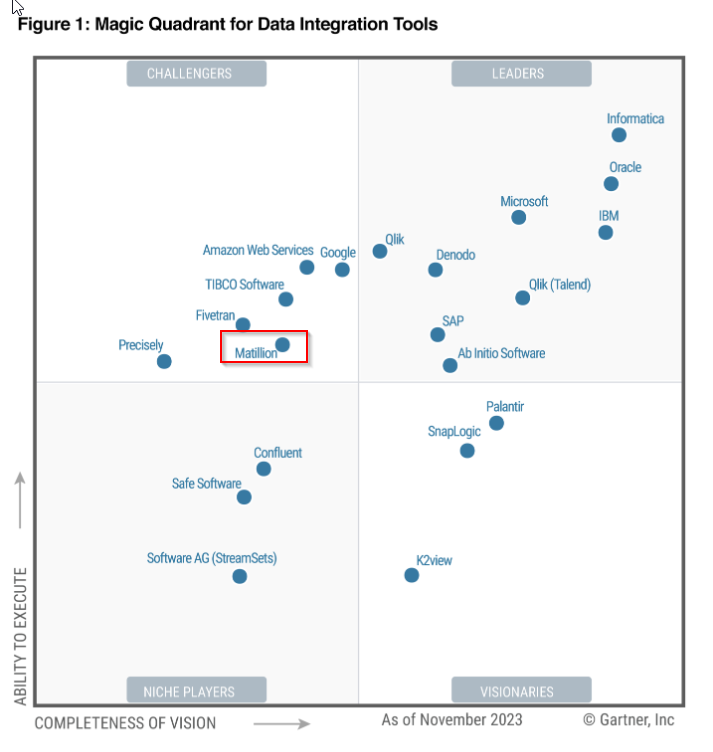
**COMPLETE END TO END PROJECT OF MATILLION INVOLVING AWS S3 & SNS, SNOWFLAKE, POWER BI AND GIT.**

# What is matillion?

Matillion ETL is a **cloud-native ETL solution** that simplifies the process of collecting, preparing, and loading data. It’s a drag and drop interface which helps build scalable data pipelines without the need for extensive coding.

DID YOU KNOW: Matillion named a **Challenger** in the **2023 Gartner®Magic Quadrant™ report** for Data Integration Tools.



# What is AWS S3?

Simply say, it’s a cloud based storage service.

# What is snowflake?

Like matillion, snowflake is built for the cloud. It’s a cloud-hosted, relational database that enables you to build data warehouses on demand

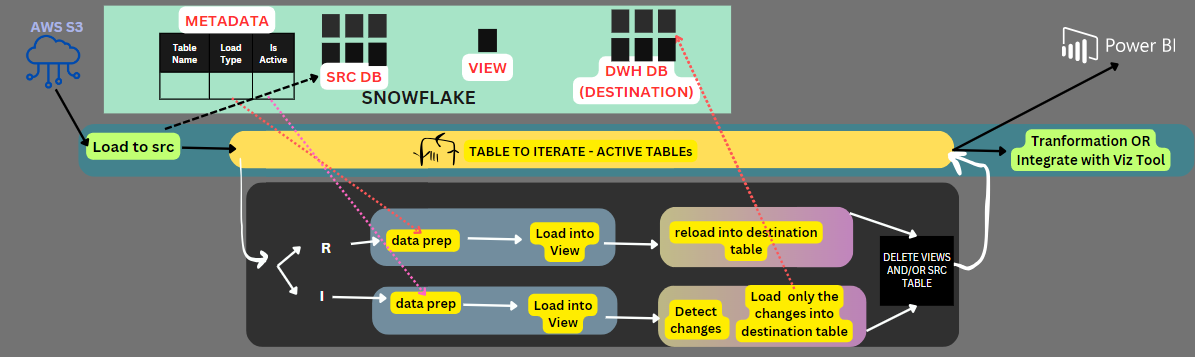
# What is always- on tables?

The data in table is available always.

Why is it a best practice?

* + Reload can take a long time for huge tables
  + When reloading, only partial data will be available to integrated tools, hence incorrect results

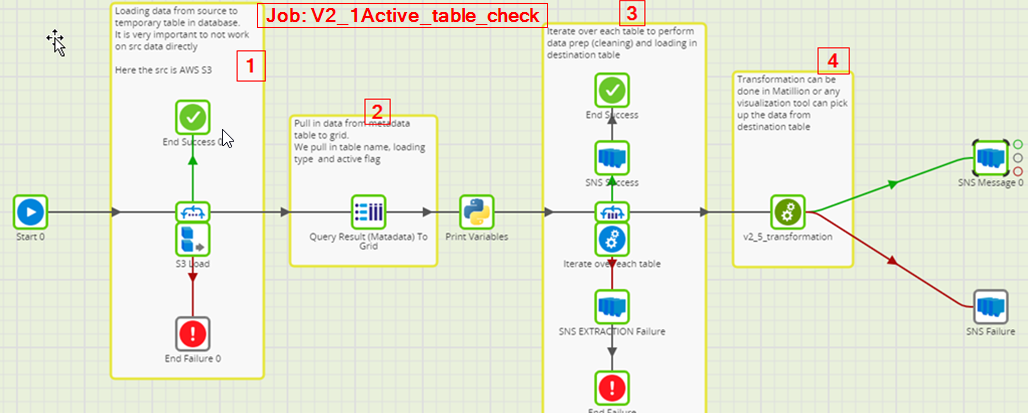
# Architecture of today’s project:



# Prep:

* + - An active AWS and Snowflake account
    - Create an matillion account (ETL setup) – refer doc: Steps To Configure Matillion.docx

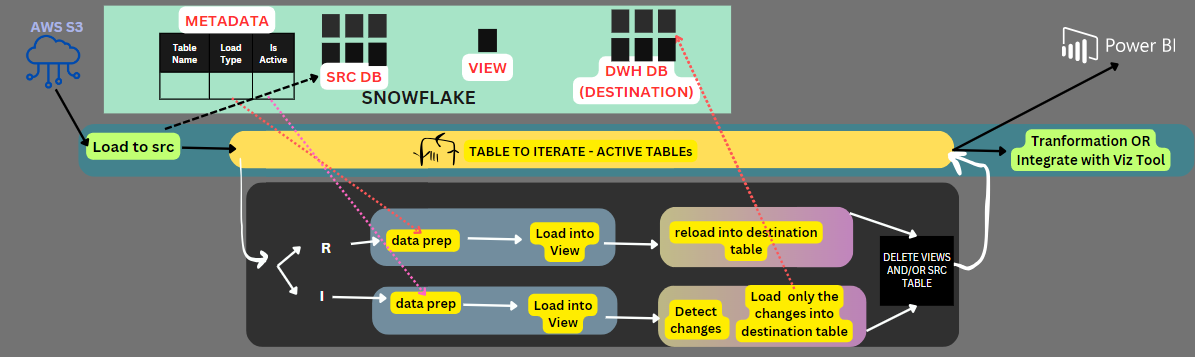
# Gist of Working:

