**INTERNAL PROJECT**

**CH Durga Prasad**

**Batch-4**

**Project Data:**



**Project Objective PPT:**



**Requirements:**

1. Google sheet Api and service account
2. Aws S3 bucket
3. Glue Job
4. Crawler
5. Data Base
6. Athena
7. Snowflake
8. Airflow
9. Some additional Libraries like:

* Pandas, gspread, json, boto3, datetime, send\_email, Dag.

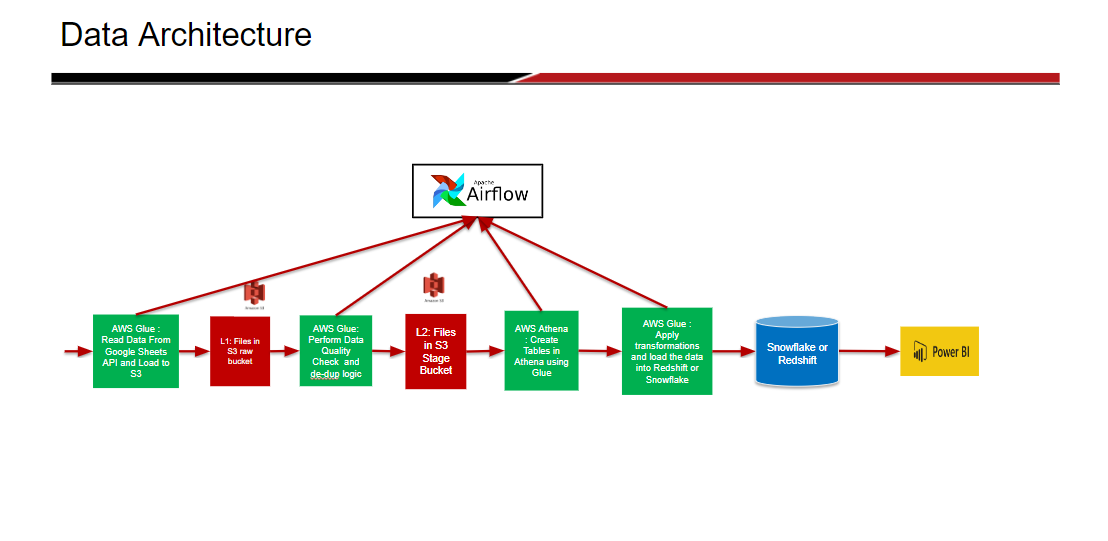
**Buckets:**

1. Kpi-batch4 >> DurgaPrasad >> raw\_data (Raw data extracted from google sheets)
2. Kpi-batch4 >> DurgaPrasad >> stagefolder (Transformed data from raw\_data using de-dup logic)
3. Kpi-batch4 >> DurgaPrasad >> target (Transformed data from stagefolder by adding specific columns to the data)
4. Kpi-batch4 >> DurgaPrasad >> denormal (Transformed data from target by adding specific columns to the data to denormal)

**Gluejob Names:**

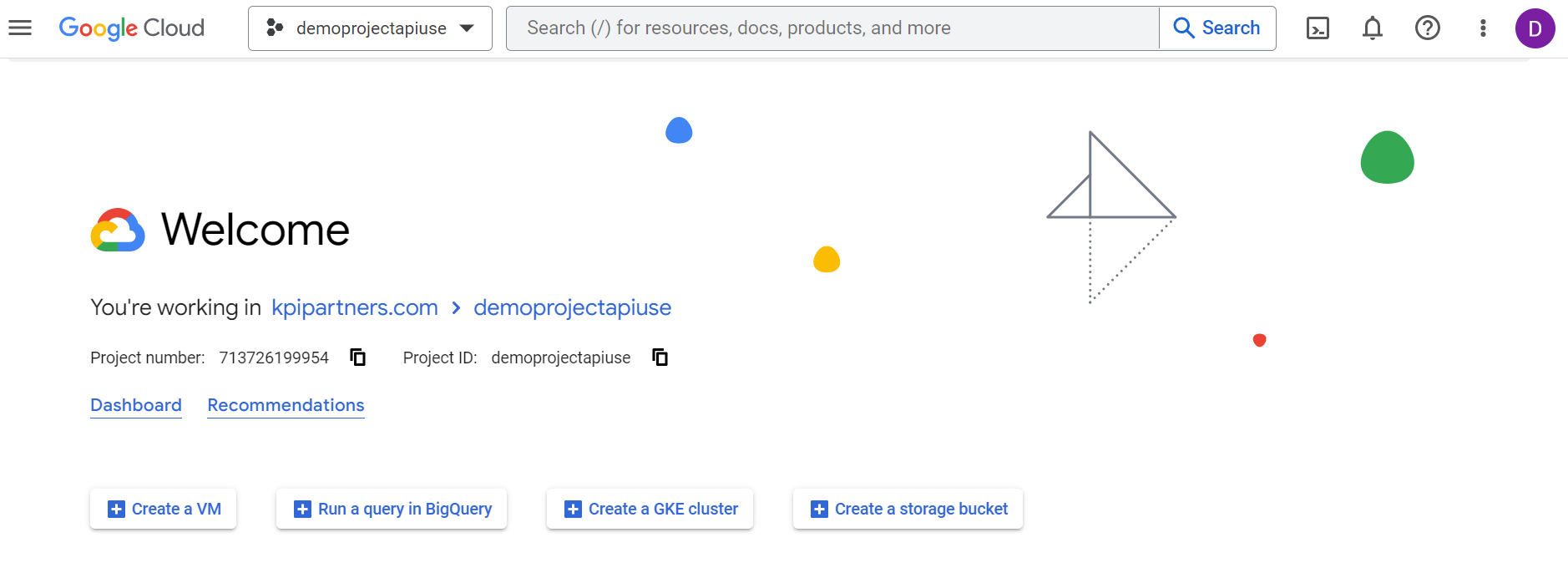
1. durga\_raw\_data (Api to raw bucket)
2. durga\_stage (Raw bucket to stage bucket)
3. durga\_job\_target (Stage bucket to Target & snowflake)
4. durga\_job\_denorm (Stage bucket to De-norm Table)

