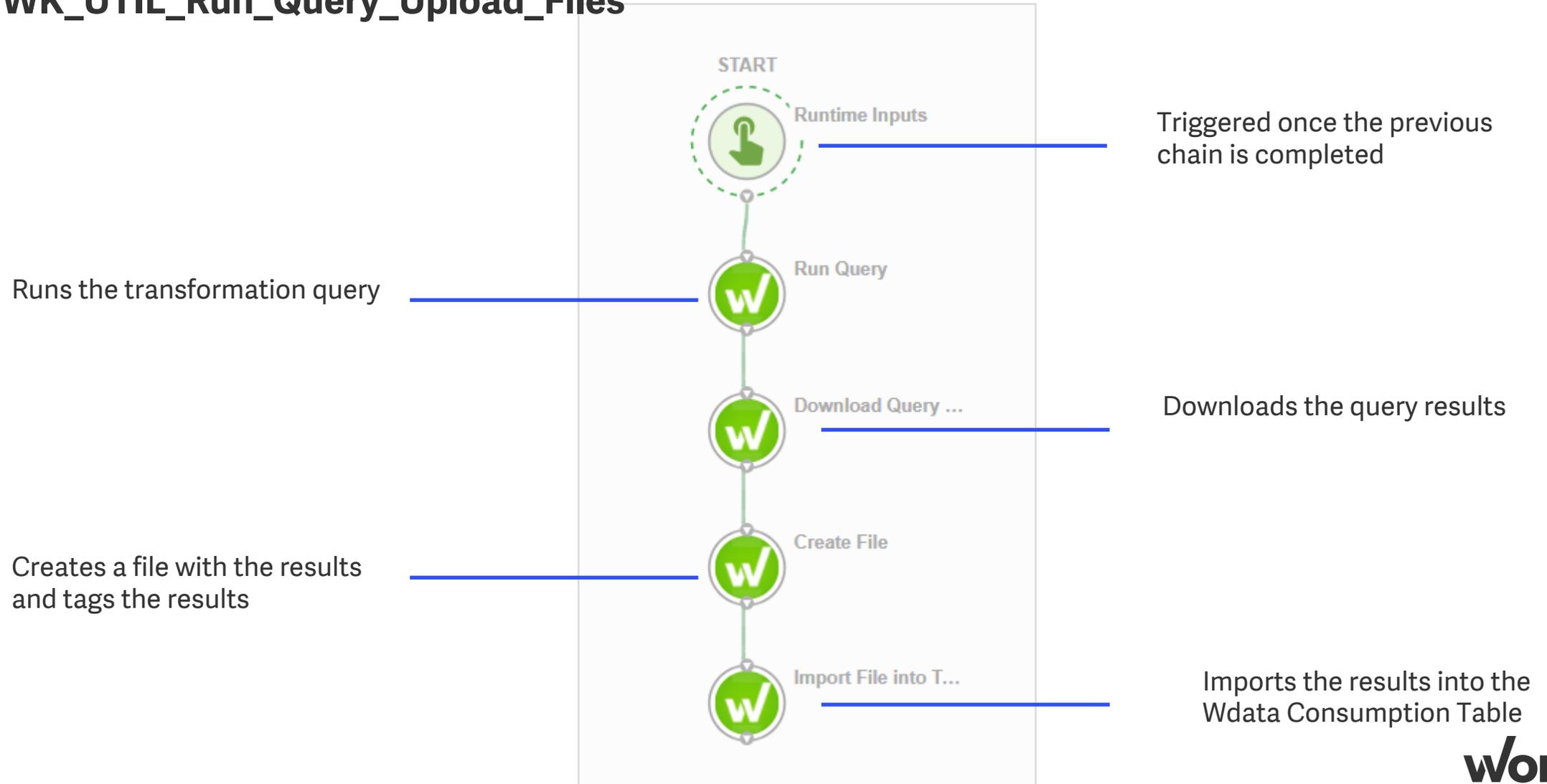
Chain 1: Checks and Removes Redundant File(s)

#WK_UTIL_Manage_Redundant_Files



Chain 2: Run Query & Upload to Wdata Table

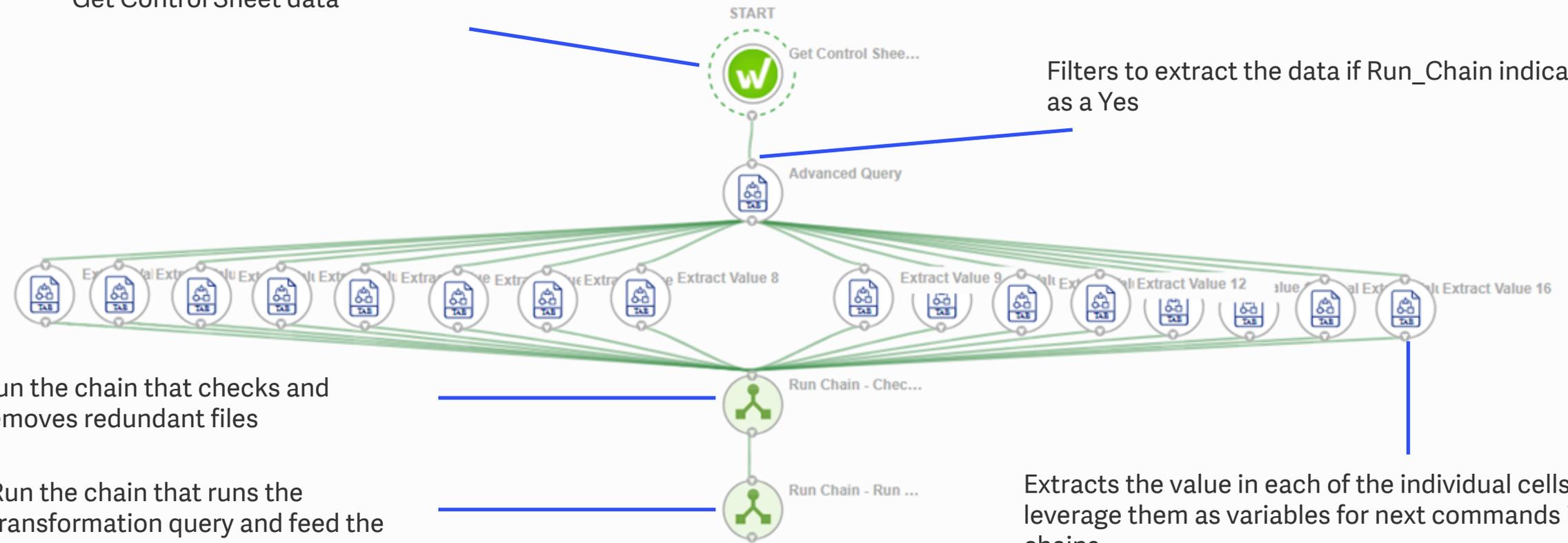
#WK_UTIL_Run_Query_Upload_Files



Chain 3: Identify Variables

#WK_TB_SRC_STG

Get Control Sheet data



Filters to extract the data if Run_Chain indicated as a Yes

Run the chain that checks and removes redundant files

Run the chain that runs the transformation query and feed the results into the Wdata consumption table

Extracts the value in each of the individual cells and leverage them as variables for next commands in the chains

Upload Resources

APAC Demo Org APAC SAS Utilities DEV MONITOR

Home Workspaces Build Chains Resources Templates Monitor Tasks Schedules Connections

Upload Resources CANCEL CREATE

Click or drag to upload files

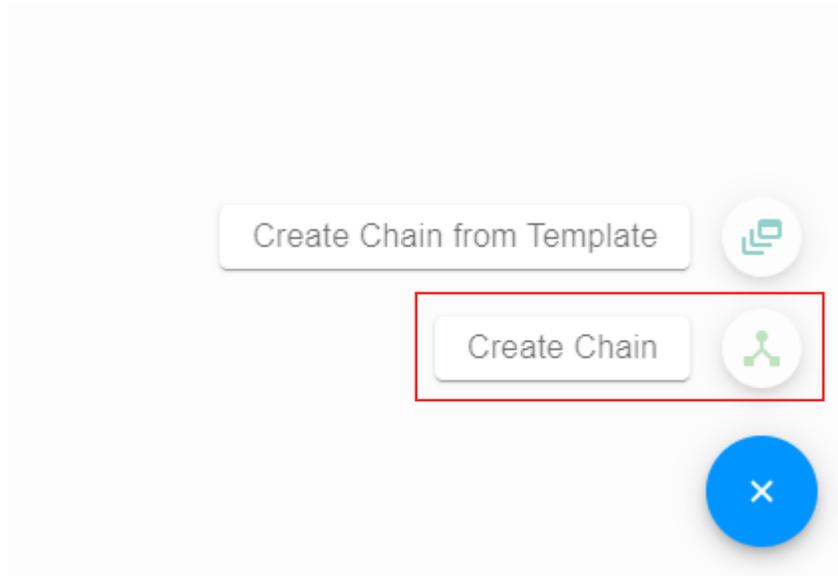
Workspace Resources + Name

Name	Created	Updated	Actions
Cost_Center_Dimension.csv	March 29th 2021 11:57 am	March 29th 2021 11:57 am	
FS_Mapping.csv	March 29th 2021 11:57 am	March 29th 2021 11:57 am	
Product_Dimension.csv	March 29th 2021 11:57 am	March 29th 2021 11:57 am	
TB_11_2020.csv	March 29th 2021 11:55 am	March 29th 2021 11:55 am	
TB_12_2019.csv	March 29th 2021 11:55 am	March 29th 2021 11:55 am	
TB_12_2020.csv	March 29th 2021 11:55 am	March 29th 2021 11:55 am	

Click on "Build">"Resources" and upload the TB data and dimension tables as shown above. This would serve as a starting point for chains.

Chain 1: Manage Redundant Files (UTIL)

Step 1: Create New Chain



Edit Chain CANCEL SAVE

i *✉* *↓*

Setup

Name

Description

Allow concurrent runs Disable execution

<> Variables +

Name	Value	Encrypt	Actions
------	-------	---------	---------

<> Dynamic Variables +

Name	Initial Value	Actions
------	---------------	---------

In "Build">"Chains" page, click the blue "+" button on the lower right corner to "Create Chain". Name the new chain as "#WK_UTIL_Manage_Redundant_Files" and hit save.

Step 2: Create List Files Node P1

Edit Runtime Inputs
TriggerEvent

DELETE CANCEL SAVE

Basic Info

Name
Runtime Inputs

Description (optional)

<> Variables

TextField

Display Name
r_FileName Required

Description
The FileName to validate in the Table

Default Value

TextField

Display Name
r_TableID Required

Description
The TableID against which the FileName needs to be Validated

Default Value

Create a "Runtime Inputs" node and drag the node in the "Start" circle. Fill in the node information as shown above. Specifically, create two Text Field variables: "r_FileName" and "r_TableID".

Step 3: Create List Files Node P2

The screenshot displays the Workiva interface for configuring a node. The top bar shows the workspace name "#WK_UTIL_Manage_Redundant_Files" and the environment "APAC SAs Utilities Environment: DEV". The main panel is titled "Edit List Files" and includes a "Workiva - List Files" node. The configuration is as follows:

- Name:** List Files
- Description (optional):** (empty)
- Iterations:** (toggle off)
- Command Properties:** Workiva - APAC SA, CloudRunner
- Table ID:** r_TableID (selected from the left panel)

A red box highlights the "R_TableID" variable in the "Runtime Inputs" section of the left panel, with a red arrow pointing to the "Table ID" field in the main configuration panel.

Create a "Workvia">"List Files" node and link the node with the Runtime Inputs node. Fill in the node information as shown above. Specifically for Table ID, choose the variable "r_TableID" under "Runtime Inputs" from the left panel.

Step 4: Create Array to CSV node P1

The screenshot displays the configuration interface for a workflow node titled "Edit Array to CSV" (JSON - Array to CSV). The left sidebar shows a tree view of variables, with "Files List" highlighted in red. The main configuration area includes:

- Name:** Array to CSV
- Description (optional):** (empty field)
- Iterations:** (toggle set to off)
- Command Properties:** JSON - APAC SA (selected) and CloudRunner (selected)
- JSON Data:** Files list (selected from the left sidebar)
- Input Text (Deprecated):** (empty field)
- Path to root:** (empty field)

Buttons for PUBLISH, EXECUTE, CHAIN SETTINGS, DELETE, CANCEL, and SAVE are visible at the top right of the node configuration panel.

Create a "JSON">"Array to CSV" node and link the node with the List Files node. Fill in the node information as shown above. Specifically for JSON Data, choose the variable "Files List" under "List Files" from the left panel.

Step 4.1: Create Array to CSV node P2

 Edit Array to CSV
JSON - Array to CSV

DELETED CANCEL SAVE

Filter

Optionally add a filter to the array of items. Leave this blank to apply no filter. (e.g. ?(@.price < 1000))

Multi-value Delimiter

, ↵

If there are multiple values retrieved for a single JSONPath in a column, they will be separated by this delimiter.

Preview Result ?

Columns

The columns to produce and their corresponding JsonPaths

ADD REMOVE ALL

Column Name	ID ↵	JSONPath	.id ↵
The name for the column in the CSV		The JSONPath to the value(s) in each item in the array. (i.e. if your array contains objects such as { "name": "OneCloud" } you would use ".name" as the path)	
Column Name	Name ↵	JSONPath	.name ↵
The name for the column in the CSV		The JSONPath to the value(s) in each item in the array. (i.e. if your array contains objects such as { "name": "OneCloud" } you would use ".name" as the path)	

REMOVE REMOVE

Delimiter

Comma

The delimiter to use in the output tabular dataset.

Continue to fill in the node information as shown above. Specifically, add two sets of fields in the Columns section. Also, make sure the "Comma" is chosen as the Delimiter.

Step 5: Create Advanced Query Node P1

#WK_UTIL_Manage_Redundant_Files
Workspace: APAC SAs Utilities Environment: DEV

PUBLISH EXECUTE CHAIN SETTINGS

DELETE CANCEL SAVE

Edit Advanced Query
Tabular Transformation - Advanced Query

Name
Advanced Query

Description (optional)

Iterations

Command Properties

Tabular Transformation - APAC SA CloudRunner

Tables
Add all of the files that will be used in the query, as well as their table name.

File
Converted File
The file to add add.

Table Name
FileList
The name to use as the table name.

REMOVE

Query
Select * from FileList where Name = 'r_FileName'

The SQL query to execute. INSERT, UPDATE, CREATE are not supported.