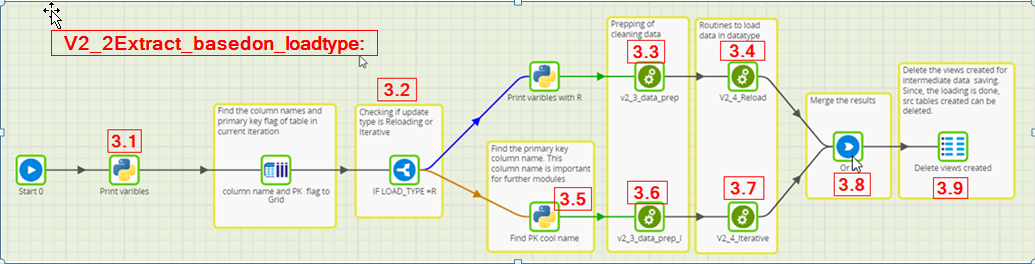


### Load the data from S3 to Snowflake src table [S3 LOAD]

### Pull in info from metadata table into the GRID Variable [QUERY RESULT TO GRID]

### Iterate all the next steps for each table [FIXED ITEARATOR + ORCHESTRATION JOB] + Notification [SNS]

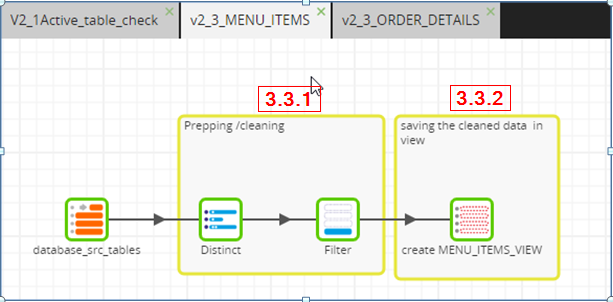




#### Retrieve the column name and primary key of the table [table metadata to grid]

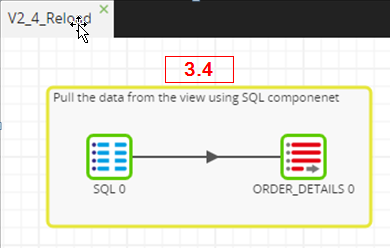
#### Check if the table needs to be iterated or reloaded [IF component]

#### For R: Prepare /clean the data [Run transformation component]



#### Cleaning [DISTINCT + FILTER NULLS]

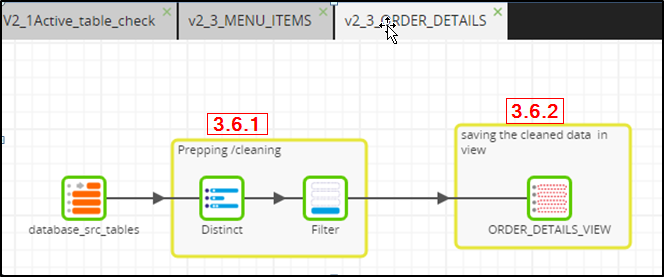
#### Save it in a view [CREATE VIEW]



#### For R : Load the data from the VIEW into the destination database [RUN transformation component]

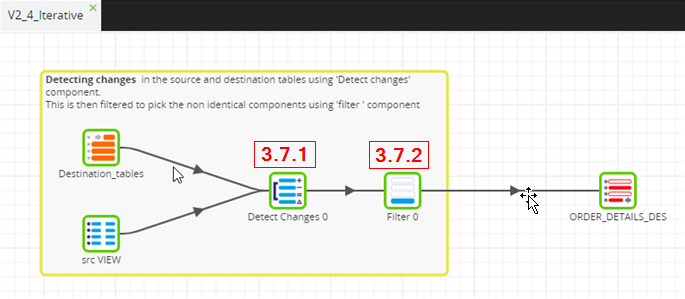
#### For I: Identify the name of the primary key using the flag [Python script]

#### For I: Prepare /clean the data [Run transformation component]



#### Cleaning [DISTINCT + FILTER NULLS]

#### Save it in a view [CREATE VIEW]



#### For I: Load the data from the VIEW into the destination database [RUN transformation component]

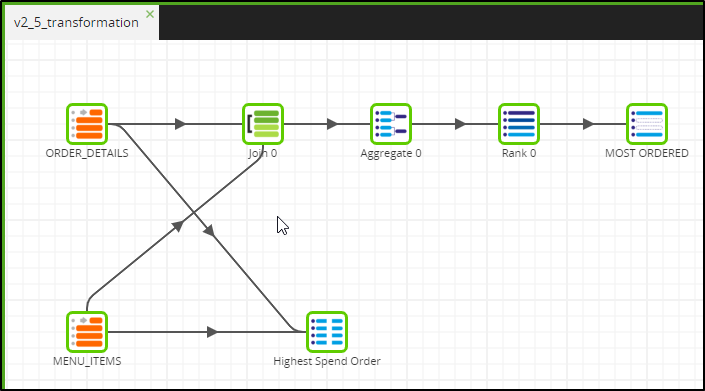
#### Detect changes [Detect changes]

#### Filter rows that need to be Upserted [filter]

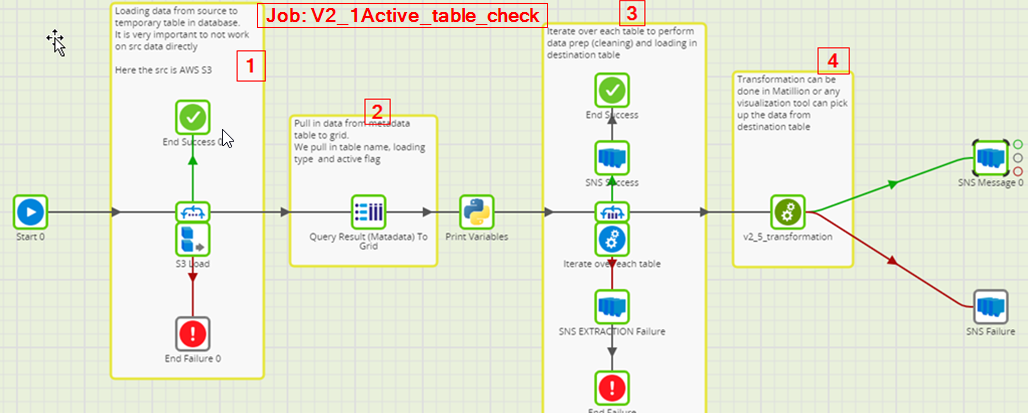
#### Irrespective of R or I , the pipeline needs to be merged [OR]

#### Since the cleaning and loading is completed, Unnecessary data/ structure is removed [SQL SCRIPT]

### You can perform transformations to retrieve value /KPI OR use the data as input to any visualization tool. [RUN transformation component]