**Project Data Architecture:**

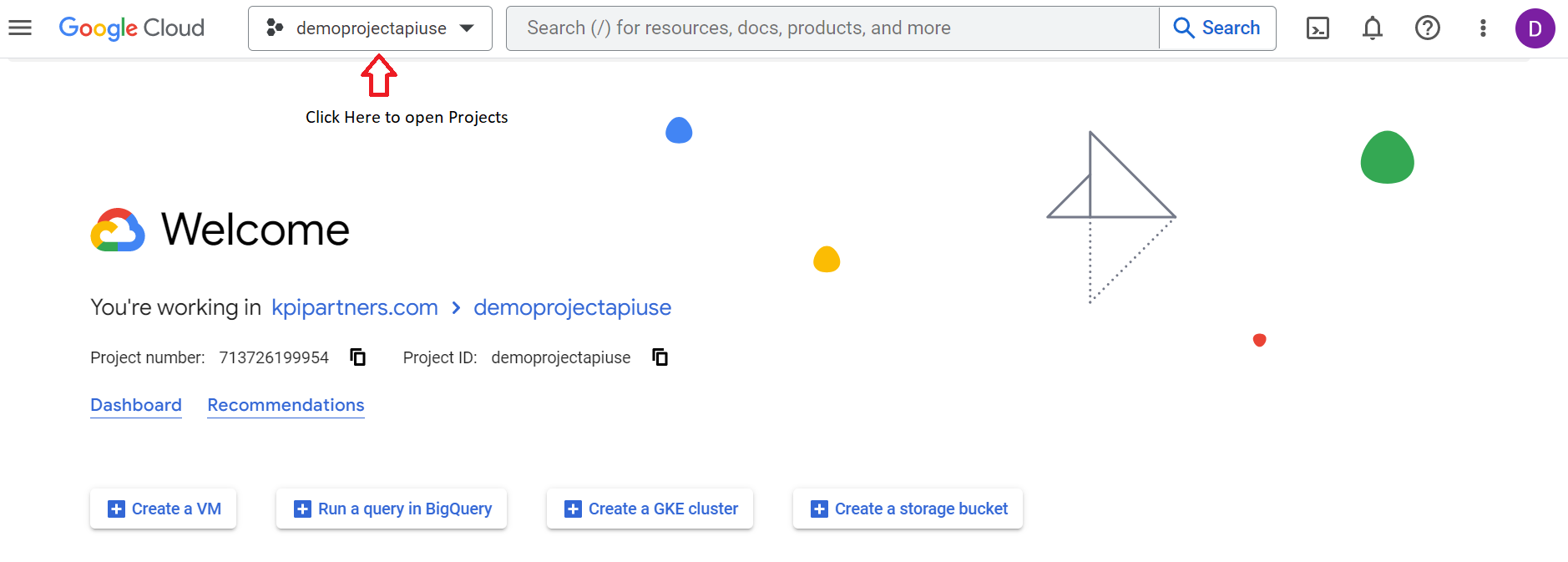


**Task 1: Setting up google sheet API to read data**

           Step - 1: Open google cloud console