Input Delimiter
Comma
The delimiter of the input CSV file, as well as the join files.

Select * from FileList where Name =

Create a "Tabular Transformation">"Advanced Query" node and link the node with the Array to CSV node. Fill in the node information as shown above. Specifically for File, choose the variable "Converted File" under "Array to CSV" from the left panel. Also, replicate the query as shown above and use the variable "r_FileName" in the query.

Step 5.1: Create Advanced Query Node P2

Input Delimiter

Comma



The delimiter of the input CSV file, as well as the join files.

Output Delimiter

Comma



The delimiter to use for the result of the query.

Preview results [?](#)

Make sure that "Comma" is chosen for both the Input Delimiter and Output Delimiter. Check the box for preview results.

Step 6: Create Extract Value node

The screenshot displays the Workiva interface for configuring an 'Extract Value' node. On the left, a sidebar lists variables, with 'Result' highlighted in a red box. Red arrows point from this box to the 'Input file' field in the main configuration area. The main configuration area is titled 'Edit Extract Value' and includes fields for Name, Description, Iterations, Command Properties, Input file, Column Index, Delimiter, and Row Index. The 'Input file' field is set to 'Result', 'Column Index' is set to '1', and 'Row Index' is set to '2'. The 'Command Properties' section shows 'Tabular Transformation - APAC SA' and 'CloudRunner'.

Create a "Tabular Transformation">"Extract Value" node and link the node with the Advanced Query node. Fill in the node information as shown above. Specifically for Input File, choose the variable "Result" under "Advanced Query" from the left panel. Make sure the Column Index is set as 1 and Row Index is set as 2.

Step 6.1: Error Handling

The screenshot displays the configuration interface for a node in a data pipeline. The node is titled "Edit Extract Value" and is part of a "Tabular Transformation - Extract Value" task. The interface includes a search bar on the left, a status bar at the top right with "PUBLISHED" and "EXECUTE" buttons, and a "CHAIN SETTINGS" link. The main configuration area is divided into several sections: "Error actions" (set to "Continue with chain"), "Timeout conditions" (set to "Max time for task to run" of 3600), "Retry attempts", "Pause between attempts", and "Error conditions". The "Error conditions" section is highlighted with a red box and contains two checked options: "Invalid arguments" and "General failure error". At the bottom, there is an "Output Interpretation" section with a download icon and a plus sign to add more options.

For error handling purpose, in the "Extract Value" node, click on the 'Triangle' tab and tick both 'Invalid arguments' & 'General failure error'. This would ensure the chain runs regardless.

Step 7: Create Conditional node

The screenshot displays the 'Edit Conditional' node configuration in the Workiva interface. On the left sidebar, the 'Value' node is highlighted with a red box. The main panel shows the configuration for a 'Conditional' node, including fields for Name, Description, and Conditions. The 'Conditions' section is expanded, showing a rule for 'String' data type with the variable 'Value' and the operation 'Is Not Blank'. A red arrow points from the 'Value' node in the sidebar to the 'Value' variable in the condition. The 'Is Not Blank' operation is also highlighted with a red box.

Create a "Conditional" node and link the node with the Extract Value node. Fill in the node information as shown above. Specifically for String Value, choose the variable "Value" under "Extract Value" from the left panel. Make sure to choose "is not blank" in the operation to test.

Step 8: Create Un-import File from Table node

The screenshot displays the 'Edit Un-import File from Table' configuration window in the Workiva interface. The window title is 'Workiva - Un-import File from Table' and it includes 'DELETE', 'CANCEL', and 'SAVE' buttons. The configuration is organized into several sections:

- Basic Info:** Name: 'Un-import File from Table'; Description (optional): empty.
- Iterations:** Includes a refresh icon, a toggle switch, and an eye icon.
- Command Properties:** Shows 'Workiva - APAC SA' and 'CloudRunner'.
- Table ID:** Set to 'r_TableID' with a dropdown arrow. Below it, the text reads 'The ID of the table to import the file into.'
- File ID:** Set to 'Value' with a dropdown arrow. Below it, the text reads 'The ID of the file'.

On the left side, the 'Select a variable' menu is visible. Under 'Value', the 'Value' variable is highlighted with a red box. Under 'Runtime Inputs', the 'R_TableID' variable is also highlighted with a red box. Red arrows point from these boxes to the corresponding fields in the main configuration panel.

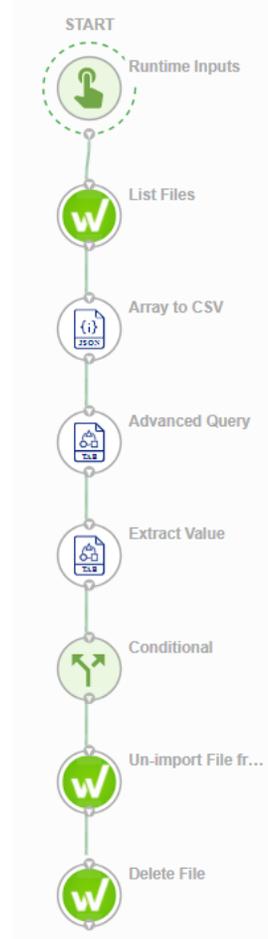
Create a "Workiva">"Un-import File from Table" node and link the node with the Conditional node. Fill in the node information as shown above. Specifically for Table ID, choose the variable "r_TableID" under "Runtime Inputs". For File ID, choose the variable "Value" under "Extract Value" from the left panel.

Step 9: Create Delete File node

The screenshot displays the 'Edit Delete File' configuration window in the Workiva interface. On the left, a sidebar lists various nodes, with 'Value' selected under the 'Extract Value' category. The main panel shows the configuration for the 'Delete File' node. The 'Name' field is set to 'Delete File'. The 'Iterations' section is visible with a toggle switch. The 'Command Properties' section shows the provider 'Workiva - APAC SA' and the engine 'CloudRunner'. The 'File ID' field is set to 'Value', with a tooltip indicating 'The ID of the file to delete.' A red arrow points from the 'Value' variable in the left sidebar to the 'File ID' field in the main panel.

Create a "Workiva">"Delete File" node and link the node with the Un-import File node. Fill in the node information as shown above. Specifically for Table ID, choose the variable "Value" under "Extract Value" on the left panel.

Step 10: Complete Chain 1 - Manage Redundant Files



Double check the structure of the chain. Remember to hit Publish.

Chain 2: Run Query & Upload Files (UTIL)

Step 1: Create New Chain

Edit Chain CANCEL SAVE

i *✉* *↓*

Setup

Name
#WK_UTIL_Run_Query_Upload_Files

Description

Allow concurrent runs Disable execution

🕒 Schedules +

<> Variables +

Name	Value	Encrypt	Actions
------	-------	---------	---------

<> Dynamic Variables +

Name	Initial Value	Actions
------	---------------	---------

Create a new chain "#WK_UTIL_Run_Query_Upload_Files" as shown above and hit save.

Step 2: Create Runtime Inputs Node

Edit Runtime Inputs
TriggerEvent

DELETED CANCEL SAVE

Basic Info

Name
Runtime Inputs

Description (optional)

<> Variables

TextField ⋮ ✖

Display Name
r_FileName Required

Description

Default Value

TextField ⋮ ✖

Display Name
r_TableID Required

Description

Default Value

TextField ⋮ ✖

Display Name
r_QueryID Required

Description

Default Value

TextField ⋮ ✖

Display Name
r_TagKey1 Required

Description

Default Value

Create a "Runtime Inputs" node and drag the node in the "Start" circle. Fill in the node information as shown above. Specifically, create 19 Text Field variables: "r_FileName", "r_TableID", "r_QueryID", "r_TagKey1", "r_TagVal1", "r_TagKey2", "r_TagVal2", "r_TagKey3", "r_TagVal3", "r_TagKey4", "r_TagVal4", "r_ParamKey1", "r_ParamVal1", "r_ParamKey2", "r_ParamVal2", "r_ParamKey3", "r_ParamVal3", "r_ParamKey4" and "r_ParamVal4". Check "Required" for the first three variables.

Step 3: Create Run Query Node

The screenshot displays the 'Edit Run Query' configuration page in the Workiva interface. On the left, a sidebar lists 'Runtime Inputs' including variables like R_FileName, R_TableID, R_QueryID, and R_ParamKey1. The main panel shows the configuration for a 'Run Query' node. The 'Query ID' is set to 'r_QueryID'. Below, four parameter pairs are defined, each with a 'Key' and a 'Value' field. Red arrows point from 'R_QueryID' in the sidebar to the 'Query ID' field. Green arrows point from 'R_ParamKey1' and 'R_ParamVal1' in the sidebar to their respective fields in the first parameter pair. The interface includes a 'Run Asynchronously' checkbox and buttons for 'DELETE', 'CANCEL', and 'SAVE'.

Create a "Workiva">"Run Query" node and link the node with the Runtime Inputs node. Fill in the node information as shown above. Specifically, create 4 sets of parameters and select the corresponding Parameter Key and Parameter Value as shown above.

Step 4: Create Download Query Result Node

The screenshot displays the 'Edit Download Query Result' configuration window in the Workiva interface. On the left, a sidebar lists variables under the 'Run Query' > 'Query Result' category. The variable 'Id' is highlighted with a red box, and a red arrow points from it to the 'Query Result ID' field in the main configuration area. The main area shows the following fields:

- Name:** Download Query Result
- Description (optional):** (empty text area)
- Iterations:** (empty text area with a toggle and eye icon)
- Command Properties:** Workiva - APAC SA (selected) and CloudRunner (selected)
- Query Result ID:** Query Result (selected) with a dropdown arrow. Below this field is the text: 'The ID of the query result.'

Create a "Workiva">"Download Query Result" node and link the node with the Run Query node. Fill in the node information. Specifically, for Query Result ID choose the variable "Id" under "Run Query">"Query Result" from the left panel.

Step 5: Create Create File Node

The screenshot displays the 'Edit Create File' configuration page in the Workiva interface. On the left sidebar, the 'Query Result' node is selected and highlighted with a red box. Below it, under 'Runtime Inputs', 'R_FileName' and 'R_TableID' are also highlighted with red boxes. Red arrows point from these boxes to the 'File' and 'Name' fields in the main configuration area. The main configuration area shows the following details:

- Name:** Create File
- Description (optional):**
- Iterations:** (toggle off)
- Command Properties:** Workiva - APAC SA, CloudRunner
- Table ID:** r_TableID
- File:** Query Result
- Name:** r_FileName
- Download URL:**

Create a "Workiva">"Create File" node and link the node with the Download Query Result node. Fill in the node information as shown above.

Step 6: Create Import File into Table Node P1

The screenshot displays the 'Edit Import File into Table' configuration page in the Workiva interface. The left sidebar shows a list of variables, with 'Id' under the 'Created' category highlighted in a red box. A red arrow points from this box to the 'File ID' field in the main configuration area, which is set to 'Result *'. Other fields include 'Table ID' set to 'r_TableID', 'Name' set to 'Import File into Table', and 'Command Properties' set to 'Workiva - APAC SA' and 'CloudRunner'. The 'Run Asynchronously' checkbox is unchecked.

Create a "Workiva">"Create File" node and link the node with the Download Query Result node. Fill in the node information as shown above. Specifically for File ID, choose the variable "Id" under "Create File">"Result" from the left panel. For Table ID, choose the variable "r_TableID" under "Trigger">"Runtime Inputs".

Step 6.1: Create Import File into Table Node P2

Stay in the previous command

The screenshot displays a software interface with a 'Runtime Inputs' panel on the left and a configuration area on the right. The 'Runtime Inputs' panel lists several variables: R_FileName, R_TableID, R_QueryID, R_TagKey1, R_TagVal1, R_TagKey2, R_TagVal2, R_TagKey3, and R_TagVal3. The variables R_TagKey1, R_TagVal1, R_TagKey2, and R_TagVal2 are highlighted with green boxes. Red arrows point from these boxes to the 'Tags' configuration area. The 'Tags' area contains two rows, each with a 'Key' and a 'Value' field. The first row has 'r_TagKey1' in the Key field and 'r_TagVal1' in the Value field. The second row has 'r_TagKey2' in the Key field and 'r_TagVal2' in the Value field. Green arrows point from the 'r_TagKey1' and 'r_TagVal1' fields to their respective input fields in the 'Tags' area. The interface also shows a 'Result' dropdown, 'Column Mappings', and a 'Run Asynchronously' checkbox.

Add two tags in the bottom and fill in the tag information as shown above. In here, we are adding tags "Data_Year" and "Data_Month".

Step 7: Complete Chain 2 - Run Query & Upload Files



Double check the structure of the chain. Remember to hit Publish.

Chain 3: File Importer (UTIL)

Step 2: Runtime Inputs

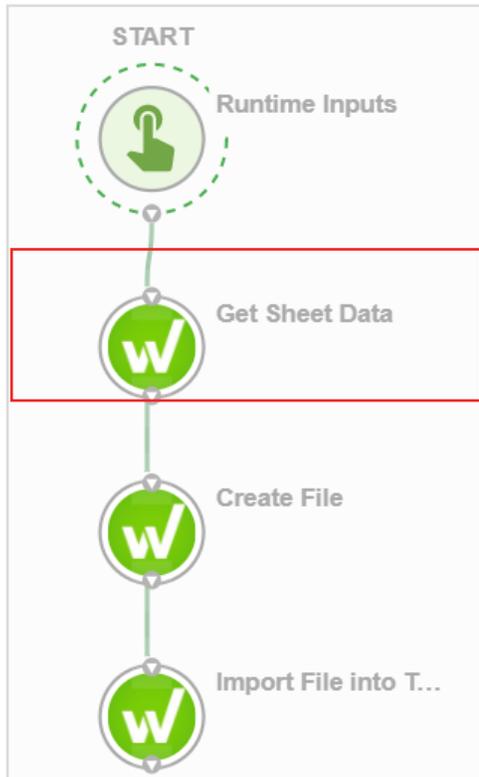


The screenshot shows the configuration for a 'TriggerEvent' node named 'Runtime Inputs'. The 'Variables' section contains five 'TextField' inputs, each with a 'Required' checkbox checked and highlighted by a red box:

- Display Name: r_FileName
- Display Name: r_TableID
- Display Name: r_SSID
- Display Name: r_SID
- Display Name: r_Region

Input a 'Runtime Inputs' node and indicate 5 fields as "Required". The five fields are: 'r_FileName, r_TableID, r_SSID, r_SID, r_Region'

Step 3: Get Sheet Data



The screenshot shows the configuration interface for the 'Get Sheet Data' node. On the left, a list of runtime inputs is shown, with the first five fields marked as 'Required' (indicated by a 'T' in a circle): R_FileName, R_TableID, R_SSID, R_SID, and R_Region. The main configuration area on the right has several fields highlighted with red boxes: 'Iterations', 'Sheet IDName', 'Calculated', and 'Revision'. The 'Iterations' field is empty. The 'Sheet IDName' field has a dropdown menu. The 'Calculated' field has a dropdown menu. The 'Revision' field has a dropdown menu.