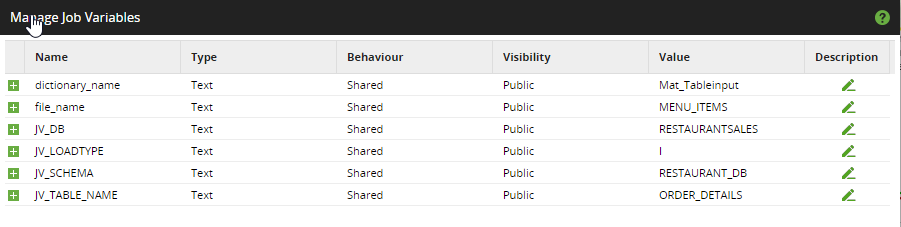
# DETAILED:



### Create an orchestration job V2\_1Active\_table\_check

### Prep:

* + - First Create following Job Variables:



Text Mode:

dictionary\_name Text Shared Public Mat\_Tableinput

file\_name Text Shared Public MENU\_ITEMS

JV\_DB Text Shared Public RESTAURANTSALES

JV\_LOADTYPE Text Shared Public I

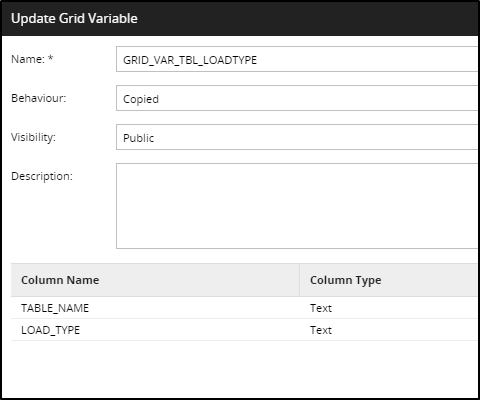
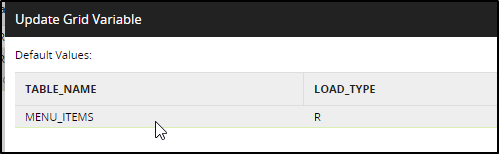
JV\_SCHEMA Text Shared Public RESTAURANT\_DB

JV\_TABLE\_NAME Text Shared Public ORDER\_DETAILS

* + - Create Grid variables:



GRID\_VAR\_TBL\_LOADTYPE

TEXT Mode:

TABLE\_NAME Text

LOAD\_TYPE Text

* Configuring SNS
  + AWS SNS > Create SNS topic > Configure subscription

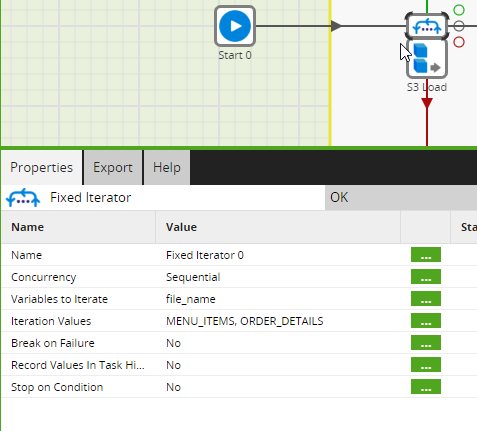
### Load the data from S3 to Snowflake src table [S3 LOAD + File Iterator]

* + **properties of S3 load:**

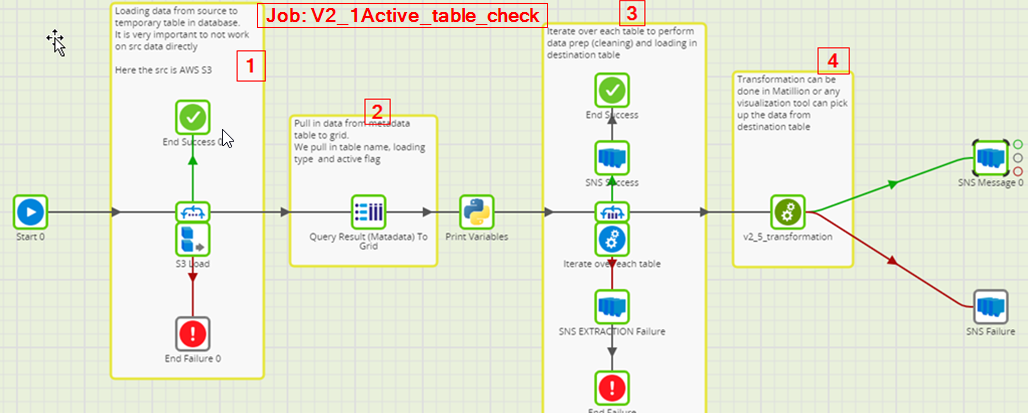
|  |  |
| --- | --- |
| Name | S3 Load |
| Stage | Matillion\_Stage |
| Pattern | ${file\_name}.csv |
| Warehouse | MATILLION\_PRG |
| Database | ${JV\_DB} |
| Schema | ${JV\_SCHEMA} |
| Target Table | ${file\_name.replace(/"/g,'')} |

NOTE: Please set ‘Force Load ‘ to True

**Properties of Fixed Iterator:**

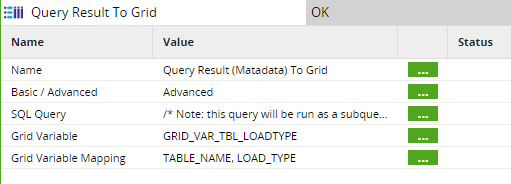


|  |  |  |
| --- | --- | --- |
| Variables to Iterate | file\_name |  |
| Iteration Values | MENU\_ITEMS, ORDER\_DETAILS |  | |



### Pull in info from metadata table into the GRID Variable [QUERY RESULT TO GRID]

Properties:



**SQL QUERY :**

SELECT

UPPER(TABLE\_NAME) TABLE\_NAME,UPPER(LOAD\_TYPE) LOAD\_TYPE,IS\_ACTIVE

FROM

RESTAURANTSALES.RESTAURANT\_METADATA.TABLE\_LOAD\_METADATA

WHERE IS\_ACTIVE =1

Python script – just to display variables (Also helps in displaying message in the logs

**Script:**

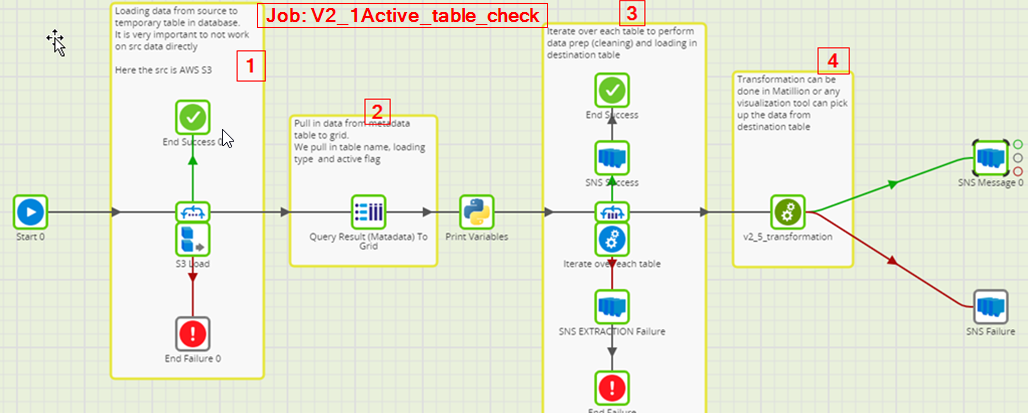
print (context.getGridVariable('GRID\_VAR\_TBL\_LOADTYPE'))

print(JV\_TABLE\_NAME)

print (JV\_LOADTYPE)