Step - 2: Create a new project

****

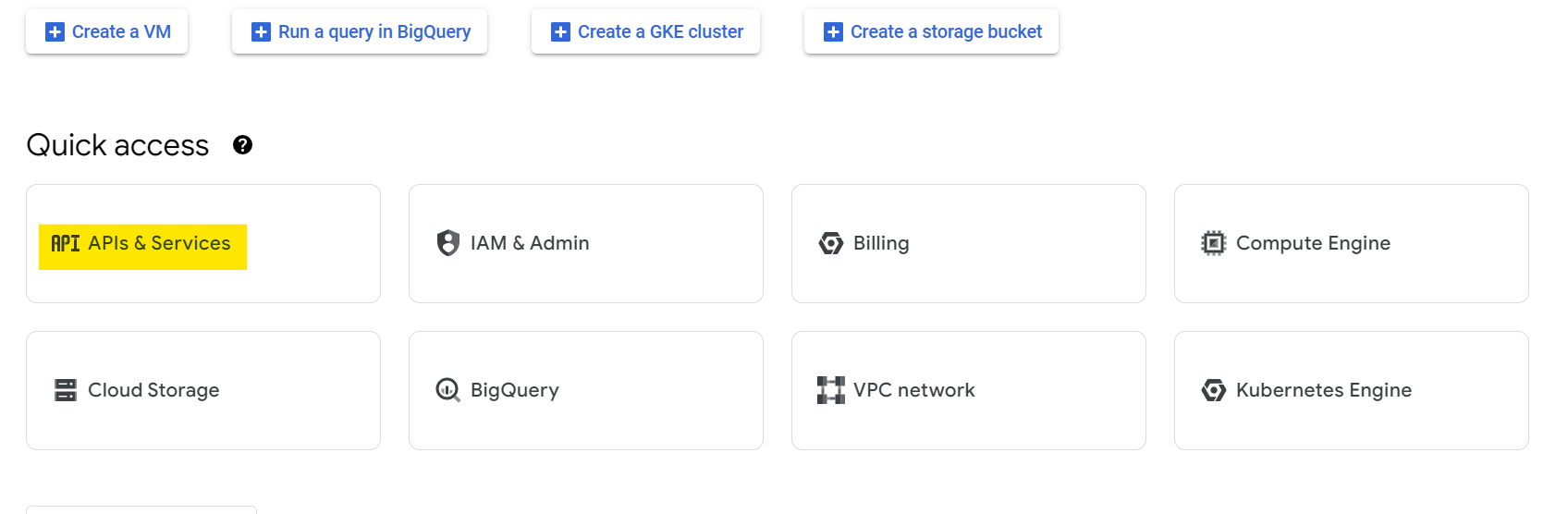
* Click on **NEW PROJECT**



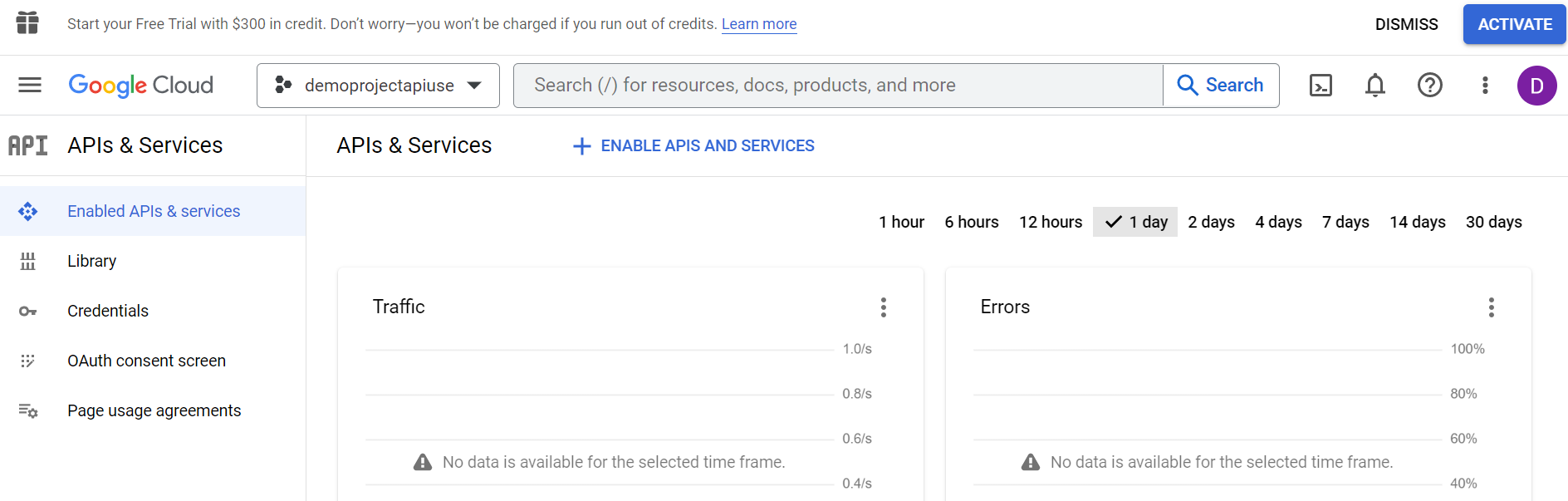
* Give **Project Name** and then click on **CREATE**