Input a 'Runtime Inputs' node and indicate 5 fields as "Required". The five fields are: 'r_FileName, r_TableID, r_SSID, r_SID, r_Region'

Step 4: Create File



The screenshot shows the 'Edit Create File' configuration interface in Workiva. The interface is divided into several sections:

- Basic Info:** Includes fields for 'Name' (Create File) and 'Description (optional)'.
- Iterations:** Includes a toggle switch.
- Command Properties:** Shows the command 'Workiva - APAC SA2' and the runner 'CloudRunner'.
- Input Fields:** Three fields are highlighted with red boxes:
 - Table ID:** Contains the variable `{_TableID}`. A red arrow points from the `R_TableID` input in the left sidebar to this field.
 - File:** Contains the variable `{Data}`. A red arrow points from the `Data` input in the left sidebar to this field.
 - Name:** Contains the variable `{_FileName}`. A red arrow points from the `R_FileName` input in the left sidebar to this field.
- Download URL:** An optional field for downloading files.

Connect a 'Create File' node and input the following input variables from the previous 'Get Sheet Data' & 'Runtime Inputs'

Step 5: Import File



#WK_UTIL_File_Importer
Workspace: APAC SAs Utilities Environment: DEV

Result

- ColumnMappings
 - Created
 - Id
 - Key
- Metadata
 - Name
 - NumErrors
 - NumRecords
 - OriginalFileSize
- Source
 - Status
 - TableId
- Tags
 - Updated
 - UserId
 - Version

Get Sheet Data

Runtime

Resources

Trigger

Runtime Inputs

- R_FileName
- R_TableID

Edit Import File into Table

Workiva - Import File into Table

Basic Info

Name: Import File into Table

Description (optional)

Iterations

Command Properties

Workiva - APAC SA2

Table ID: r_TableID
The ID of the table to import the file into.

File ID: Result *
The ID of the file

Column Mappings

Run Asynchronously

Tags

- Key: r_TagKey1 = Value: r_TagVal1
- Key: r_TagKey2 = Value: r_TagVal2

Variable Transform: Result
Command Output: Create File

Input	Transformation	Output	Value
<>	Get Value from JSON	T	= id

SELECT transformation +

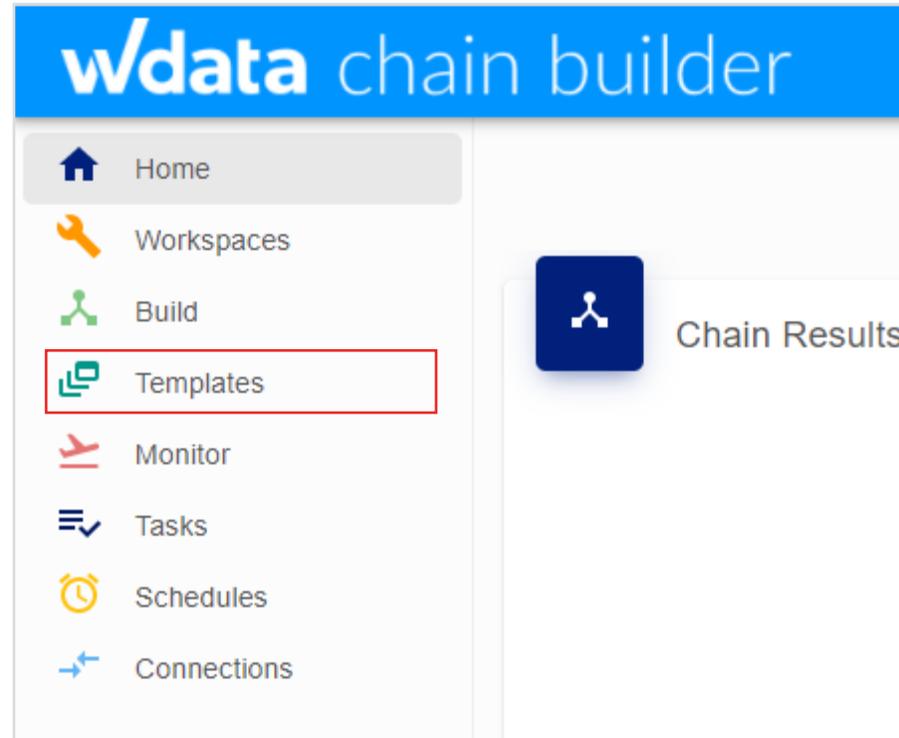
CANCEL ACCEPT

Connect a 'Import File' node and input the following input variables from the previous 'Create File' & 'Runtime Inputs'



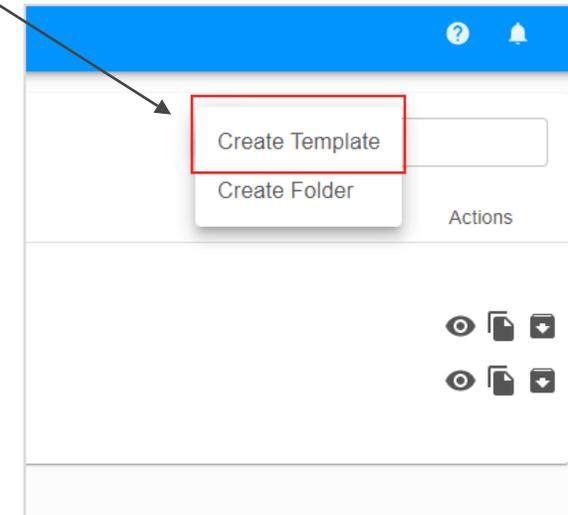
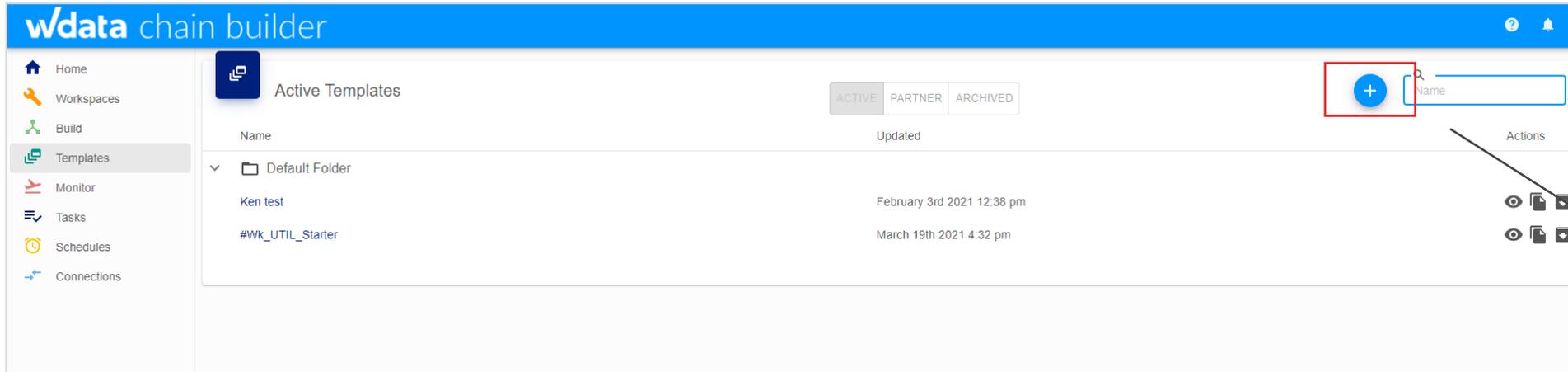
Chain 4: Creating Template Chain

Step 1: Go to Templates



Open the Chain Builder and click "Templates" on the left panel.

Step 2: Create Template



In the Templates homepage, click the blue "+" button on the right and click "Create Template".

Step 3: Edit Template Settings

The screenshot shows the 'Edit Template' interface. At the top right, there are 'CANCEL' and 'SAVE' buttons. Below the header, there is a folder icon labeled 'Default Folder' with an edit icon. The main form contains the following fields:

- Name:** #WK_UTIL_Starter
- Description:** Basic Control Sheet Reader
- Template Variables:** A table with columns for Name, Value, Type, and Actions.

Name	Value	Type	Actions
Control_SSID		<input type="checkbox"/> Dynamic <input type="checkbox"/> Resource	× ⋮
Control_SID		<input type="checkbox"/> Dynamic <input type="checkbox"/> Resource	× ⋮
Control_Region	A1:T	<input type="checkbox"/> Dynamic <input type="checkbox"/> Resource	× ⋮

Red arrows point to the Name, Description, and the Name and Value fields of the Template Variables table.

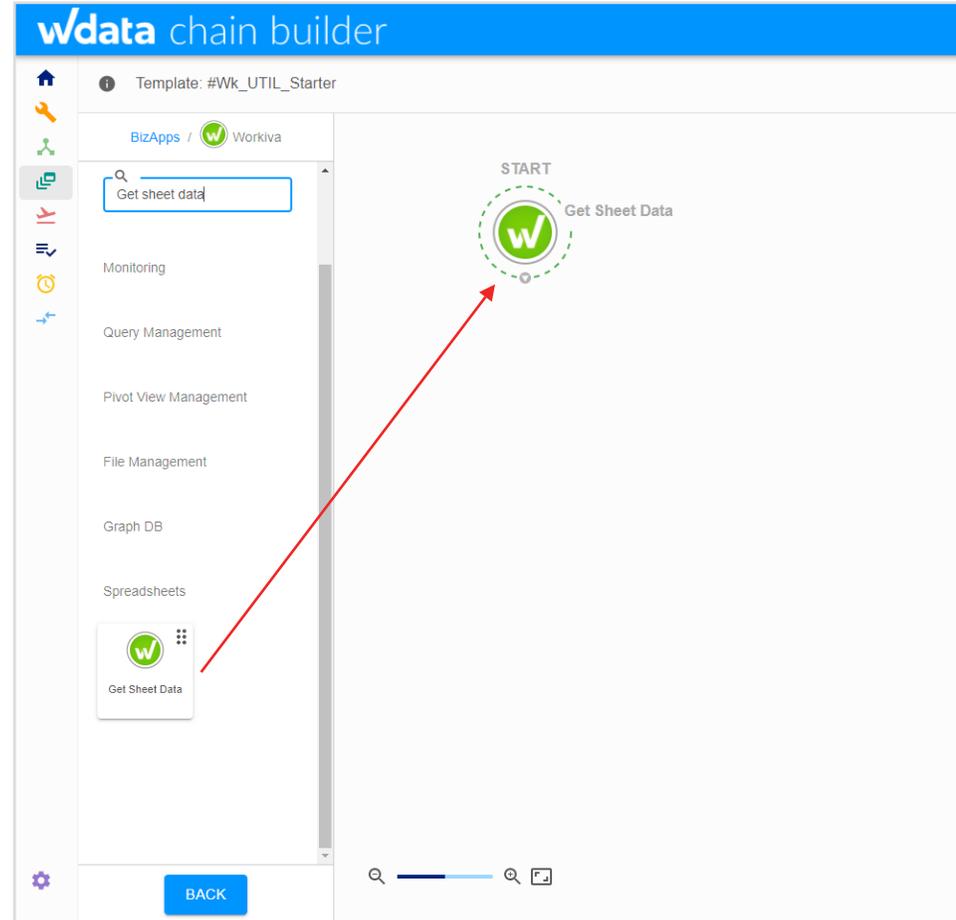
Edit the template settings by filling in the above information in the fields as indicated by the red arrows.

Step 4: Create the "Get Sheet Data" Node P1

The screenshot displays the 'wdata chain builder' interface. On the left, a search bar contains the text 'workiva'. Below it, the 'Available BizApps' section shows a 'Workiva' icon highlighted with a red box. An arrow points from this icon to a larger view on the right. This view shows a grid of Workiva nodes, with the 'Get Sheet Data' node highlighted by a red box. The main workspace on the left contains a 'START' node and a red warning message: 'Use the menu to begin dragging a node into the chain start'.

In the blank template, go to the BizApp search bar and search for "Workiva". Click into Workiva icon and find the node "Get Sheet Data".

Step 4.1: Create the "Get Sheet Data" Node P2



Drag the node from the BizApp panel to the "Start" circle and double click on the node to edit.

Step 4.2: Edit "Get Sheet Data" Node

The screenshot displays the 'wdata chain builder' interface. On the left, a sidebar shows a list of variables under the 'Template' section: 'Control_SSID', 'Control_SID', and 'Control_Region'. These three variables are highlighted with red boxes. Red arrows point from these boxes to the corresponding input fields in the main configuration area. The main area is titled 'Edit Get Sheet Data' and contains the following fields:

- Name:** Get Sheet Data
- Description (optional):** Reads the Control Sheet Data to Get the Chain Variables specific to the Chain
- Iterations:** (toggle is off)
- Command Properties:**
 - Spreadsheet ID:** Control_SSID (selected from dropdown)
 - Sheet ID/Name:** Control_SID (selected from dropdown)
 - Region:** Control_Region (selected from dropdown)
- Value Style:** Calculated
- Revision:** -1

At the top right of the configuration area, there are three buttons: 'DELETE', 'CANCEL', and 'SAVE'. The 'SAVE' button is highlighted with a red box.