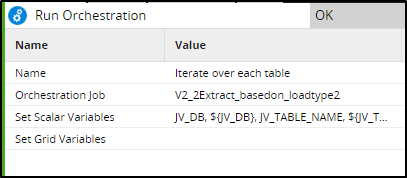
Logs:

### Iterate all the next steps for each table [FIXED ITEARATOR + ORCHESTRATION JOB] + Notification [SNS]



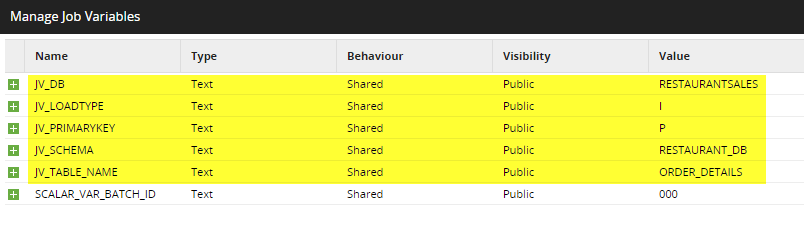
Points to ponder: Why we have added iterator here? Since we are iterating to another transformation job, why don’t we do it in the same job?

**Run Orchestration: V2\_2Extract\_basedon\_loadtype2**



First configure variables:

Job Variables:



Text mode:

JV\_DB Text Shared Public RESTAURANTSALES

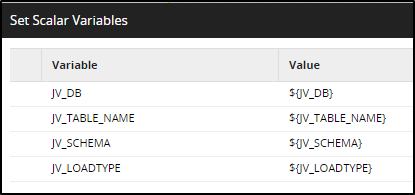
JV\_LOADTYPE Text Shared Public I

JV\_PRIMARYKEY Text Shared Public P

JV\_SCHEMA Text Shared Public RESTAURANT\_DB

JV\_TABLE\_NAME Text Shared Public ORDER\_DETAILS

—--------------------------------------------------------------------------------------------------------------------------------------------------



Text Mode:

JV\_DB ${JV\_DB}

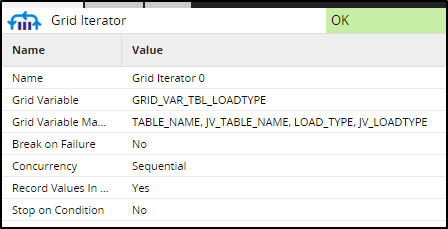
JV\_TABLE\_NAME ${JV\_TABLE\_NAME}

JV\_SCHEMA ${JV\_SCHEMA}

JV\_LOADTYPE ${JV\_LOADTYPE}

Points to ponder? What is set scalar variables?

**Iterator:**

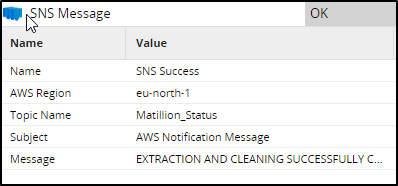
****

Text Mode:

TABLE\_NAME JV\_TABLE\_NAME

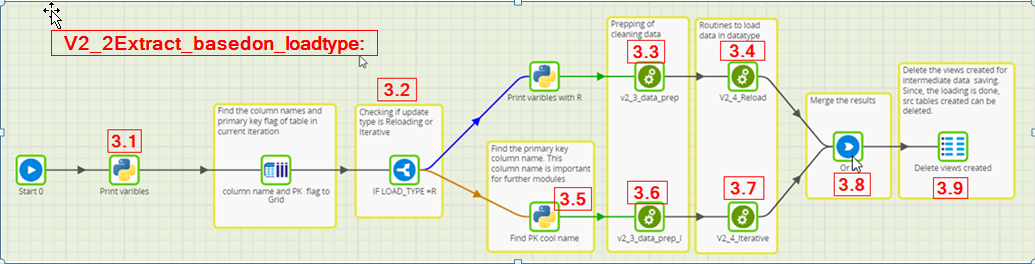
LOAD\_TYPE JV\_LOADTYPE

SNS Message:

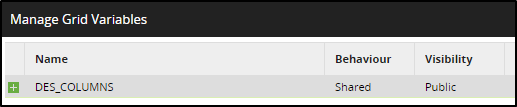


|  |  |
| --- | --- |
| Message for success:  EXTRACTION AND CLEANING SUCCESSFULLY COMPLETED  JOB ID:${job\_id}  JOB NAME: ${job\_name}  MESSAGE:${component\_message} | Message for failure:  Needs attention!!! Job failed :-(  JOB ID:${job\_id}  JOB NAME: ${job\_name}  MESSAGE:${component\_message}  DETAILED ERROR:${detailed\_error} |

End success and end failure:

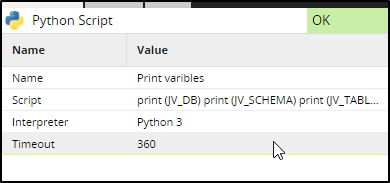


Create another Orchestration job: **V2\_2Extract\_basedon\_loadtype2**



|  |  |
| --- | --- |
| Text Mode:  COl\_name Text  Primary\_key Text | Text Mode:  ORDER\_DETAILS\_ID Yes  ORDER\_ID No  ORDER\_DATE No  ORDER\_TIME No |

#### Retrieve the column name and primary key of the table [table metadata to grid]

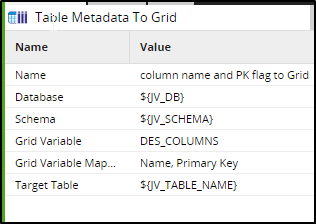


print (JV\_DB)

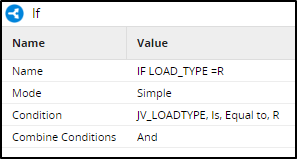
print (JV\_SCHEMA)

print (JV\_TABLE\_NAME)

Table Metadata To Grid:

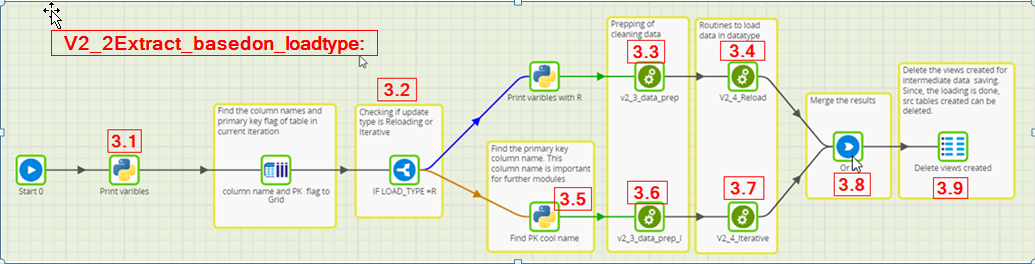


#### Check if the table needs to be iterated or reloaded [IF component]





#### For R: Prepare /clean the data [Run transformation component]



**Python component:**

print (JV\_DB)

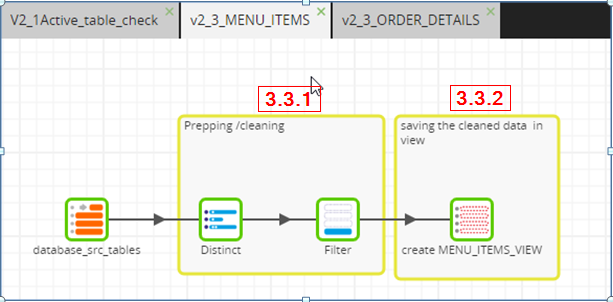
print (JV\_SCHEMA)

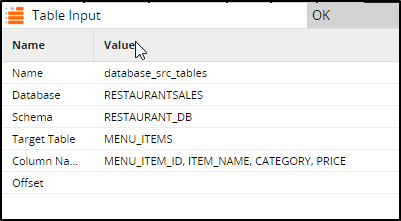
print (JV\_TABLE\_NAME)

print(context.getGridVariable('DES\_COLUMNS'))

—------------------------------------------------------------------------------------

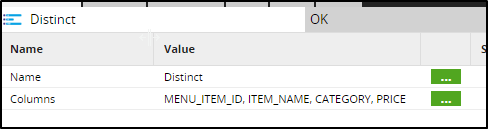
Create a Transformation job – V2\_3\_MENU\_ITEMS



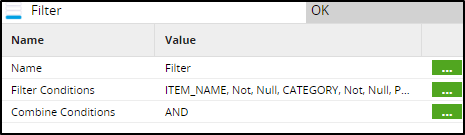


#### Cleaning [DISTINCT + FILTER NULLS]

Distinct:



Filter:



Filter Condition (Text Mode)

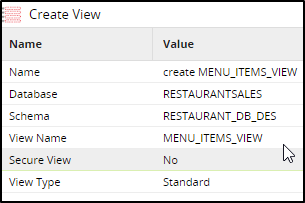
ITEM\_NAME Not Null

CATEGORY Not Null

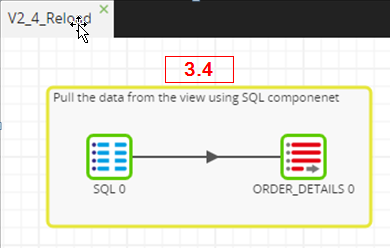
PRICE Not Null

#### Save it in a view [CREATE VIEW]

Create view:



**Create a Transformation job – V2\_4\_Reload**



* Configure job variables:

JV\_DB Text Shared Public RESTAURANTSALES

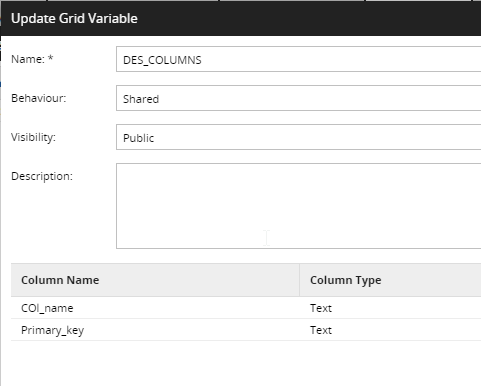
JV\_SCHEMA Text Shared Public RESTAURANT\_DB

JV\_TABLE\_NAME Text Shared Public MENU\_ITEMS

* configure grid variable

DES\_COLUMNS grid variables

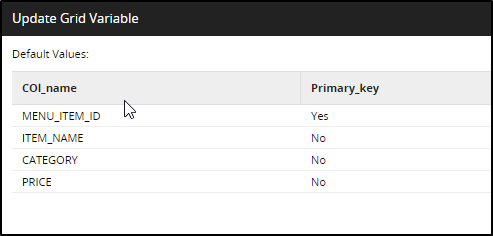




TEST mode:

COl\_name Text

Primary\_key Text



TExt mode:

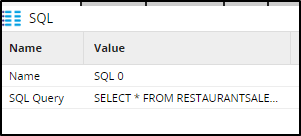
MENU\_ITEM\_ID Yes

ITEM\_NAME No

CATEGORY No

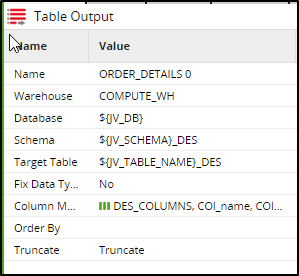
PRICE No

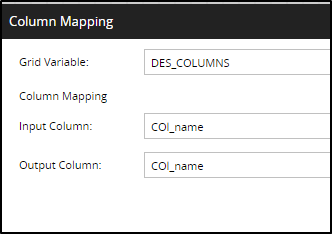
* SQL:



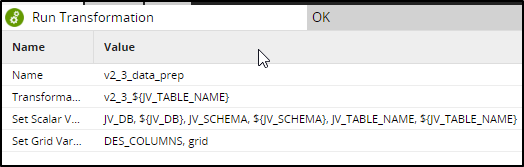
SELECT \* FROM RESTAURANTSALES.RESTAURANT\_DB\_DES.${JV\_TABLE\_NAME}\_VIEW

Table output: - write an input data flow out to an existing output table.