Step - 3: Enable google sheets API and google drive API

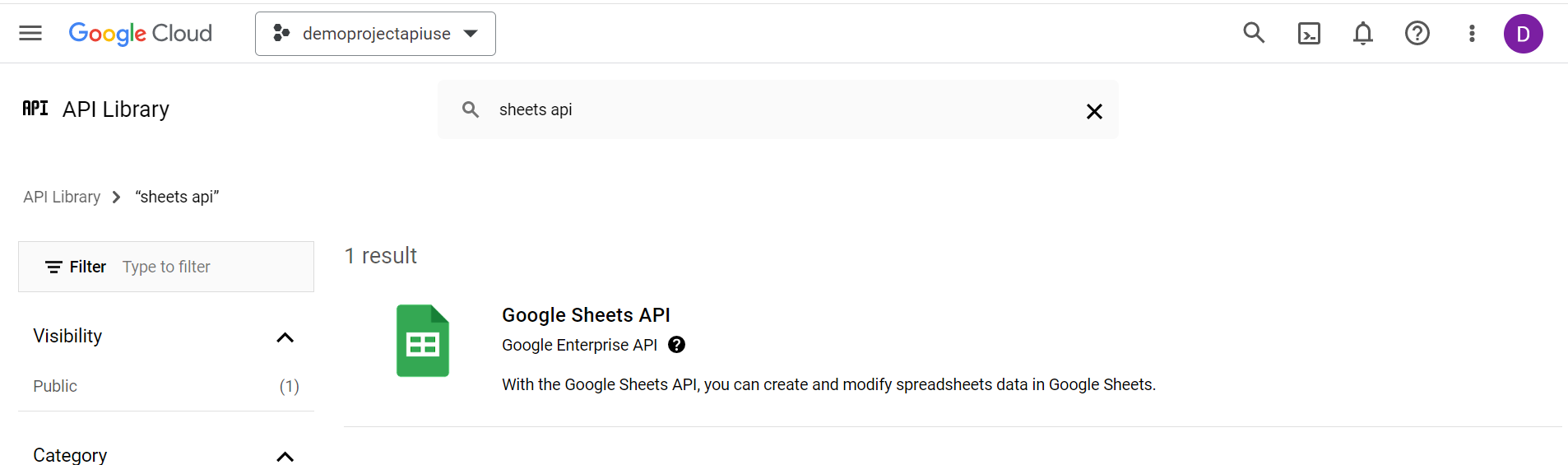
* Go to API and Services

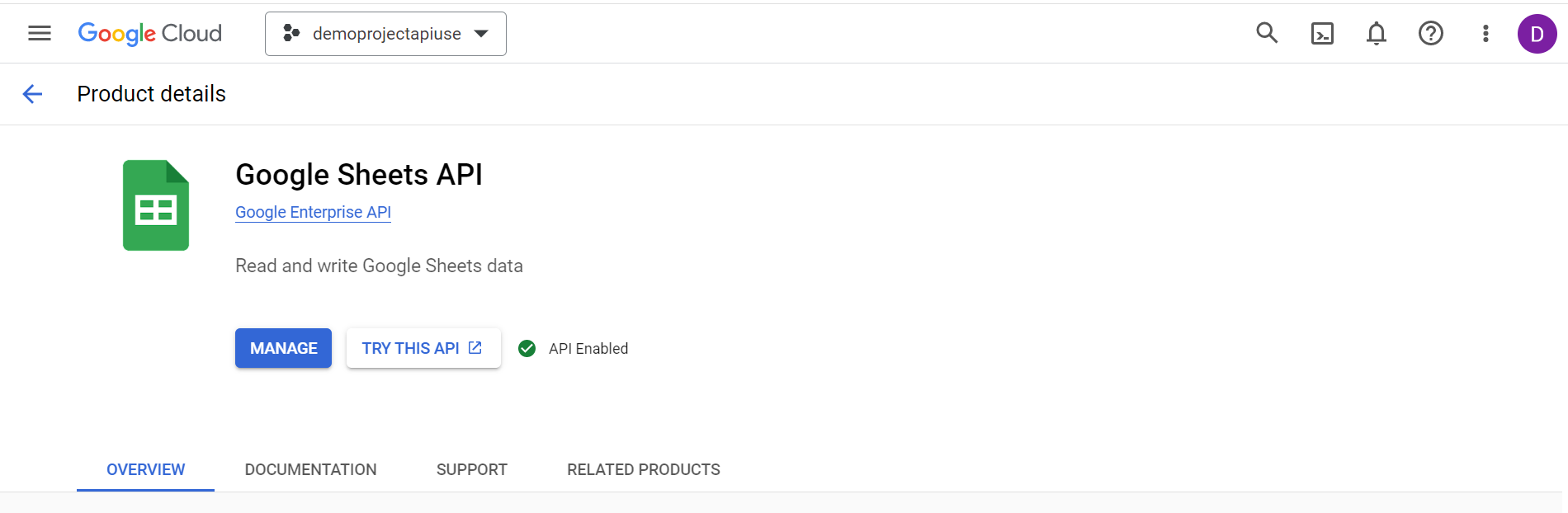


* Click on Enable API and Services



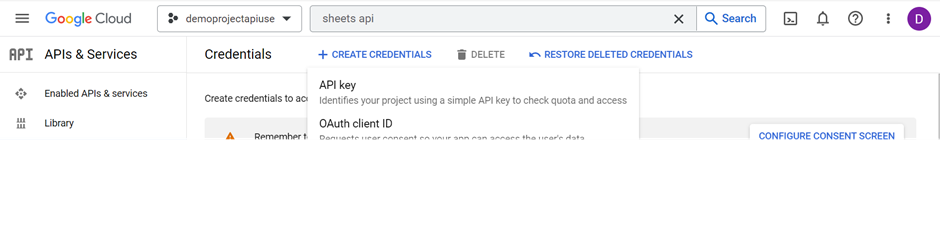
* Search for sheets api and drive api and Enable it