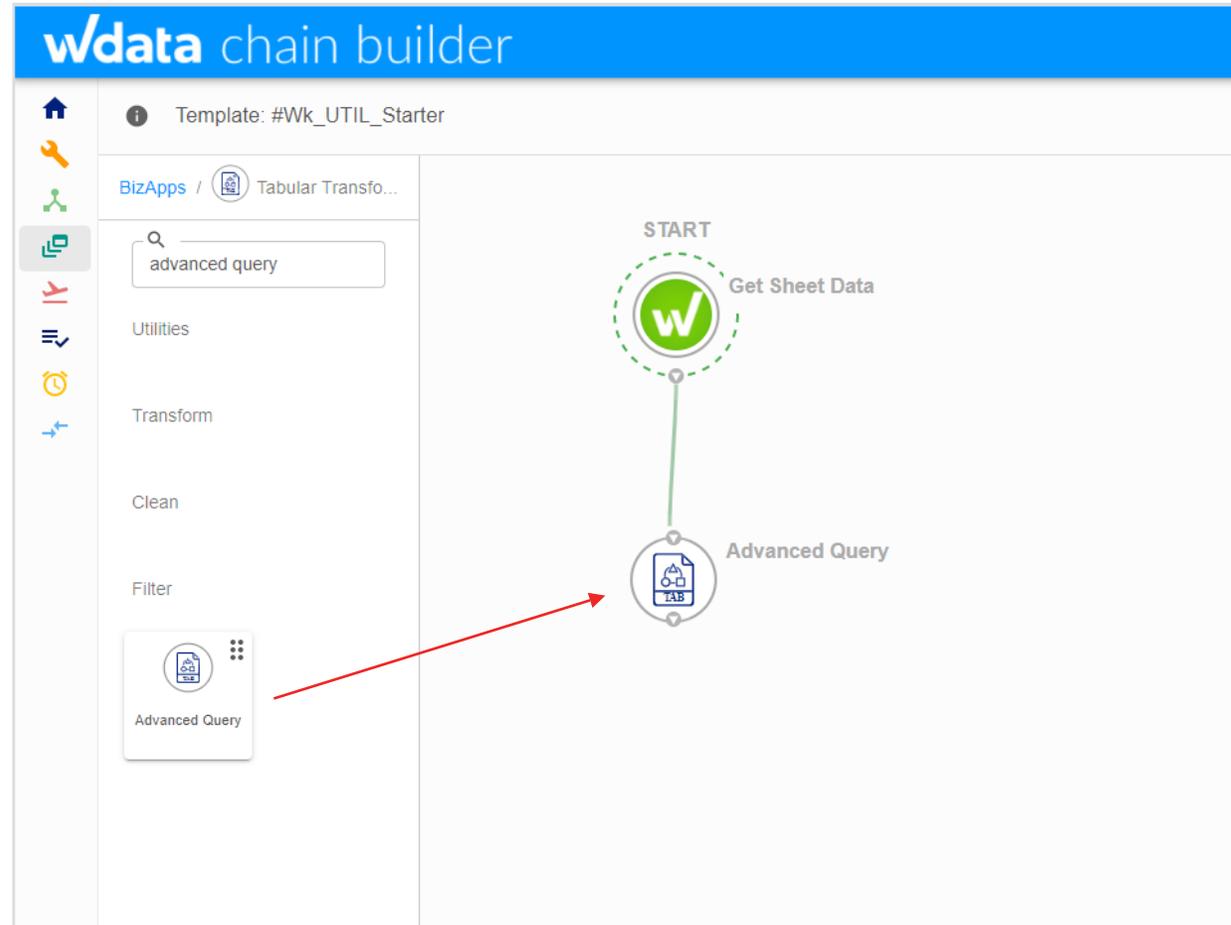
Input the node information as shown above. For the Spreadsheet ID, Sheet ID/Name and Region fields, go to the left panel and select the corresponding variables under 'Template'. Then, click the "Save" button on the upper right corner. The first node is now created.

Step 5: Create the "Advanced Query" Node P1

The screenshot displays the 'wdata chain builder' interface. At the top, a blue header contains the 'wdata chain builder' logo. Below the header, the main workspace shows a 'START' node labeled 'Get Sheet Data'. To the left, a sidebar contains a search bar with the text 'tabular transformation' and a list of available BizApps. The 'Tabular Transformation' node is highlighted with a red box. An arrow points from this node to the right, where a list of data processing nodes is displayed. The 'Advanced Query' node is highlighted with a red box in this list. Other nodes visible include 'Concat Files', 'Pivot', 'Unpivot', 'Clean', 'Clean Unquoted Newlines', 'Filter', 'Extract Value', and 'Smart Filter Rows'.

Create the "Advanced Query" node by searching "Tabular Transformation" > "Advanced Query" in the BizApp search bar.

Step 5.1: Create the "Advanced Query" Node P2



Drag the node from the BizApp panel to the blank space in the template. Then, connect the "Get Sheet Data" node to the "Advanced Query" node. Double click on the "Advanced Query" node to edit.

Step 5.2: Edit "Advanced Query" Node

Template: #WK_UTIL_Starter

PUBLISHED NEW CHAIN TEMPLATE SETTING

Select a variable

- Command
- Get Sheet Data
 - Command Details
 - Data**
 - Row Count
- Runtime
- Template

Edit Advanced Query
Tabular Transformation - Advanced Query

DELETE CANCEL SAVE

Basic Info

Name
Advanced Query

Description (optional)

Iterations

Command Properties

Tables
Add all of the files that will be used in the query, as well as their table name.

File	Table Name
Data	ControlsData

ADD REMOVE ALL

REMOVE

Query
Select * from ControlsData where RUN_CHAIN = 'Yes'

The SQL query to execute. INSERT, UPDATE, CREATE are not supported.

Input Delimiter
Comma

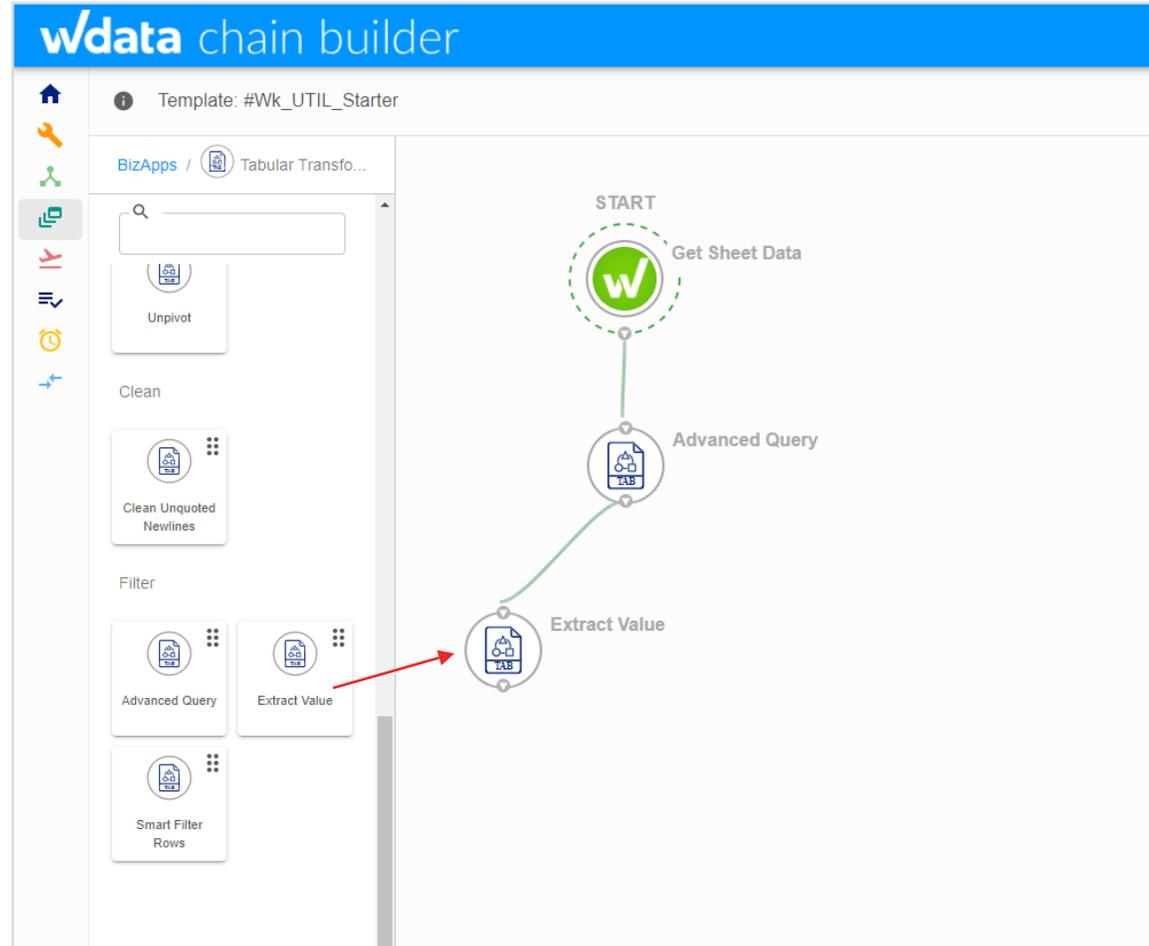
The delimiter of the input CSV file, as well as the join files.

Output Delimiter
Comma

The delimiter to use for the result of the query.

Input the node information as shown above. For the File field, go to the left panel and select the variable 'Data' under 'Get Sheet Data'. Then, click the "Save" button.

Step 6: Create the "Extract Value" Nodes



Create the "Extract Value" node by searching "Tabular Transformation" > "Extract Value" in the BizApp search bar. Remember to connect the "Advanced Query" node to the "Extract Value" node.

Step 6.1: Edit "Extract Value" Nodes

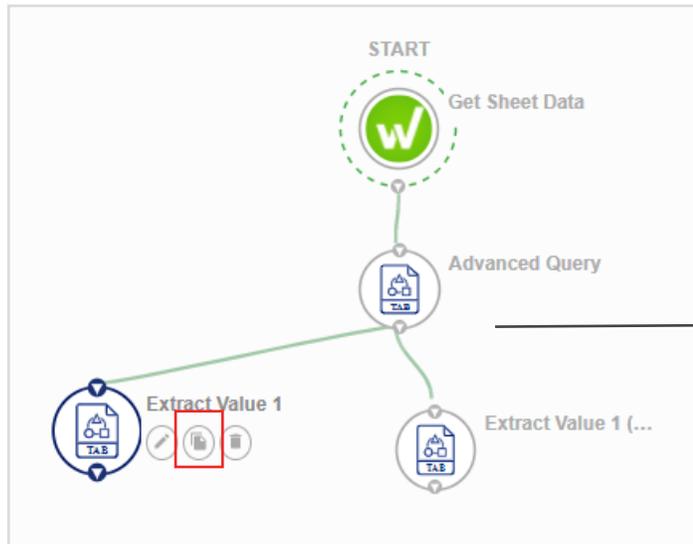
The screenshot displays the 'Edit Extract Value 1' configuration window. On the left, a sidebar lists variables: Command, Advanced Query, Command Details, Record Count, Result (highlighted with a red box and arrow), Get Sheet Data, Runtime, and Template. The main panel shows the configuration for 'Extract Value 1' with the following fields:

- Name: Extract Value 1
- Description (optional):
- Iterations: [Refresh] [Toggle] [Eye]
- Command Properties:
- Input file: Result
- The DSV file to transform:
- Column Index: 1
- The column to extract the value from (This value is based on the first line in the file being row 1). Leave this empty to extract the entire row.
- Delimiter: Comma
- The delimiter of the input DSV file.
- Row Index: 2
- The row to extract the value from (This value is based on the first line in the file being row 1)

Buttons for DELETE, CANCEL, and SAVE are located at the top right of the main panel.

Input the node information as shown above. For the Input File, go to the left panel and select the variable 'Result' under 'Advanced Query'. By doing this, we are taking the output of the previous node as the input file of this node. Then click "Save".

Step 6.2: Duplicate "Extract Value" Nodes



DELETED CANCEL SAVE

Basic Info

Name
Extract Value 2

Description (optional)

Iterations ↺ 👁

Command Properties

Input file
Result

The DSV file to transform

Column Index
2

The column to extract the value from (This value is based on the first line in the file being row 1). Leave this empty to extract the entire row.

Delimiter
Comma

The delimiter of the input DSV file.

Row Index
2

Next, duplicate the "Extract Value" node by clicking on the duplicate icon. A copy of the node will automatically pop out. Link the copy with the Advanced Query node and edit the information of the new node.

Note that each node extracts information from each column on the control sheet. As such, you would need to create 20 "Extract Value" nodes to extract information from 20 columns in the control sheet. Remember to name the nodes and fill in the Column Index following a numerical order. (i.e. name your second node "Extract Value 2". This node extracts the information in the second column of the control sheet, which is File Name)

Step 6.3: Create Chain Event Node

 Edit NULL
ChainEvent DELETE CANCEL SAVE

Basic Info

Name
NULL

Description (optional)
Just in Place to Save the time to keep connecting every single extract value node to the next step.

Conditions

AND + RULE + GROUP

String <> Is Blank 

The data type to test The operation to test

Create a "Conditional" node and link the node with all of the Extract Value node. Fill in the node information as shown above. This node is just in place to hold the connections from all of the Extract Value nodes.

Step 6.4: Complete the Template Chain

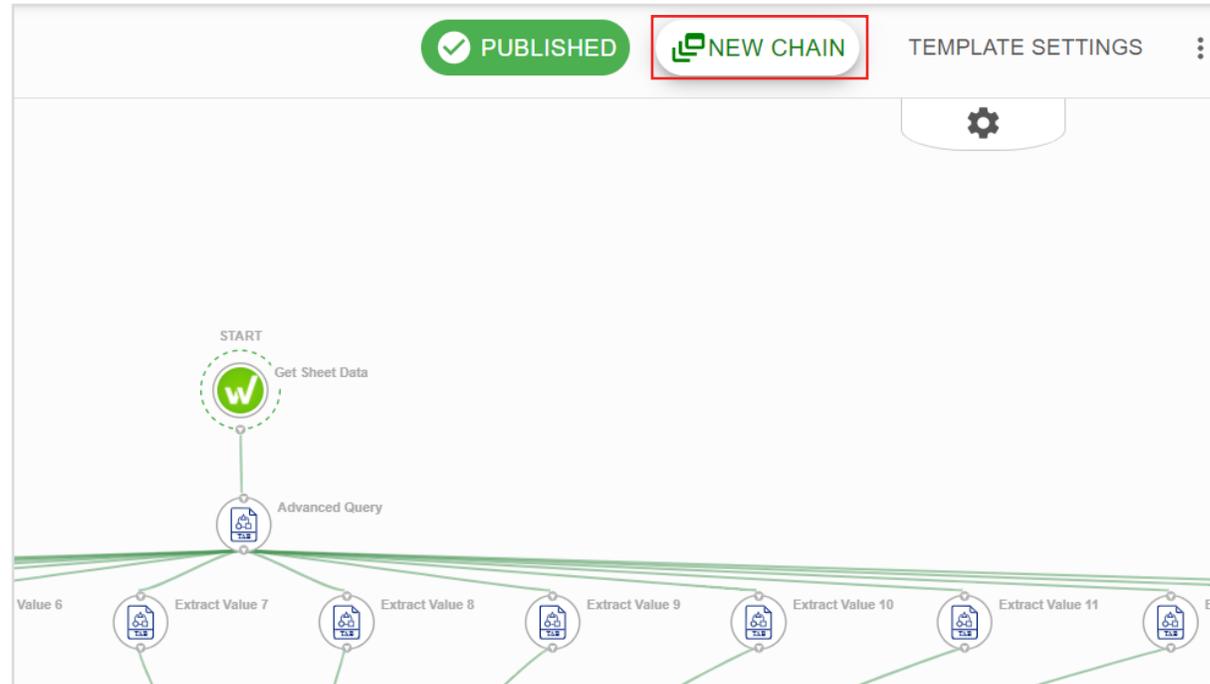


Your chain is now completed. Hit "Publish" on the upper right corner. (Tips: if the nodes are all over the place, the Layout Graph function could re-arrange the chain structure automatically)



Chain 4.1: External to Source

Step 1: Create New Chain from Template P1



In the template, click "New Chain" on the upper right corner. The External to Source Chain helps us import the data from the external systems to the source table.