—-----------------------------------------------------------------------------------



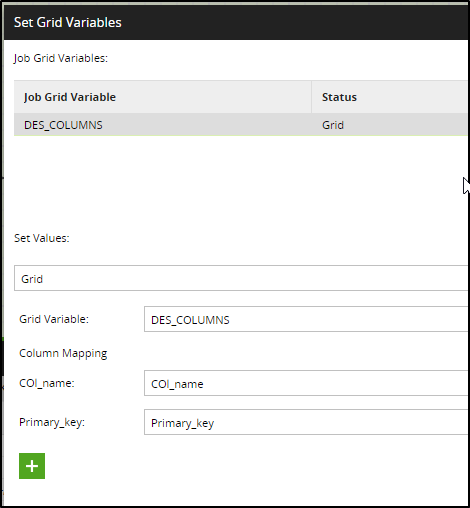
v2\_3\_${JV\_TABLE\_NAME}

Text Mode:

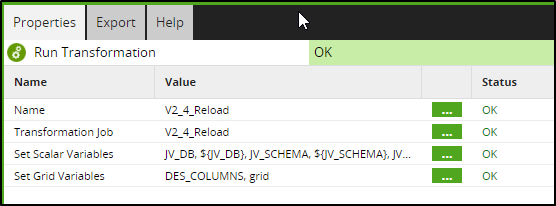
JV\_DB ${JV\_DB}

JV\_SCHEMA ${JV\_SCHEMA}

JV\_TABLE\_NAME ${JV\_TABLE\_NAME}



#### For R : Load the data from the VIEW into the destination database [RUN transformation component]

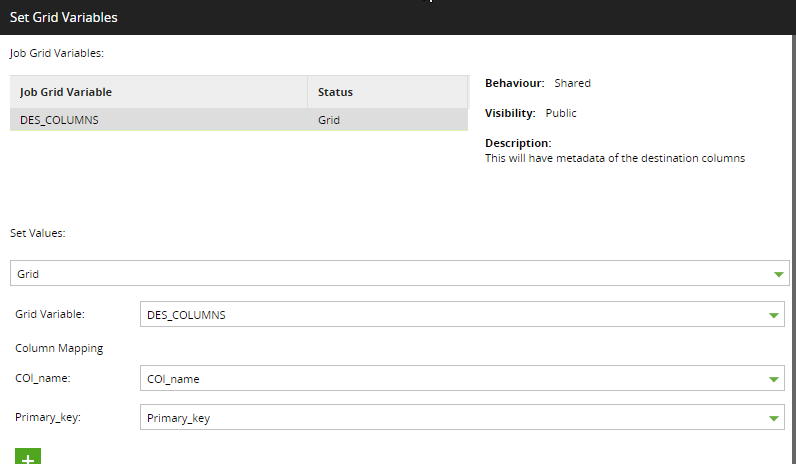


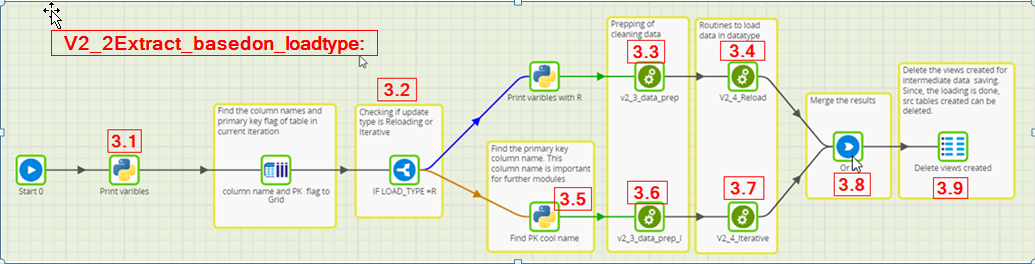
set scalar variable:

JV\_DB ${JV\_DB}

JV\_SCHEMA ${JV\_SCHEMA}

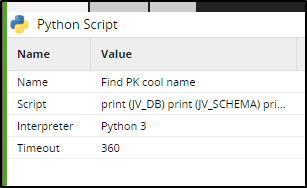
JV\_TABLE\_NAME ${JV\_TABLE\_NAME}





#### For I: Identify the name of the primary key using the flag [Python script]

Python script:



for i in context.getGridVariable('DES\_COLUMNS'):

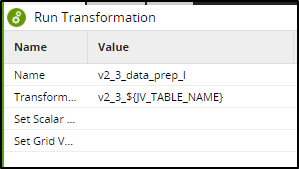
if i[1]=='Yes':

context.updateVariable('JV\_PRIMARYKEY', i[0])

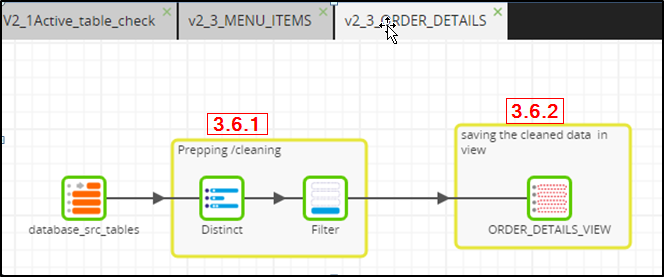
print(JV\_PRIMARYKEY)

#### For I: Prepare /clean the data [Run transformation component]

Run transformation component:



Create a Transformation job – V2\_3\_ORDER\_DETAILS



adding variables:

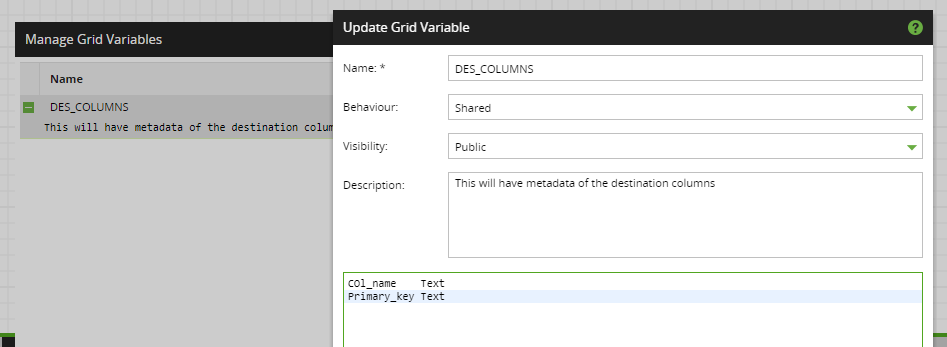
text mode:

JV\_DB Text Shared Public RESTAURANTSALES

JV\_SCHEMA Text Shared Public RESTAURANT\_DB

JV\_TABLE\_NAME Text Shared Public ORDER\_DETAILS

Grid variables:



COl\_name Text

Primary\_key Text

Values:

ORDER\_DETAILS\_ID Yes

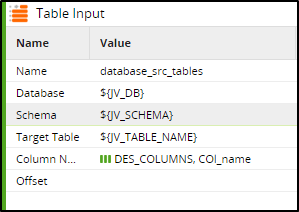
ORDER\_ID No

ORDER\_DATE No

ORDER\_TIME No

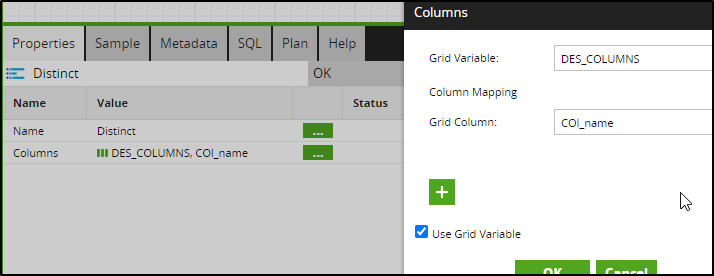
ITEM\_ID No

Table input:

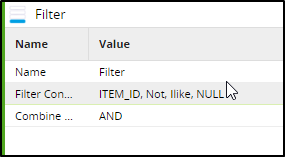


#### Cleaning [DISTINCT + FILTER NULLS]

Distinct

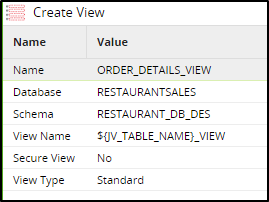


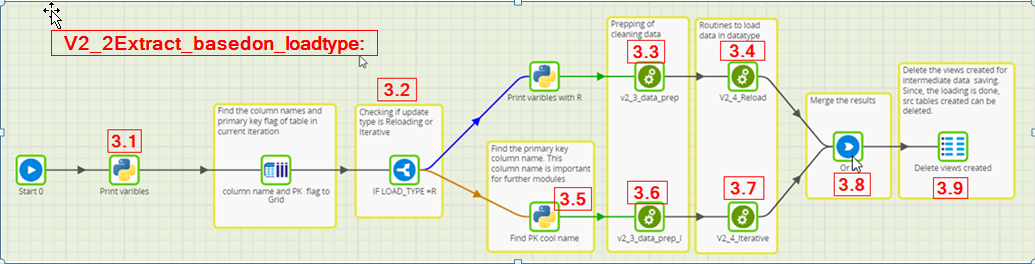
Filter



#### Save it in a view [CREATE VIEW]

View

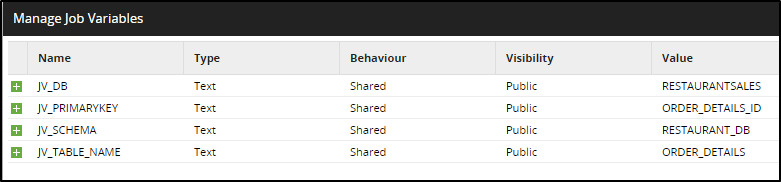


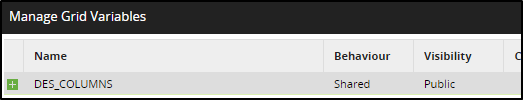


#### For I: Load the data from the VIEW into the destination database [RUN transformation component]

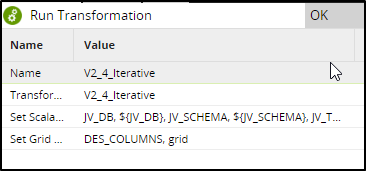
Create a Transformation job – V2\_4\_Iterative

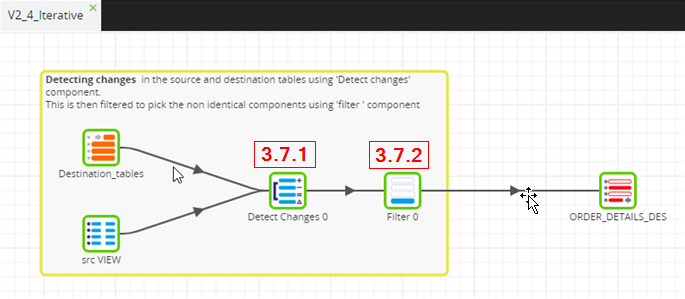
Copy the Grid variables and Job variables





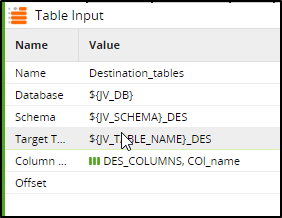
Run transformation component:





#### Detect changes [Detect changes]

Table input:

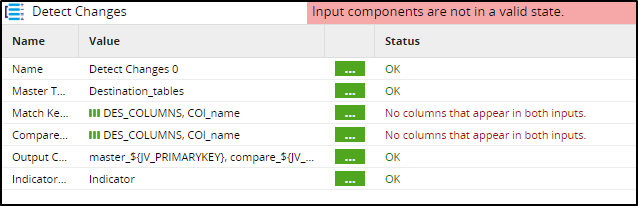


**SQL query**

SELECT \* FROM RESTAURANTSALES.RESTAURANT\_DB\_DES.${JV\_TABLE\_NAME}\_VIEW

**Detect changes**

Don’t worry about the errors at this point. It’s because these don’t have columns its searching for. Once the iteration start, the right columns will come in.



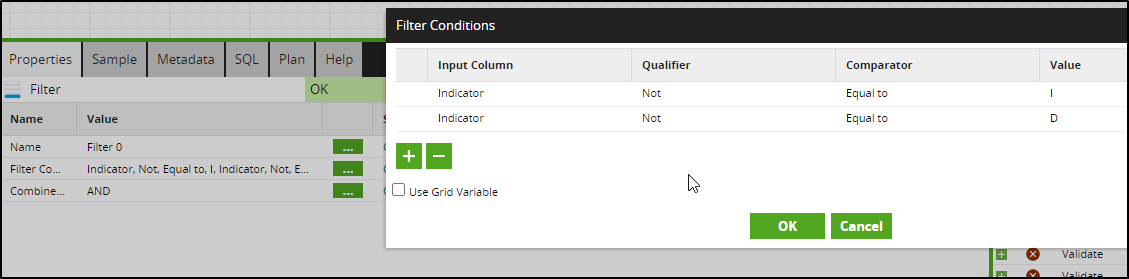


Test mode:

master\_${JV\_PRIMARYKEY} compare\_${JV\_PRIMARYKEY}

#### Filter rows that need to be Upserted [filter]

**Filter**

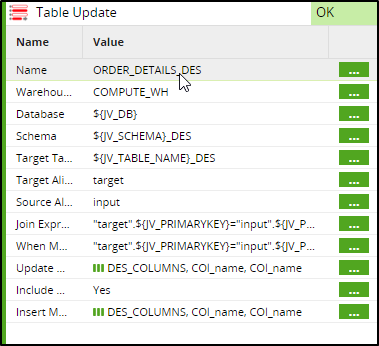


Text mode:

Indicator Not Equal to I

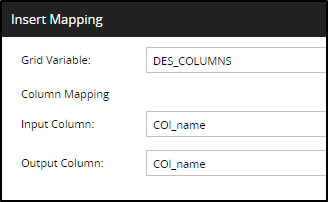
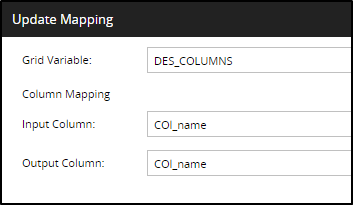
Indicator Not Equal to D