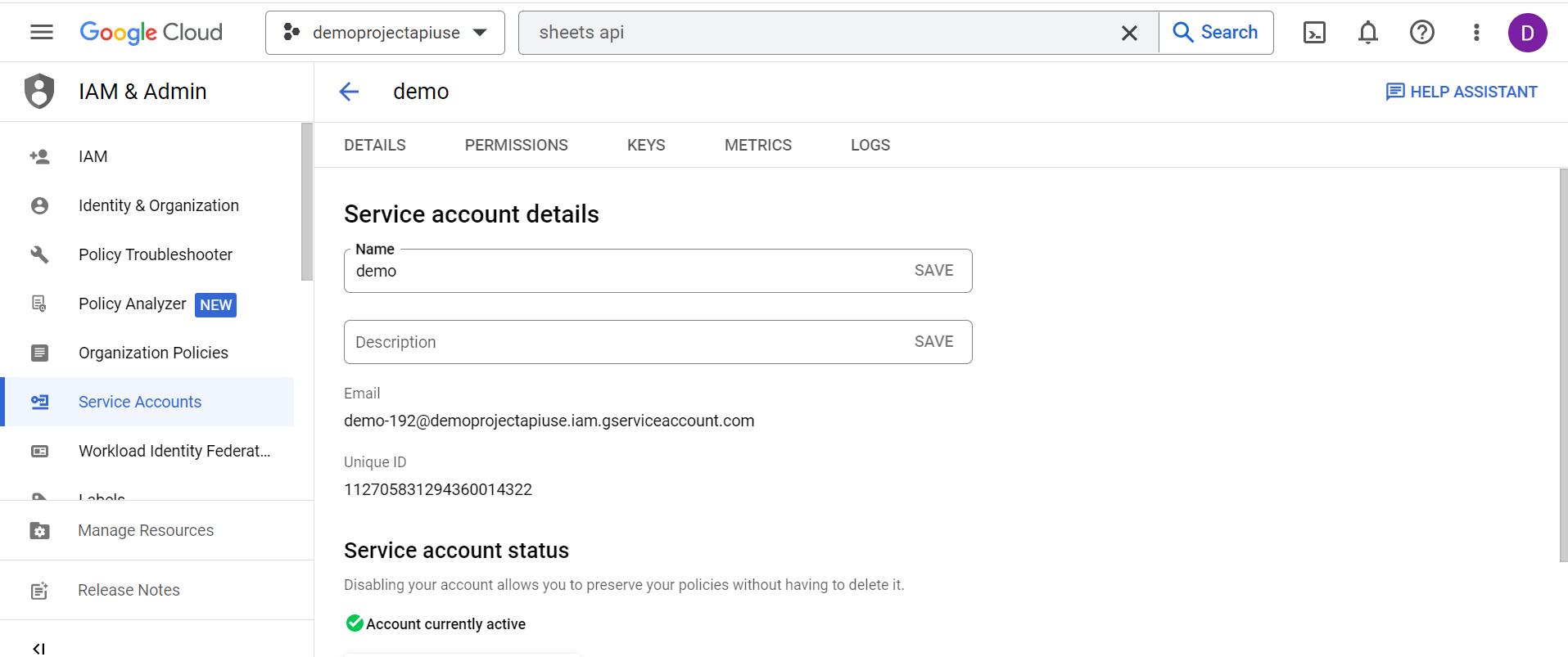


Step - 4: Create Service Account Credentials for API access

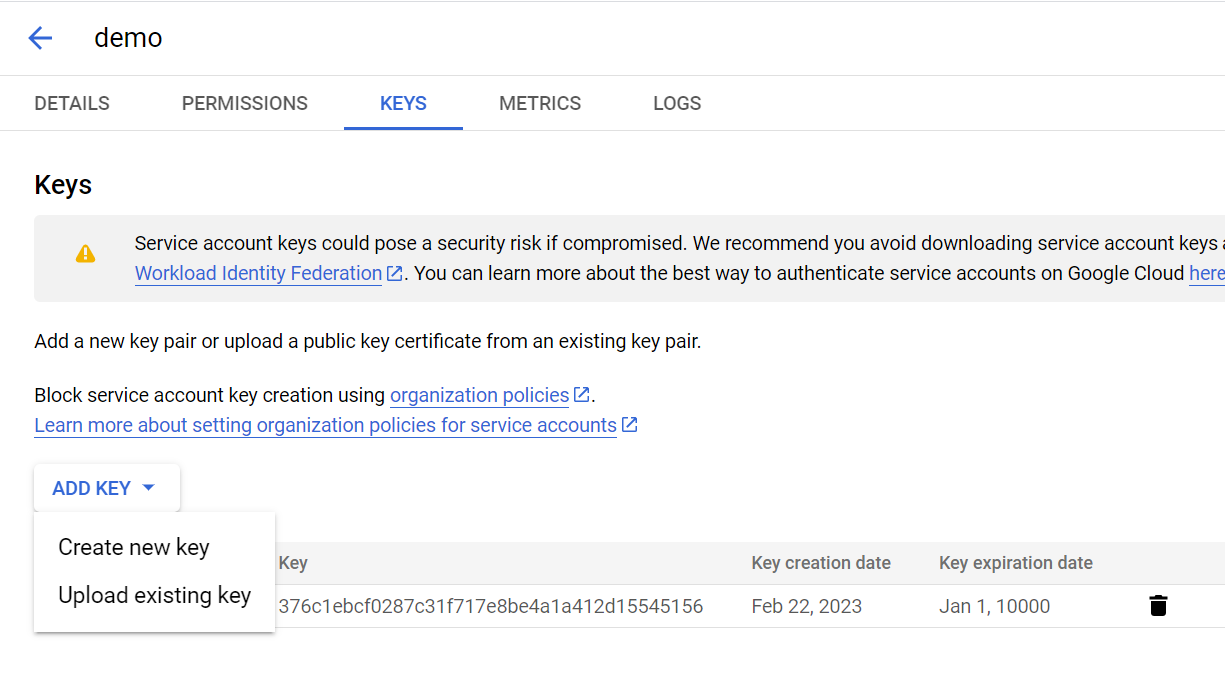
* Go to **Credential** in **API and Services and Click on CREATE CREDENTIALS** and select **SERVICE ACCOUNT.**



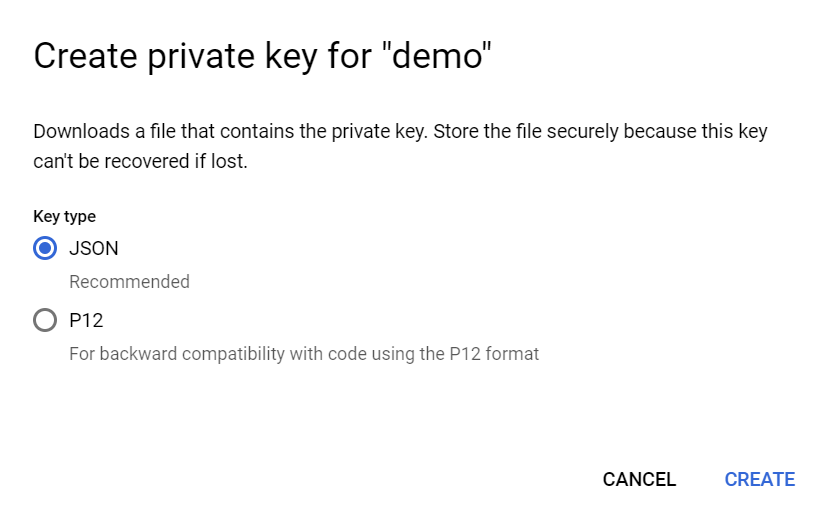
Open the created service account.