Step 1.1: Create New Chain from Template P2

Template Wizard - #Wk_UTIL_Starter
Convert template to a chain

[VIEW TEMPLATE](#) [BACK TO SELECTION](#)

1 Select Environment ————— 2 Variable Mappings
Optional ————— 3 Connection Mappings
Optional

[BACK](#) [NEXT](#)

New Chain Name
#WK_TB_EXT_SRC

Select Workspace*
APAC SAs Utilities

Select Environment*
DEV

Name the new chain as "#WK_TB_EXT_SRC". Select the corresponding workspace and choose "DEV" as the environment. Hit Next.

Step 1.2: Create New Chain from Template P3

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE [BACK TO SELECTION](#)

1 Select Environment 2 Variable Mappings Optional 3 Connection Mappings Optional

[BACK](#) [NEXT](#)

Map template's variables to existing variables
Choose the type of variable to which each template variable will be mapped. If the 'Variable Type' field is disabled, your template is enforcing which type of variable you can choose

Control_SSID	→	Variable Type* Chain
	→	New Variable Name* Control_SSID
Control_SID	→	Variable Type* Chain
	→	New Variable Name* Control_SID
Control_Region	→	Variable Type* Chain
	→	New Variable Name* Control_Region

Choose "Chain" as the Variable Type and input the Variable Names again as shown above.

Step 1.3: Create New Chain from Template P4

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE BACK TO SELECTION

✓ Select Environment — Variable Mappings Optional — 3 Connection Mappings Optional

BACK SUBMIT

Connection Mappings

Workiva	→	 Workiva - APAC SA	x v	CloudRunner	x v
Tabular Transformation	→	 Tabular Transformation - APAC SA	x v	CloudRunner	x v

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE BACK TO SELECTION

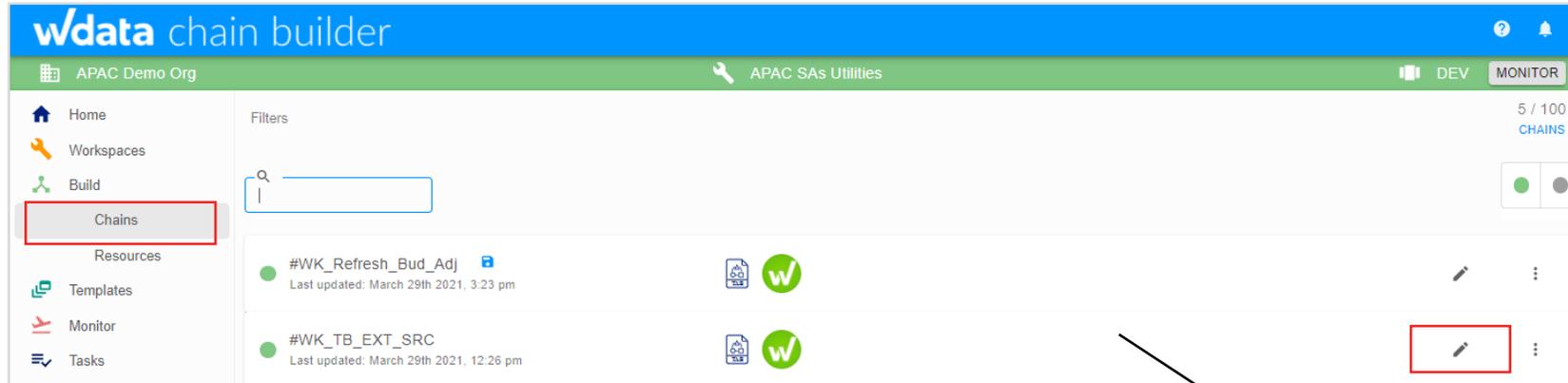
✓ Select Environment — Variable Mappings Optional — 3 Connection Mappings Optional



VIEW YOUR NEW CHAIN

Select the above connections as shown above. Click submit. The new chain is now set up.

Step 2: Edit Chain Settings P1



Go to "Build">"Chains" on the left panel and click into the edit icon. Then, choose "Chain Settings" on the upper right corner to edit the settings.

Step 2.1: Edit Chain Settings P2



Edit Chain CANCEL SAVE

Setup

Name

Description

Allow concurrent runs Disable execution

Schedules +

Variables +

Name	Value	Encrypt	Actions
<input type="text" value="Control_SSID"/>	<input type="text" value="bcb1b0b6afb4e0f950ece3c23465f3d"/>	<input type="checkbox"/>	× ⋮
<input type="text" value="Control_SID"/>	<input type="text" value="a96efbdc053540809870753ead198b76"/>	<input type="checkbox"/>	× ⋮
<input type="text" value="Control_Region"/>	<input type="text" value="A1:T"/>	<input type="checkbox"/>	× ⋮

Fill in the sheet ID and spreadsheet ID of the control sheet. To find those IDs, go to your control sheet to check the URL. The alphanumeric string after "/spreadsheet/" is the spreadsheet ID and the string after "/sheet/" is the sheet ID.

Step 3: Create Run Chain Node

The screenshot displays the 'Edit Run Chain' configuration for a 'ChainEvent' node. The interface is divided into a left sidebar and a main configuration panel. The sidebar lists various 'Extract Value' nodes, with 'Value' under 'Extract Value 2' and 'Value' under 'Extract Value 3' highlighted in red boxes. Red arrows point from these boxes to the 'Chain to Run' dropdown (set to '#WK_UTIL_Manage_Redundant_Files') and the 'r_FileName' and 'r_TableID' input fields (both set to 'Value'). The main panel shows the configuration for the 'Run Chain' node, including fields for Name, Description, Iterations, and Chain Runtime Inputs.

Create a "Run Chain" node and link the node with the Conditional node. Fill in the node information as shown above. For r_FileName, choose the variable "Value" under "Extract Value 2" and for r_TableID, choose the variable "Value" under "Extract Value 3". (the Extract Value 2 node extracts the FileName column and Extract Value 3 node extracts the TableID from the control sheet)

Step 4: Create Conditional Node - Edit Dec-12 File

The screenshot displays the Workiva interface for configuring a 'Conditional' node. The workspace is identified as '#WK_TB_EXT_SRC' with the environment 'APAC SAs Utilities Environment: DEV'. The node is titled 'Edit Dec-12 File' and is a 'ChainEvent'. The 'Basic Info' section shows the name 'Dec-12 File' and an optional description field. The 'Conditions' section is active, showing a rule configuration with the following elements:

- Logic: AND
- Buttons: + RULE, + GROUP
- Condition 1: String (The data type to test) compared to Value (The operation to test)
- Condition 2: Dec-12_2020.csv (The operation to test)

Red arrows indicate the selection of 'Value' from the 'Extract Value 2' list in the left sidebar and the '=' operator in the condition configuration.

Create a "Conditional" node and link the node with the Run Chain node. Fill in the node information as shown above. Please note that for value, choose the variable "Value" under "Extract Value 2". Note that we will later create other 2 conditional nodes to account for the situation in which we upload the November 2020 and December 2019 files.

Step 5: Create Create File Node

The screenshot displays the Workiva interface for editing a 'Create File' node. The left sidebar shows a list of nodes, with 'Value' nodes highlighted in red boxes. Red arrows point from these 'Value' nodes to the 'Table ID', 'File', and 'Name' fields in the main configuration panel. The 'Table ID' field is set to 'Value', the 'File' field is set to 'TB_12_2020.csv', and the 'Name' field is set to 'Value'. The 'Download URL' field is empty. The top right of the interface shows 'PUBLISHED', 'EXECUTE', and 'CHAIN SETTINGS' buttons.

Create a "Workiva">"Create File" node and link the node with the previous Conditional node. Fill in the node information as shown above.

Step 6: Create Import File into Table Node P1

The screenshot displays the Workiva interface for configuring a node. On the left, a sidebar lists variables under 'ColumnMappings', with 'Id' selected. The main workspace shows the 'Edit Import File into Table' node configuration. The 'Table ID' field is set to 'Value', and the 'File ID' field is set to 'Result'. A red box labeled 'Extract Value 3' points to the 'Value' dropdown in the 'Table ID' field.

Create a "Workiva">"Import File into Table" node and link the node with the previous Conditional node. Fill in the node information as shown above. For TableID, choose the variable "Value" under "Extract Value 3"

Step 6.1: Create Import File into Table Node P2

Column Mappings +

Run Asynchronously ?

Tags +

Key	Value	=	Value
	Extract Value 5 (TagKey1)		Extract Value 6 (TagVal1)
			🗑️
Key	Value	=	Value
	Extract Value 7 (TagKey2)		Extract Value 8 (TagVal2)
			🗑️

Add two tags in the bottom and fill in the tag information as shown above. Select variables "Value" under "Extract Value 5" and "Extract Value 7" as tag keys and "Extract Value 6" and "Extract Value 8" as tag values.

Step 7: Duplicate Conditional Node, Create File Node and Import File into Table Node

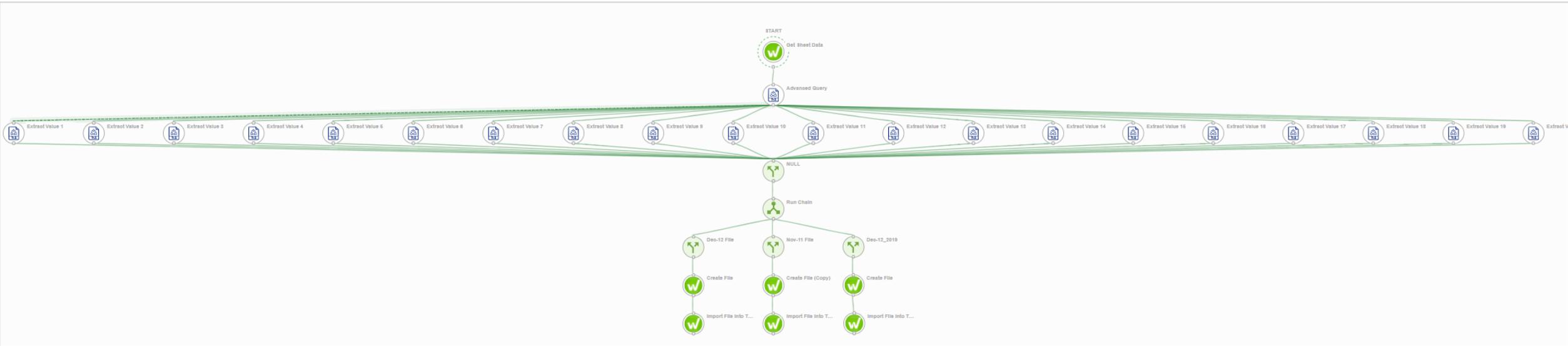
The image displays a workflow editor interface with three main components:

- Workflow Diagram:** A central 'Run Chain' node is connected to three parallel paths. Each path starts with a 'Conditional Node' (represented by a green circle with a 'Y' icon), followed by a 'Create File Node' (green circle with a 'W' icon), and ends with an 'Import File into Table Node' (green circle with a 'W' icon). The paths are labeled 'Dec-12 File', 'Nov-11 File', and 'Dec-12_2019'. The 'Nov-11 File' path is highlighted with a black box.
- 'Edit Nov-11 File' Panel:** This panel shows the configuration for a 'ChainEvent' node. Under the 'Conditions' section, a rule is defined: 'String' (data type) is equal to 'Nov-11_2020.csv' (operation). A red arrow points to the 'Nov-11_2020.csv' value.
- 'Edit Create File (Copy)' Panel:** This panel shows the configuration for a 'Workiva - Create File' node. Under the 'Command Properties' section, the 'File' field is set to 'TB_11_2020.csv'. A red arrow points to this field.

Below the 'Edit Create File (Copy)' panel, a list of resources is shown, with 'TB_11_2020.Csv' highlighted by a red box.

Duplicate two sets of Conditional Node, Create File Node and Import File into Table Node and connect the nodes as shown above. For each set, edit the csv file names in the first two nodes as indicated in red.

Step 8: Complete Chain 3.1 - External to Source

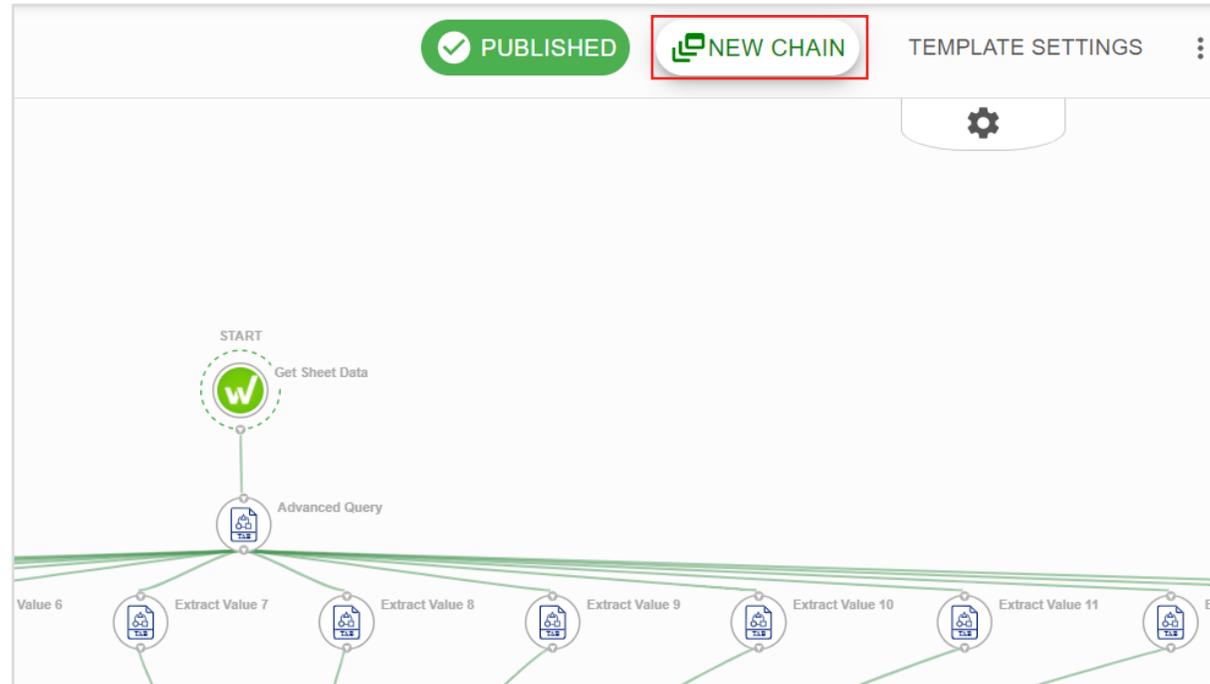


Double check the structure of the chain. Remember to hit Publish.