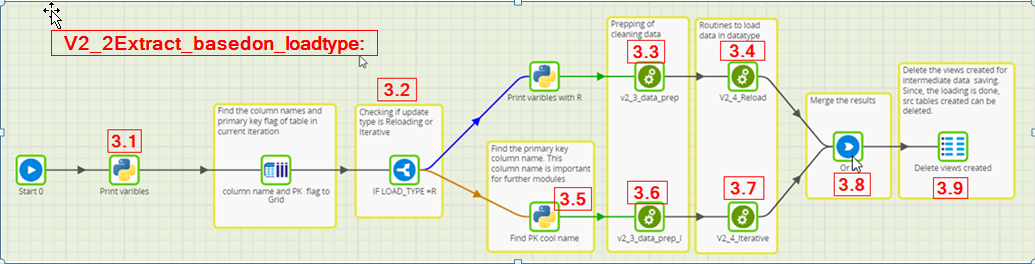
**Upsert**



JOIN expression: "target".${JV\_PRIMARYKEY}="input".${JV\_PRIMARYKEY}

WHEN marched: "target".${JV\_PRIMARYKEY}="input".${JV\_PRIMARYKEY} Update



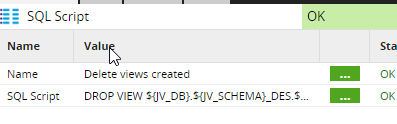


#### Irrespective of R or I , the pipeline needs to be merged [OR]

OR

#### Since the cleaning and loading is completed, Unnecessary data/ structure is removed [SQL SCRIPT]

SQL script

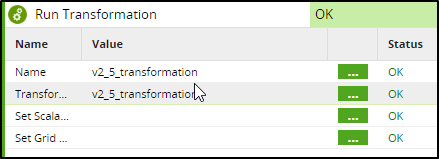


Script:

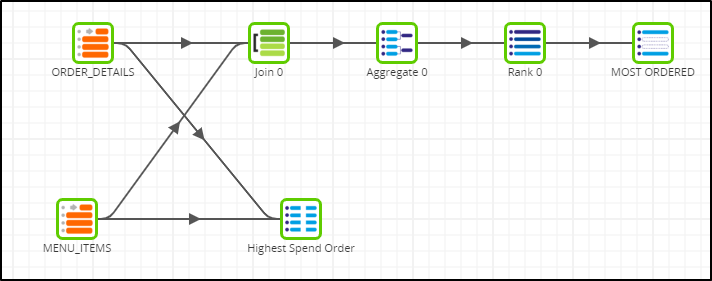
DROP VIEW ${JV\_DB}.${JV\_SCHEMA}\_DES.${JV\_TABLE\_NAME}\_VIEW;

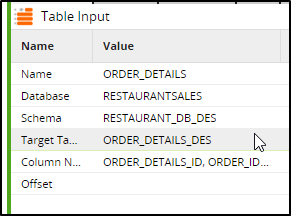
-- TRUNCATE TABLE ${JV\_DB}.${JV\_SCHEMA}.${JV\_TABLE\_NAME}

#### You can perform transformations to retrieve value /KPI OR use the data as input to any visualization tool. [RUN transformation component]

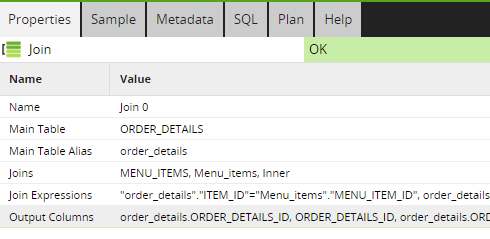


Create Transformation Job – V2\_5\_Tranformation



ORDER DETAILS/Menu Items:  


JOIN:



JOINS: "order\_details"."ITEM\_ID"="Menu\_items"."MENU\_ITEM\_ID"

JOIN Expression: "order\_details"."ITEM\_ID"="Menu\_items"."MENU\_ITEM\_ID"

OUTPUT COLUMS:

order\_details.ORDER\_DETAILS\_ID ORDER\_DETAILS\_ID

order\_details.ORDER\_ID ORDER\_ID

order\_details.ORDER\_DATE ORDER\_DATE

order\_details.ORDER\_TIME ORDER\_TIME

order\_details.ITEM\_ID ITEM\_ID

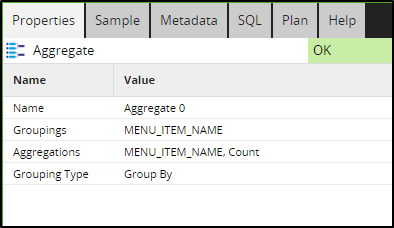
Menu\_items.MENU\_ITEM\_ID MENU\_ITEM\_ID

Menu\_items.ITEM\_NAME MENU\_ITEM\_NAME

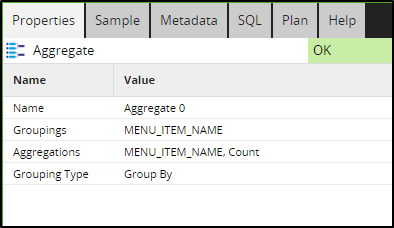
Menu\_items.CATEGORY MENU\_CATEGORY

Menu\_items.PRICE MENU\_PRICE

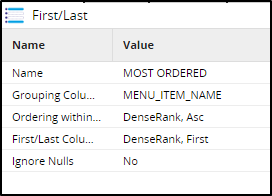
AGGREGATE:



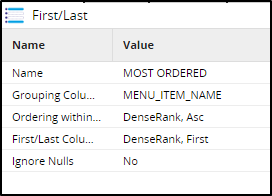
RANK:



FIRST/LAST:



SQL:



SELECT OD.order\_ID, SUM(PRICE) ORDER\_SPEND FROM RESTAURANTSALES.RESTAURANT\_DB\_DES.ORDER\_DETAILS\_DES OD

INNER JOIN RESTAURANTSALES.RESTAURANT\_DB\_DES.MENU\_ITEMS\_DES MI

ON OD.ITEM\_ID=MI.MENU\_ITEM\_ID

GROUP BY order\_ID

ORDER BY 2 DESC LIMIT 1

SNS Message: Same as first SNS

**Input to Viz tool:**

You could use this data you stored in the destination database (in snowflake) as input to any visualization tool.

Here, we will just integrate with power BI and if time permits we can change the data src to new destination database.

**GIT Integration:**

* + - * + [**https://docs.matillion.com/data-productivity-cloud/designer/docs/git-overview/**](https://docs.matillion.com/data-productivity-cloud/designer/docs/git-overview/)

**Scheduling:**

* + - * + **Project>schedules>add schedules>create new schedules**
        + **Uses Cron expressions**
        + [**https://docs.matillion.com/data-productivity-cloud/designer/docs/schedules/**](https://docs.matillion.com/data-productivity-cloud/designer/docs/schedules/)

References:

1. <https://www.matillion.com/blog/matillion-recognized-as-a-challenger-in-the-2023-gartner-magic-quadrant-for-data-integration-tools>
2. <https://docs.matillion.com/metl/docs/2037630/>
3. <https://www.matillion.com/blog/building-data-pipelines-always-on-tables-with-matillion-etl>
4. <https://www.matillion.com/blog/what-is-etl-the-ultimate-guide>
5. <https://docs.matillion.com/metl/docs/2917841/>
6. <https://docs.matillion.com/metl/docs/2943425/#contact-support>