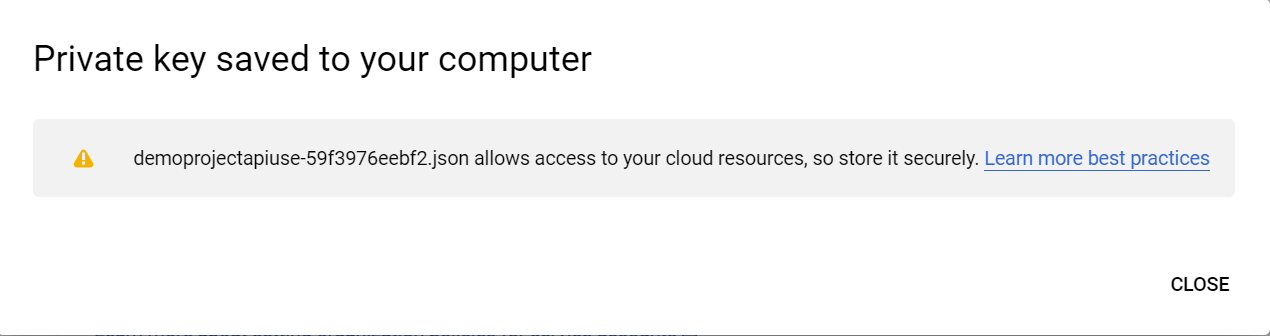
* Go to KEYS and Click on ADD KEY >> Create New Key



* Select **Key Type** as **json** and Click on **Create.**

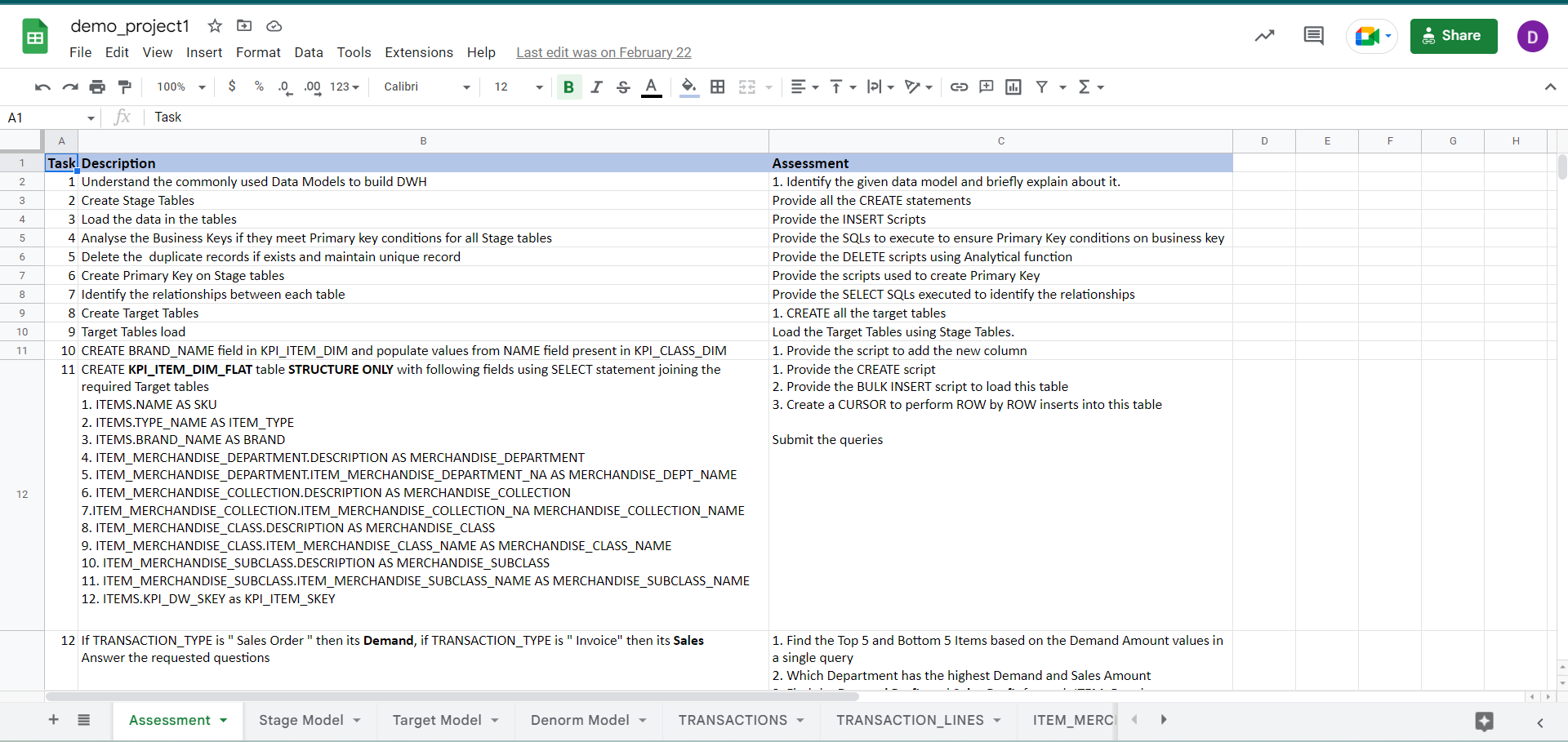


* It will create a credential .json file save it on the working directory to access the sheet API and read data.