Chain 4.2: Source to Staging

Step 1: Create New Chain from Template P1



In the template, click "New Chain" on the upper right corner. The External to Source Chain helps us import the data from the external systems to the source table.

Step 1: Edit Chain Settings

#WK_TB_SRC_STG
Workspace: APAC SAs Utilities Environment: DEV

PUBLISHED EXECUTE CHAIN SETTINGS

Search

No variables are currently available

Edit Chain

CANCEL SAVE

Setup

Name: #WK_TB_SRC_STG

Description: Basic Control Sheet Reader

Allow concurrent runs Disable execution

Schedules

Variables

Name	Value	Encrypt	Actions
Control_Region	A1:T~	<input type="checkbox"/>	× ⋮
Control_SID	d81fdc1d15b43049b52b99533212a22~	<input type="checkbox"/>	× ⋮
Control_SSID	bcb1b0b6afb4e0f950ece3c23465f3d~	<input type="checkbox"/>	× ⋮

Dynamic Variables

Name	Initial Value	Actions
------	---------------	---------

Repeat step 1 (including sub-steps) and build a new chain "#WK_TB_SRC_STG" from the template. Make sure to fill in the correct Control Sheet ID and Control Spreadsheet ID.

Step 1.1: Create New Chain from Template P2

Template Wizard - #Wk_UTIL_Starter
Convert template to a chain

[VIEW TEMPLATE](#) [BACK TO SELECTION](#)

1 Select Environment ————— 2 Variable Mappings
Optional ————— 3 Connection Mappings
Optional

[BACK](#) [NEXT](#)

New Chain Name
#WK_TB_EXT_SRC

Select Workspace*
APAC SAs Utilities

Select Environment*
DEV

Name the new chain as "#WK_TB_EXT_SRC". Select the corresponding workspace and choose "DEV" as the environment. Hit Next.

Step 1.2: Create New Chain from Template P3

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE [BACK TO SELECTION](#)

1 Select Environment 2 Variable Mappings Optional 3 Connection Mappings Optional

[BACK](#) [NEXT](#)

Map template's variables to existing variables
Choose the type of variable to which each template variable will be mapped. If the 'Variable Type' field is disabled, your template is enforcing which type of variable you can choose

Control_SSID	→	Variable Type* Chain
	→	New Variable Name* Control_SSID
Control_SID	→	Variable Type* Chain
	→	New Variable Name* Control_SID
Control_Region	→	Variable Type* Chain
	→	New Variable Name* Control_Region

Choose "Chain" as the Variable Type and input the Variable Names again as shown above.

Step 1.3: Create New Chain from Template P4

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE BACK TO SELECTION

✓ Select Environment ————— ✓ Variable Mappings
Optional

3 Connection Mappings
Optional

BACK SUBMIT

Connection Mappings

Workiva	→	 Workiva - APAC SA	x v	CloudRunner	x v
Tabular Transformation	→	 Tabular Transformation - APAC SA	x v	CloudRunner	x v

Template Wizard - #WK_UTIL_Starter
Convert template to a chain

VIEW TEMPLATE BACK TO SELECTION

✓ Select Environment ————— ✓ Variable Mappings
Optional

3 Connection Mappings
Optional



VIEW YOUR NEW CHAIN

Select the above connections as shown above. Click submit. The new chain is now set up.

Step 2: Create Run Chain Node P1

The screenshot displays the 'Edit Run Chain' configuration for a 'ChainEvent' node. The interface is divided into a left sidebar and a main configuration panel. The sidebar contains a search bar and a list of 'Extract Value' items, with two 'Value' entries highlighted in red boxes. The main panel shows the 'Edit Run Chain' form with the following sections:

- Basic Info:** Name (Run Chain), Description (optional).
- Iterations:** Iterations (toggle off).
- Chain to Run:** A dropdown menu showing '#WK_UTIL_Manage_Redundant_Files'.
- Chain Runtime Inputs:** Two input fields: 'r_FileName' (Value) and 'r_TableID' (Value). The 'r_TableID' field has a tooltip that reads 'The TableID against which the FileName needs to be Validated'.

Red arrows indicate the mapping from the 'Value' entries in the sidebar to the 'Chain to Run' dropdown and the 'r_TableID' input field in the main panel.

Repeat step 3 in Chain 1.1 to create a "Run Chain" node and link the node with the conditional node.

Step 2.1: Create Run Chain Node P2

#WK_TB_SRC_STG
Workspace: APAC SAs Utilities Environment: DEV

PUBLISH EXECUTE CHAIN SETTINGS

Edit Run Chain
ChainEvent

DELETE CANCEL SAVE

Basic Info

Name
Run Chain

Description (optional)

Iterations

Chain to Run
#WK_UTIL_Run_Query_Upload_Files

Chain Runtime Inputs

r_FileName	Value	Extract Value 2
r_TableID	Value	Extract Value 3
r_QueryID	Value	Extract Value 4
r_TagKey1	Value	Extract Value 5
r_TagVal1	Value	Extract Value 6
r_TagKey2	Value	Extract Value 7
r_TagVal2	Value	Extract Value 8
r_TagKey3	Value	Extract Value 9
r_TagVal3	Value	Extract Value 10
r_TagKey4	Value	Extract Value 11
r_TagVal4	Value	Extract Value 12
r_ParamKey1	Value	Extract Value 13
r_ParamVal1	Value	Extract Value 14
r_ParamKey2	Value	Extract Value 15
r_ParamVal2	Value	Extract Value 16
r_ParamKey3	Value	Extract Value 17
r_ParamVal3	Value	Extract Value 18
r_ParamKey4	Value	Extract Value 19
r_ParamVal4	Value	Extract Value 20

Create another "Run Chain" node and link the node with the previous Run Chain node. Fill in the node information as shown above. Select the variable "Value" under the Extract Value node that corresponds to each field.

Step 3.1: Logging Time Stamp

The screenshot shows the 'Edit Create File' configuration window. The 'Text' field is set to 'Chain.ExecutionDateTime'. The 'Transformation' is set to 'Parse Date / Time'. The 'Output' is set to 'T'. The 'Value' field is configured with 'ISO Extended (Platform Standard)', 'On - %d/%m/%Y At- %l:%M %p', '(UTC) Dublin, Edinburgh, Lisbon, London', and 'Kuala Lumpur, Singapore'.

Input	Transformation	Output	Value
			ISO Extended (Platform Standard)
			On - %d/%m/%Y At- %l:%M %p
			(UTC) Dublin, Edinburgh, Lisbon, London
			Kuala Lumpur, Singapore

In the 'Text', select "Chain.ExecutionDateTime". Click on the "Chain.ExecutionDateTime", under 'Select Transformation', select "Parse Date / Time", and configure according to the screenshot above. Note, the input below 'ISO Extended (Platform Standard)' is, copy the following: On - %d/%m/%Y At- %l:%M %p

On - %d/%m/%Y At- %l:%M %p

Step 3.2: Write Sheet Data

The screenshot displays the Workiva configuration interface for the 'Write Sheet Data' command. On the left, a workflow diagram shows a sequence of steps: NULL, Run Chain, Run Chain, Create File, and Write Sheet Data. The 'Write Sheet Data' step is highlighted with a red box. The main configuration panel shows the command properties for 'Workiva - APAC SA2'. The 'Chain' section lists variables: Control_Region, Control_SID, and Control_SSID. The 'Command Details' section shows the 'Created File' variable selected. The 'Region' field is set to 'V2'. The 'Data File' field is set to 'Created file'.

In the 'Write Sheet Data', input the Spreadsheet, Sheet ID, Created File from the Chain and 'Create File' variables. In the Region, indicate "V2" as this is the cell where we would want the timestamp log to be inputted in the Master Control Sheet.

Step 4: Completed Chain 3.2 - Source to Staging



Double check the structure of the chain. Remember to hit Publish.



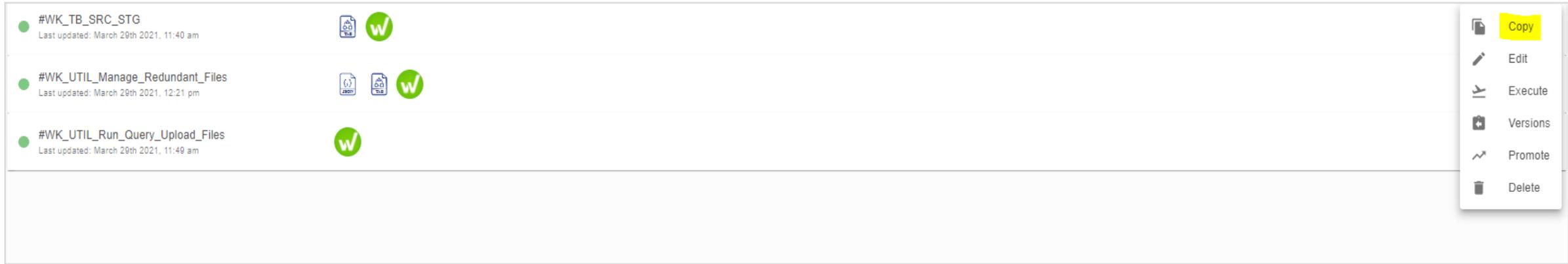
Chain 5: Budget & Adjustment Refresh

Step 1: Budget & Adjustment Refresh sheet

	A	B	C	D
1	Run_Chain	FileName	TableID	SheetID
2	Yes	Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	e91d388e5e5b4f74924eb3345829c657
3	Yes	Dec-12_2020.csv	244f85d061b44aeeb3d97eaecf688b29	857b6c33458944b5b81c4277bcac5baa
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

Create a 'Budget & Adjustment Refresh' control sheet, TableID indicated at the consumption table '#WK_CNS_TB_Sys'; SheetID indicated in 'Adjustment & Budget' sheets

Step 2: Duplicate the "#WK_TB_SRC_STG" chain



The screenshot shows a list of three workflow chains. The first chain, "#WK_TB_SRC_STG", is selected. A context menu is open over it, showing options: Copy, Edit, Execute, Versions, Promote, and Delete. The "Copy" option is highlighted in yellow.

Chain Name	Last updated	Icons
#WK_TB_SRC_STG	March 29th 2021, 11:40 am	File icon, W icon
#WK_UTIL_Manage_Redundant_Files	March 29th 2021, 12:21 pm	File icon, File icon, W icon
#WK_UTIL_Run_Query_Upload_Files	March 29th 2021, 11:49 am	W icon

Click on the 'three dots' on the right of the chain, click on 'Copy'. Rename the chain to '#WK_Refresh_Bud_Adj'

Step 3: Updating the 'Control_SID' Variable in Chain Settings

#WK_Refresh_Bud_Adj
Workspace: APAC SAs Utilities Environment: DEV

PUBLISHED EXECUTE CHAIN SETTINGS

EDIT CHAIN CANCEL SAVE

Setup

Name: #WK_Refresh_Bud_Adj
Description: Basic Control Sheet Reader

Allow concurrent runs Disable execution

Schedules +

Variables +

Name	Value	Encrypt	Actions
Control_SSID	bc1b0b6afb4e0f950ece3c23465f3d	<input type="checkbox"/>	× ⋮
Control_SID	b651737a90fd4ffea45a08c50fa356ba	<input type="checkbox"/>	× ⋮
Control_Region	A1.T	<input type="checkbox"/>	× ⋮

Dynamic Variables +

Name	Initial Value	Actions
------	---------------	---------

Navigate to 'Chain Settings', update the "Control_SID" to the control sheet that controls 'Budget' & 'Adjustment' spreadsheets.

Step 4: Updating Get Sheet Data



#WK_Refresh_Bud_Adj
Workspace: APAC SAs Utilities Environment: DEV

Select a variable

- Runtime
- Resources
- Chain
 - Control_Region
 - Control_SID
 - Control_SSID

Basic Info

Name: Get Sheet Data

Description (optional): Reads the Control Sheet Data to Get the Chain Variables specific to the Chain

Iterations

Command Properties

Workiva - APAC SA2 | CloudRunner

Spreadsheet ID: Control_SSID

The unique identifier of the spreadsheet

Sheet ID/Name: Control_SID

The unique identifier of the sheet

Region: Control_Region

[Start Column][Start Row][Stop Column][Stop Row] A1 style representation of a cell or range. A range may be unbounded in any/all directions by leaving off the corresponding column or row.

Value Style: Calculated

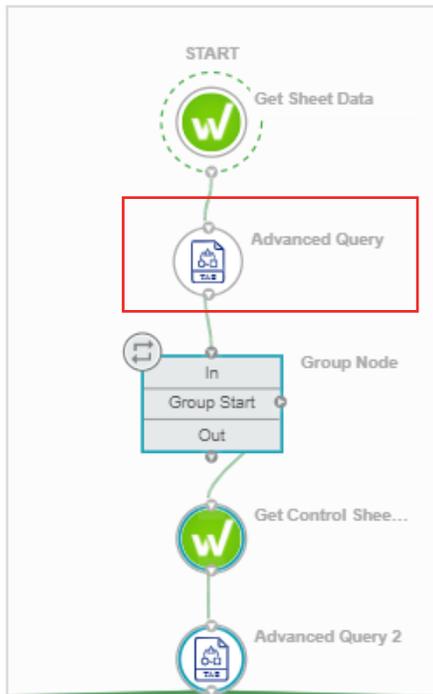
The style of cell value to return. For example, if a cell's value is =1+1 Raw value =1+1 or use Calculated to get the calculated value 2

Revision: -1

The revision of the sheet to use. (Use the value -1 to get the latest revision)

Naturally, the Spreadsheet ID, Sheet ID, Region would be inherited as the chain was copied, but check that they are correctly pointed to the variables defined in the 'Chain Settings' listed in the previous step

Step 5: Adding an 'Advanced Query' after the 'Get Sheet Data'



The screenshot shows the 'Edit Advanced Query' interface. The left sidebar has a tree view with 'Data' selected. The main panel shows the query configuration, including the query text: `SELECT FileName FROM ControlData WHERE Run_Chain = 'Yes'`. The interface also shows fields for Name, Description, Iterations, Command Properties, Tables, and Query.

In edit of 'Advanced Query', edit the query to " SELECT FileName FROM ControlData WHERE Run_Chain = 'Yes' "

Select FileName from ControlsData where RUN_CHAIN = 'Yes'

Step 5.1: Iteration: Adding Dynamic Output

Edit Advanced Query
Tabular Transformation - Advanced Query

DELETE CANCEL SAVE

Dynamic Outputs
Parse the text of this command's outputs to create new outputs for use later in your chain. TEST

Name: 1
Select a name

Original output: Result
Select the output to transform

Match text: *
Select expression to match

Match type: Regular Expression (regex)
Select the match type (regular expression or exact)

Match result
Select how to display regex result

Lines to check: 2-50
OPTIONAL: Lines to check. Separate with commas and specify ranges with a - (i.e. 1-10)

Output type: Multiple
Select whether you want to return a single output or list of outputs

Case sensitive Trim matches (no whitespace)

+ ADD DYNAMIC OUTPUT

Click on the 'Lightning' icon indicated in the red box to access the 'Dynamic Outputs'. Populate the values as illustrated. The objective of populating the values here is to create an Iterator, which would be used in the Command Group (next step).

Step 6: Adding the 'Command Group' node



The screenshot shows the 'Edit Node' configuration for a 'Command Group' node. The 'Iterations' field is set to 'I', and the 'Nodes' list includes 'Advanced Query 2' and several 'Extract Value' nodes. A red arrow points from the 'I' in the 'Iterations' field to the 'I' in the 'Command Details' section of the left sidebar.

First, search 'Command Group' and join it after the 'Advanced Query' node. Then, in edit mode, enable 'Iterations' and input the 'I' Dynamic Output value that was defined in previous step

Step 7: Editing the second 'Get Sheet Data' node

The screenshot displays the Workiva workflow editor interface. On the left, a workflow diagram shows a sequence of nodes: 'START', 'Get Sheet Data', 'Advanced Query', a 'Group Node' (containing 'In', 'Group Start', and 'Out'), 'Get Control Sheet Data' (highlighted with a red box), and 'Advanced Query 2'. A red arrow points from the 'Get Control Sheet Data' node in the diagram to the configuration panel on the right.

The configuration panel, titled 'Edit Get Control Sheet Data', shows the following settings:

- Command Properties:**
 - Workiva - APAC SA2
 - CloudRunner
- Spreadsheet ID:** Control_SSID
- Sheet ID/Name:** Control_SID
- Region:** Control_Region
- Value Style:** Calculated
- Revision:** -1

The second 'Get Sheet Data' node is joined to the start of the 'Command Group' node. With the iteration inherited from previous step, this node will read the fields in the Control Sheet and extract the values accordingly.

Step 8: Editing the second 'Advanced Query' node

The image displays the Workiva interface for editing a workflow. On the left, a workflow diagram shows a sequence of nodes: START, Get Sheet Data, Advanced Query, Group Node (containing In, Group Start, and Out), Get Control Shee..., and Advanced Query 2. A red box highlights the 'Advanced Query 2' node. An arrow points from this node to the main configuration window on the right.

The main configuration window is titled 'Edit Advanced Query 2' and shows the following settings:

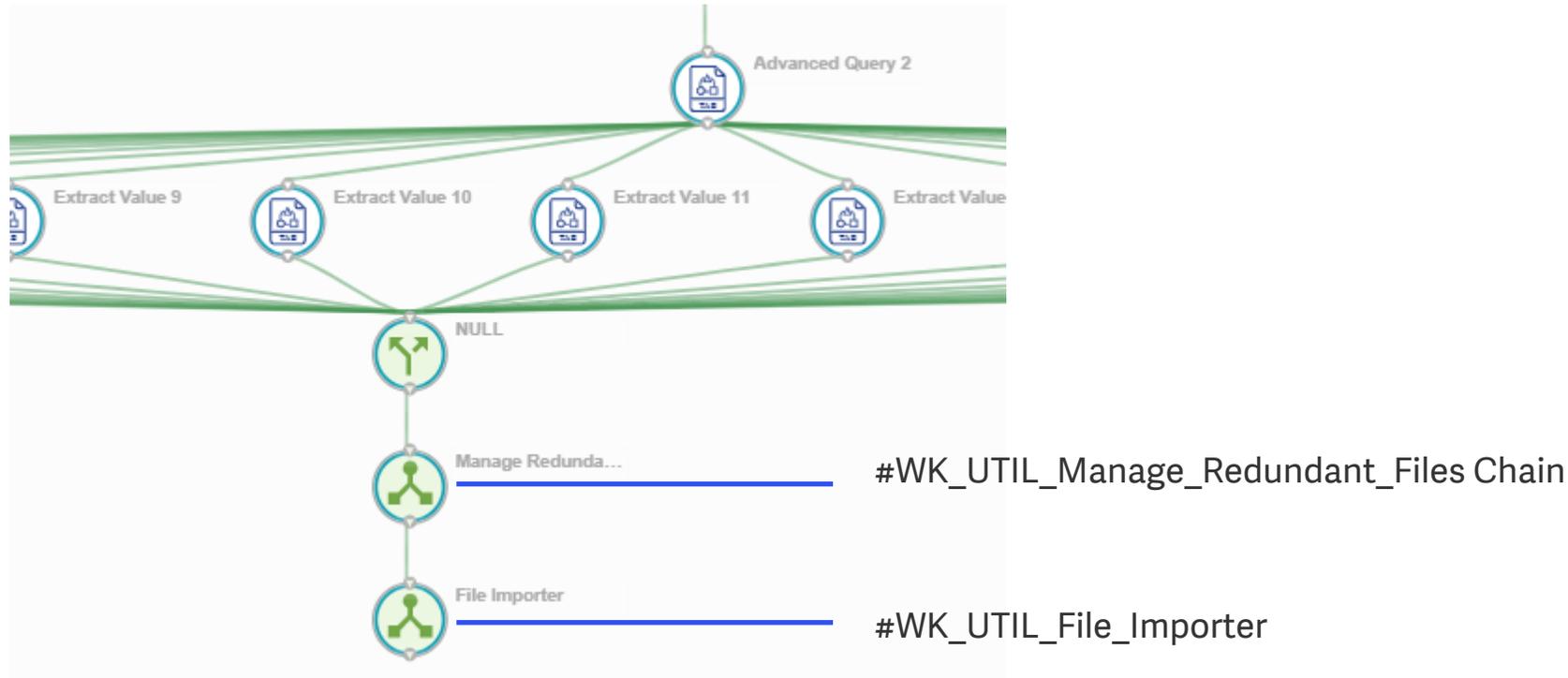
- Basic Info:** Name: Advanced Query 2
- Iterations:** Command Properties: Tabular Transformation - APAC SA, CloudRunner
- Tables:** ControlsData
- Query:** `Select * from ControlsData where Run_Chain = 'Yes' and Filename = '{Iteration}'`
- Input Delimiter:** Comma
- Output Delimiter:** Comma

Red boxes and arrows highlight the 'Data' variable in the 'Command Details' section and the 'Group Iterator' section, which are used to populate the query's filename parameter.

The second 'Advanced Query' node extracts the Data from the second 'Get Sheet Data' node (effectively the control sheet values via the iterators). Here we input the query to identify Run_Chain = 'Yes' & the inherited Filename(s) from the Iterator.

Select* from ControlsData where Run_Chain = 'Yes' and Filename =

Step 9: Second Part of the Chain



At the bottom part of the the chain, we would be running two UTIL chains -
'#WK_UTIL_Manage_Redundant_Files' & '#WK_UTIL_File_Importer'

Step 9.1: Configuring the Runtime Inputs for File Importer

The screenshot shows the 'Edit File Importer' configuration page in the Workiva interface. The page is titled 'Edit File Importer ChainEvent' and includes a 'PUBLISHED' status indicator and 'EXECUTE' and 'CHAIN SETTINGS' buttons. The main configuration area is divided into several sections:

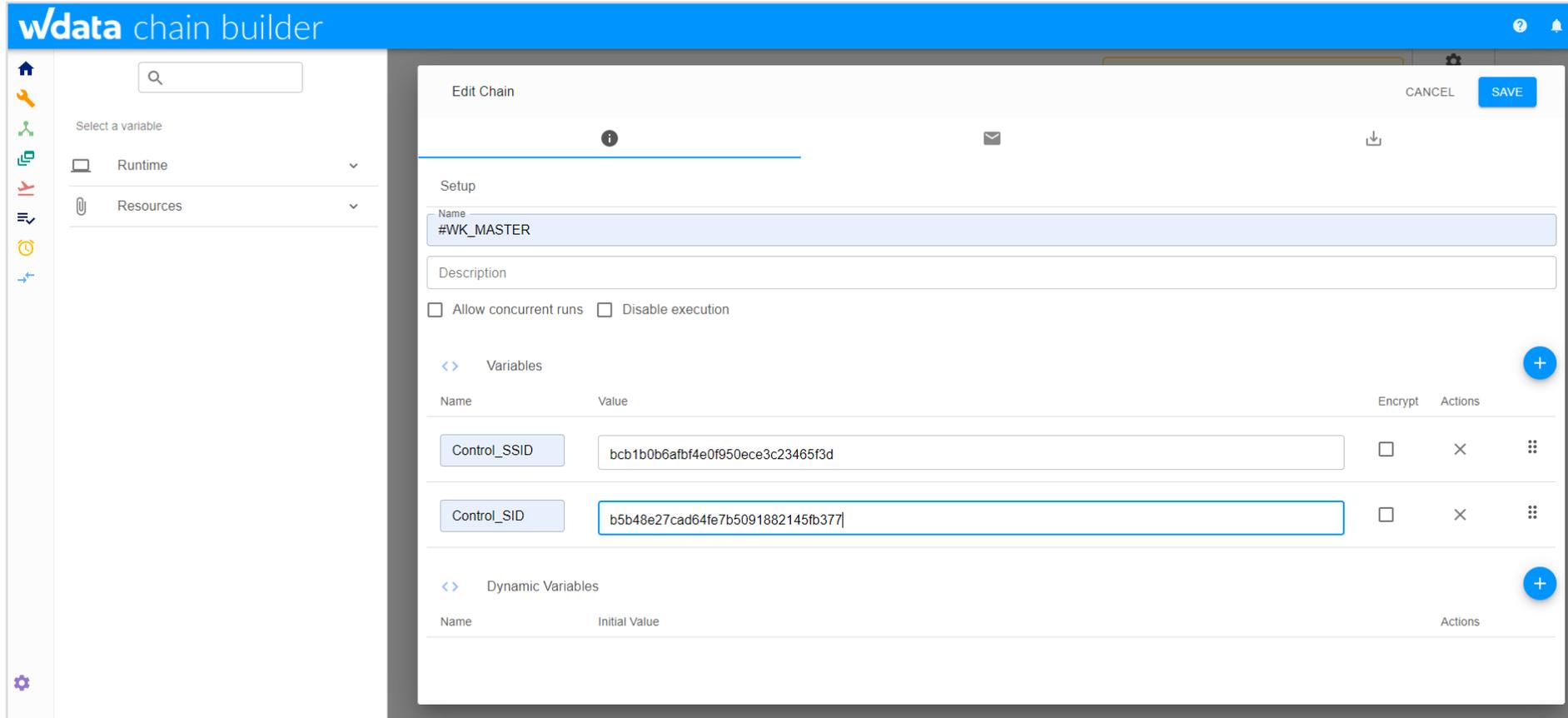
- Basic Info:** Includes fields for 'Name' (File Importer) and 'Description (optional)'.
- Iterations:** A section with a refresh icon and a toggle switch.
- Chain to Run:** A dropdown menu showing '#WK_UTIL_File_Importer'.
- Chain Runtime Inputs:** A section with five input fields, each with a 'Value' label and a dropdown menu:
 - r_FileName: Extract Value 2
 - r_TableID: Extract Value 3
 - r_SSID: Extract Value 22
 - r_SID: Extract Value 4
 - r_Region: Extract Value 21

A left-hand sidebar contains a search bar and a list of items including 'Extract Value 2' through 'Extract Value 9', 'Get Control Sheet Data', 'Get Sheet Data', 'Runtime', 'Resources', 'Chain', and 'Group Iterator'.

Input the Runtime input values, that is required to run the UTIL Chain - '#WL_UTIL_File_Importer'; mainly the "FileName, TableID, SSID, SID, Region"

Chain 6: Master Chain

Step 1: Create New Chain



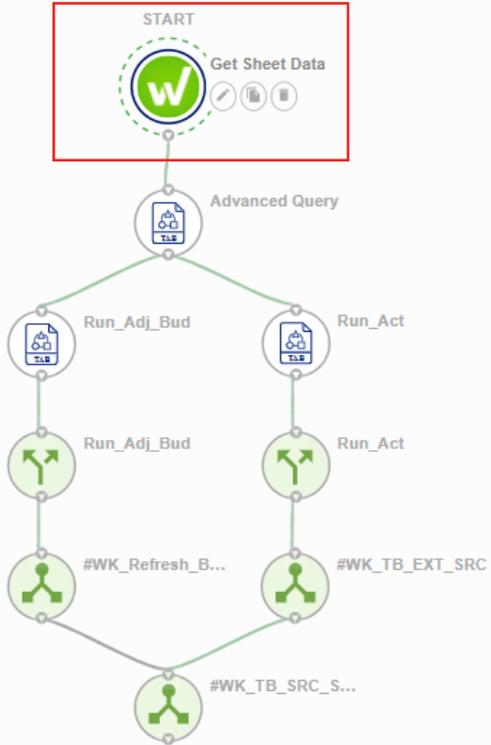
The screenshot shows the 'wdata chain builder' interface. The main window is titled 'Edit Chain' and has a 'SAVE' button in the top right corner. The 'Setup' section includes a 'Name' field with the value '#WK_MASTER' and an empty 'Description' field. Below this are two checkboxes: 'Allow concurrent runs' and 'Disable execution', both of which are unchecked. The 'Variables' section is expanded, showing a table with two rows of variables. The first row is 'Control_SSID' with the value 'bcb1b0b6afb4e0f950ece3c23465f3d'. The second row is 'Control_SID' with the value 'b5b48e27cad64fe7b5091882145fb377'. Each row has an 'Encrypt' checkbox (unchecked) and an 'Actions' column with a close icon and a menu icon. The 'Dynamic Variables' section is also expanded but currently empty.