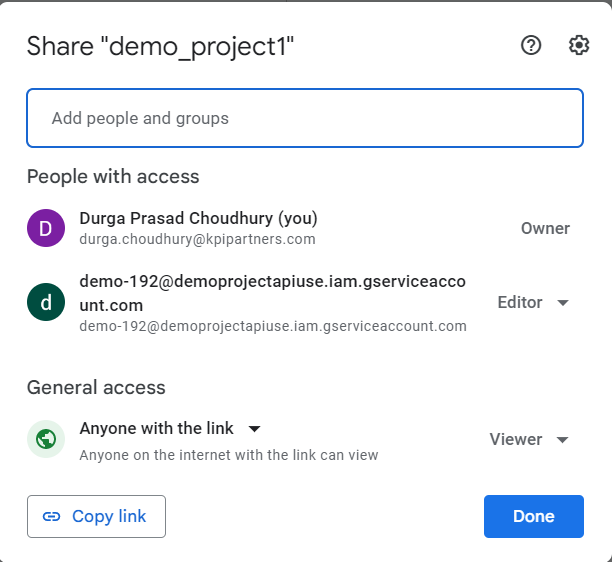


Step - 5: Add the service account email to the data sheet to read data.

* Open required google spread sheet and click on **Share**.



* Copy paste the service account email in add people and groups and click on **share**.

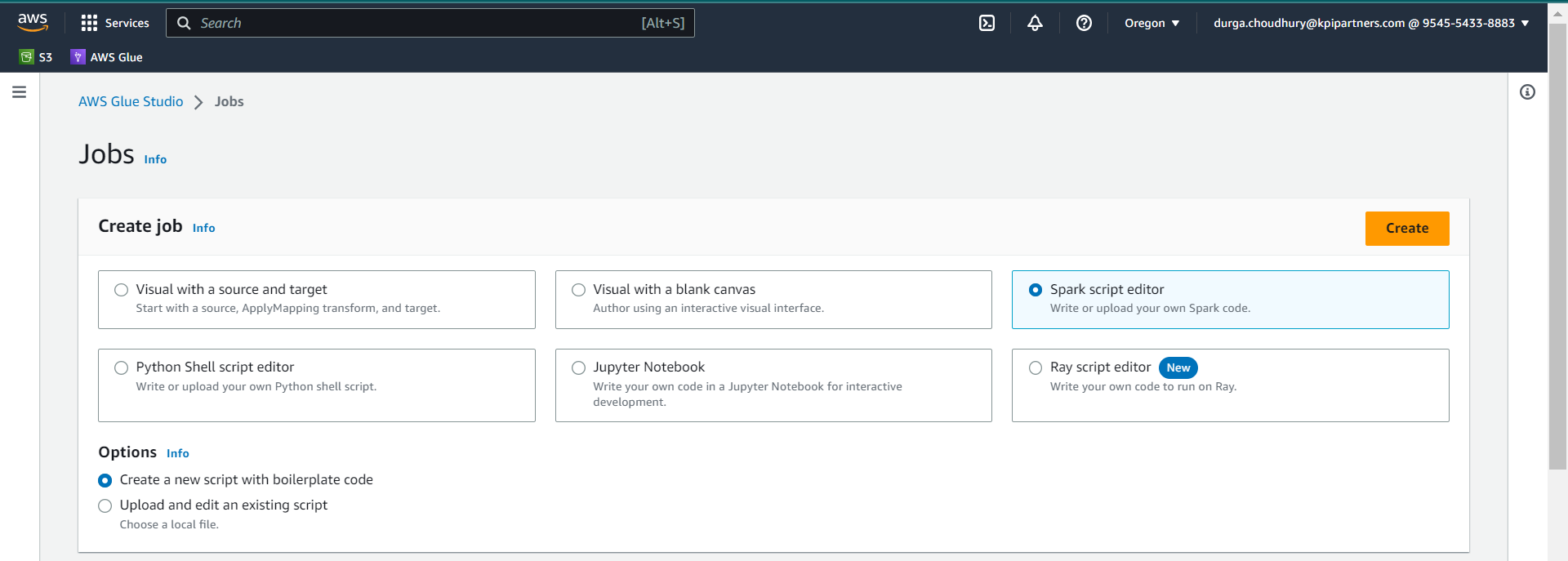


* **Steps to create Glue Job Script**

          Step - 1: Open AWS Glue and click on **Jobs**.