Name	Value	Encrypt	Actions
Control_SSID	bcb1b0b6afb4e0f950ece3c23465f3d	<input type="checkbox"/>	× ⋮
Control_SID	b5b48e27cad64fe7b5091882145fb377	<input type="checkbox"/>	× ⋮

Create a New Chain (not from template). Name is "#WK_MASTER", and input the following variables:

- i. Control_SSID - Master Control's spreadsheet ID
- ii. Control_SID - Master Control's sheet ID

Step 2: Get Sheet Data



#WK_MASTER
Workspace: APAC SAs Utilities Environment: DEV

PUBLISHED EXECUTE CHAIN SETTINGS

Select a variable

- Runtime
- Resources
- Chain

Control_SSID
Control_SID

Edit Get Sheet Data

Workiva - Get Sheet Data

DELETE CANCEL SAVE

Get Sheet Data

Description (optional)

Iterations

Command Properties

- Workiva - APAC SA2 CloudRunner

Spreadsheet ID
Control_SSID
The unique identifier of the spreadsheet

Sheet ID/Name
Control_SID
The unique identifier of the sheet

Region
A1:F2
[Start Column][Start Row]:[Stop Column]:[Stop Row] A1 style representation of a cell or range. A range may be unbounded in any/all directions by leaving off the corresponding column or row.

Value Style
Calculated
The style of cell value to return. For example, if a cell's value is =1+1 Raw value =1+1 or use Calculated to get the calculated value 2

Revision
-1
The revision of the sheet to use. (Use the value -1 to get the latest revision)

Start with a 'Get Sheet Data', input the Spreadsheet & Sheet ID from the Chain Variables

Step 3: Advanced Query

The screenshot displays the 'Edit Advanced Query' configuration window. The 'Command Properties' section shows the command is 'Tabular Transformation - APAC SA' running on 'CloudRunner'. Under the 'Tables' section, a 'Data' file is added, and its table name is set to 'Master'. The 'Query' field contains the SQL query: `SELECT * from Master`. The 'Input Delimiter' and 'Output Delimiter' are both set to 'Comma'. The 'Preview results' checkbox is checked.

In 'Advance Query', input the "Data" file from the previous 'Get Sheet Data' node. Under 'Query', input: **"SELECT * from Master"**

Step 4: Extract Value Run_Adj_Bud

The image displays the Workiva interface for configuring a workflow. On the left, a workflow diagram shows a sequence of nodes: START, Get Sheet Data, Advanced Query, Run_Adj_Bud, Run_Act, and #WK_TB_SRC_S... The 'Run_Adj_Bud' node is highlighted with a red box. In the center, a 'Select a variable' menu is open, with 'Result' selected and highlighted by a red box. On the right, the 'Edit Run_Adj_Bud' configuration window is shown, with a red arrow pointing from the 'Result' variable to the 'Input file' field. The configuration includes the following fields:

- Name: Run_Adj_Bud
- Description (optional):
- Iterations: [Toggle]
- Command Properties:
 - Tabular Transformation - APAC SA
 - CloudRunner
- Input file: Result
- The DSV file to transform:
- Column Index: 6
- The column to extract the value from (This value is based on the first line in the file being row 1). Leave this empty to extract the entire row.
- Delimiter: Comma
- The delimiter of the input DSV file.
- Row Index: 2
- The row to extract the value from (This value is based on the first line in the file being row 1)

In 'Extract Value', input the "Result" file from the previous 'Advance Query' node. Input "6" / "2" under 'Column Index' / 'Row Index'

Step 5: Extract Value Run_Act

The screenshot displays the Workiva interface for editing a workflow. On the left, a workflow diagram shows a sequence of nodes: START, Get Sheet Data, Advanced Query, Run_Adj_Bud, Run_Act, and #WK_TB_SRC_S... The 'Run_Act' node is highlighted with a red box. In the center, a search bar and a list of variables are shown, with 'Result' selected and highlighted in a red box. On the right, the 'Edit Run_Act' configuration panel is open, showing the 'Input file' field set to 'Result', 'Column Index' set to '5', and 'Row Index' set to '2'. The 'Delimitter' is set to 'Comma'. The 'Command Properties' section shows 'Tabular Transformation - APAC SA' and 'CloudRunner'.

Workspace: APAC SAs Utilities Environment: DEV

SEARCH

Select a variable

- Command
- Advanced Query
- Command Details
- Record Count
- Result**
- Get Sheet Data
- Runtime
- Resources
- Chain

DELETED CANCEL SAVE

Basic Info

Name
Run_Act

Description (optional)

Iterations

Command Properties

Tabular Transformation - APAC SA CloudRunner

Input file
Result

The DSV file to transform

Column Index
5

The column to extract the value from (This value is based on the first line in the file being row 1). Leave this empty to extract the entire row.

Delimitter
Comma

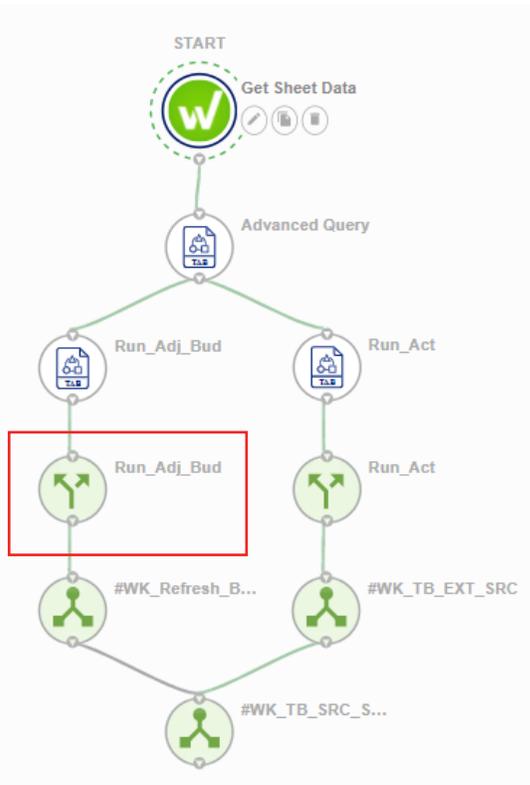
The delimiter of the input DSV file.

Row Index
2

The row to extract the value from (This value is based on the first line in the file being row 1)

In 'Extract Value', input the "Result" file from the previous 'Advance Query' node. Input "5" / "2" under 'Column Index' / 'Row Index'

Step 6: Conditional Run_Adj_Bud



#WK_MASTER
Workspace: APAC SAs Utilities Environment: DEV

PUBLISHED EXECUTE CHAIN SETTINGS

Search

Select a variable

- Command
- Advanced Query
- Get Sheet Data
- Run_Adj_Bud
- Command Details
- Row
- Value
- Runtime
- Resources
- Chain

Edit Run_Adj_Bud
ChainEvent

DELETE CANCEL SAVE

Basic Info

Name
Run_Adj_Bud

Description (optional)

Conditions

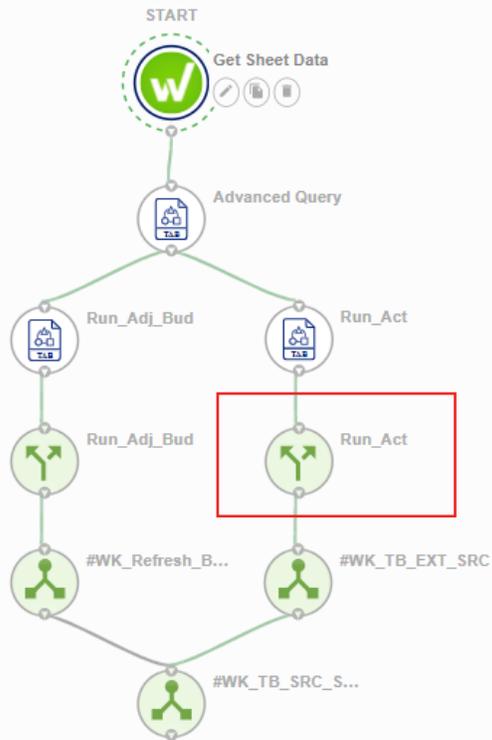
AND + RULE + GROUP

String = Yes

The data type to test The operation to test

In 'Conditions', input the "Value" from the previous 'Extract Value' node. Modify the Condition to "=" to Yes (value in the Run_Adj_Bud columns in the Master Control Sheet).

Step 7: Conditional Run_Act



Search

Select a variable

- ⚡ Command
- Advanced Query
- Get Sheet Data
- Run_Act
- <> Command Details
- <> Row
- T Value**
- Runtime
- Resources
- Chain

Edit Run_Act
ChainEvent

DELETE CANCEL SAVE

Basic Info

Name
Run_Act

Description (optional)

Conditions

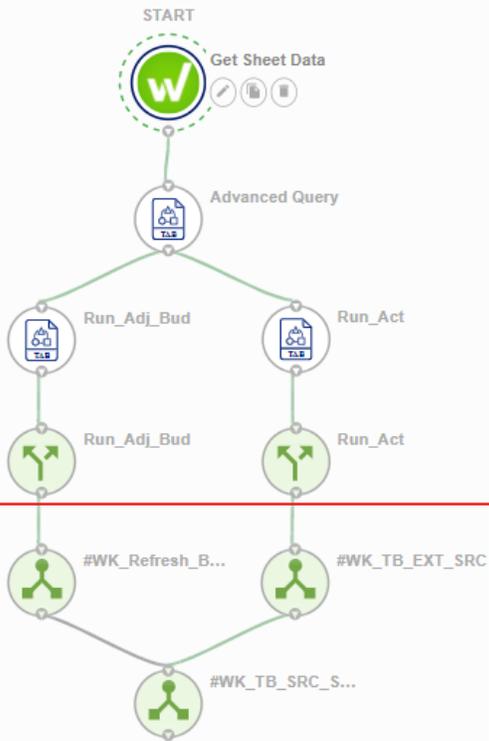
AND + RULE + GROUP

String Value = Yes

The data type to test The operation to test

In 'Conditions', input the "Value" from the previous 'Extract Value' node. Modify the Condition to "=" to Yes (value in the Run_Act columns in the Master Control Sheet).

Step 8: Run Chains



No variables are currently available

Search

Edit #WK_TB_EXT_SRC
ChainEvent

DELETE CANCEL SAVE

Basic Info

Name
#WK_TB_EXT_SRC

Description (optional)

Iterations

Chain to Run

#WK_TB_EXT_SRC|

#WK_Refresh_Bud_Adj

#WK_TB_EXT_SRC

#WK_TB_SRC_STG

#WK_UTIL_Manage_Redundant_Files

#WK_UTIL_Run_Query_Upload_Files

Place the listed chain in the Run Chain nodes following the order of the image to the left of the slide

- i. #WK_TB_EXT_SRC
- ii. #WK_Refresh_Bud_Adj
- iii. #WK_TB_SRC_STG

Step 9: Schedule Timer

#WK_MASTER
Workspace: APAC SAs Utilities Environment: DEV

PUBLISHED EXECUTE CHAIN SETTINGS

Search

No variables are currently available

Edit Chain

Setup

Name: #WK_MASTER

Description:

Allow concurrent runs Disable execution

Schedules

Color: [Purple Circle]

Every 1 day(s) at 2:18 am +08 (2:18 am +08)

Variables

Name	Value	Encrypt	Actions
Control_SSID	bcb1b0b6afb4e0f950ece3c23465f3d	<input type="checkbox"/>	✕ ⋮

Schedule your chain

Kuala Lumpur, Singapore

Runs

2:18 AM

Frequency

Minutes Daily Weekly Monthly

Recurring every: 1 Day(s)

Date Range

04/06/2021 · End date

CANCEL APPLY

Access 'Chain Settings', add a 'Schedule', set it to run at 0200 Daily. This schedule job would run the Master Chain Daily at 2am.

Step 9.1: Schedule for Timer Batch Job

The screenshot displays the 'Schedules' section of the Workiva interface. The top navigation bar shows 'APAC Demo Org' and 'APAC SAs Utilities'. The left sidebar contains navigation options: Home, Workspaces, Build, Templates, Monitor, Tasks, Schedules (highlighted), and Connections. The main content area is titled 'Filter by days' and includes a sub-header 'This includes all chains scheduled with hourly frequency or above'. A dropdown menu on the right is set to 'Next 6 days'. Below the filter, there are 'SHOW ALL' and 'HIDE ALL' buttons. The main visualization is a horizontal timeline with six vertical grid lines representing days. A purple dot is positioned on each day line, indicating a daily run. A legend above the timeline shows a purple square labeled '#WK_MASTER'.

In the 'Schedules' you would be able to see the schedules of the timer batch jobs that would run daily.

Step 10: Master Controls Sheet

C2 X ✓ f_x =IF(B2="Manual";"https://h.demo.wdesk.com/s/wdata/oc/app/apac-demo-org/workspace/654/environment/943/studio/chain/29926";"No link, running Auto")

	A	B	C	D	E	F
1	Chain	Run_Mode	Manual_Chain_Link	Last_Run	Run_Act	Run_Bud_Adj
2	#WK_MASTER	Auto	No link, running Auto	On - 11/05/2021 At- 10:54 AM	No	Yes
3						

Create a 'Master' control sheet, with the following columns. Set 'Run_Mode' to "Auto" and input the formula in Cell C2: "=IF(B2="Manual", "{URL of the Master Chain}", "No link, running Auto")"

X ✓ f_x ='TB SRC_STG'!V2

	A	B	C	D	E	F	G
	Chain	Run_Mode	Manual_Chain_Link	Last_Run	Run_Act	Run_Bud_Adj	
	#WK_MASTER	Auto	No link, running Auto	On - 11/05/2021 At- 10:54 AM	No	Yes	

In Cell D2, link it to Cell V2 from the 'SRC_STG' Controls Sheet.