Step - 2: Select **Spark Script Editor** and Click on **Create.**



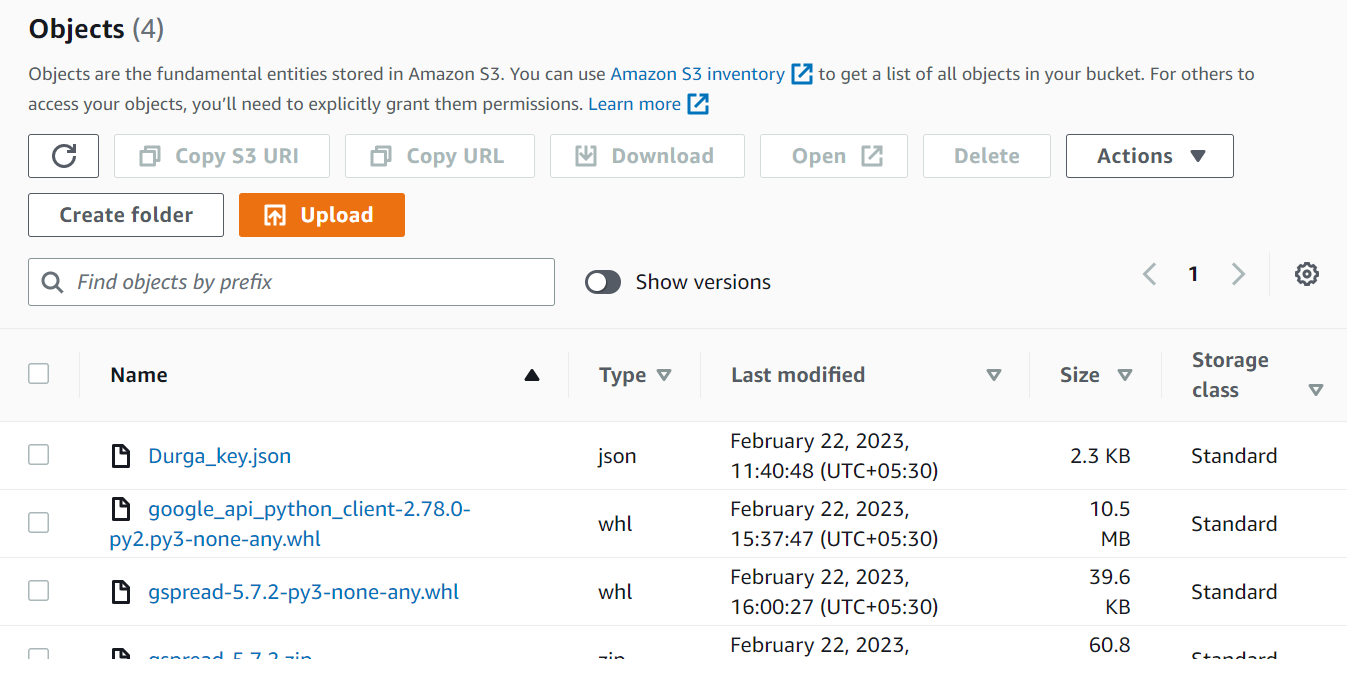
Step- 3: Give a name of your choice:

Step-4: Go to the Job Details section:

Step – 5:  Give the same name as of job name.

Step – 6: Choose the required IAM role

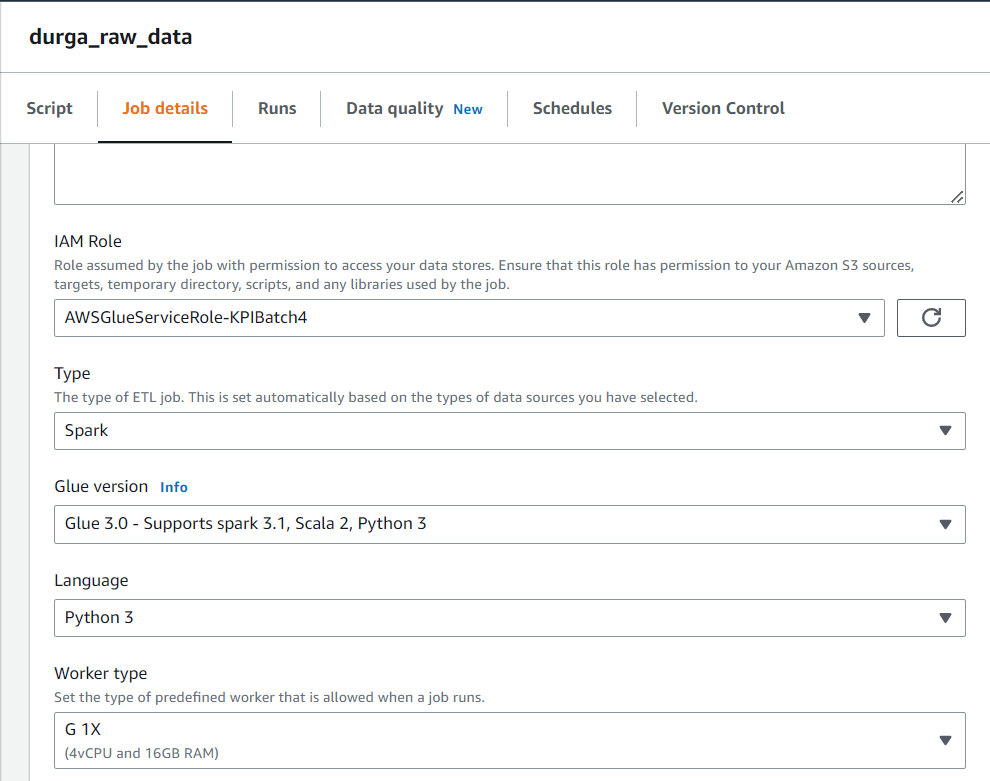
Step – 7: Download a “gspread-5.7.2-py3-none-any.whl” file to read gspread into s3 glue job.



* Download the “json\_key ” from google api at the time of creating and enabling the Google Sheet API and Google Drive API from your service account
* Store the json\_key and .whl file in the s3 bucket in the jar\_files of your s3 bucket.

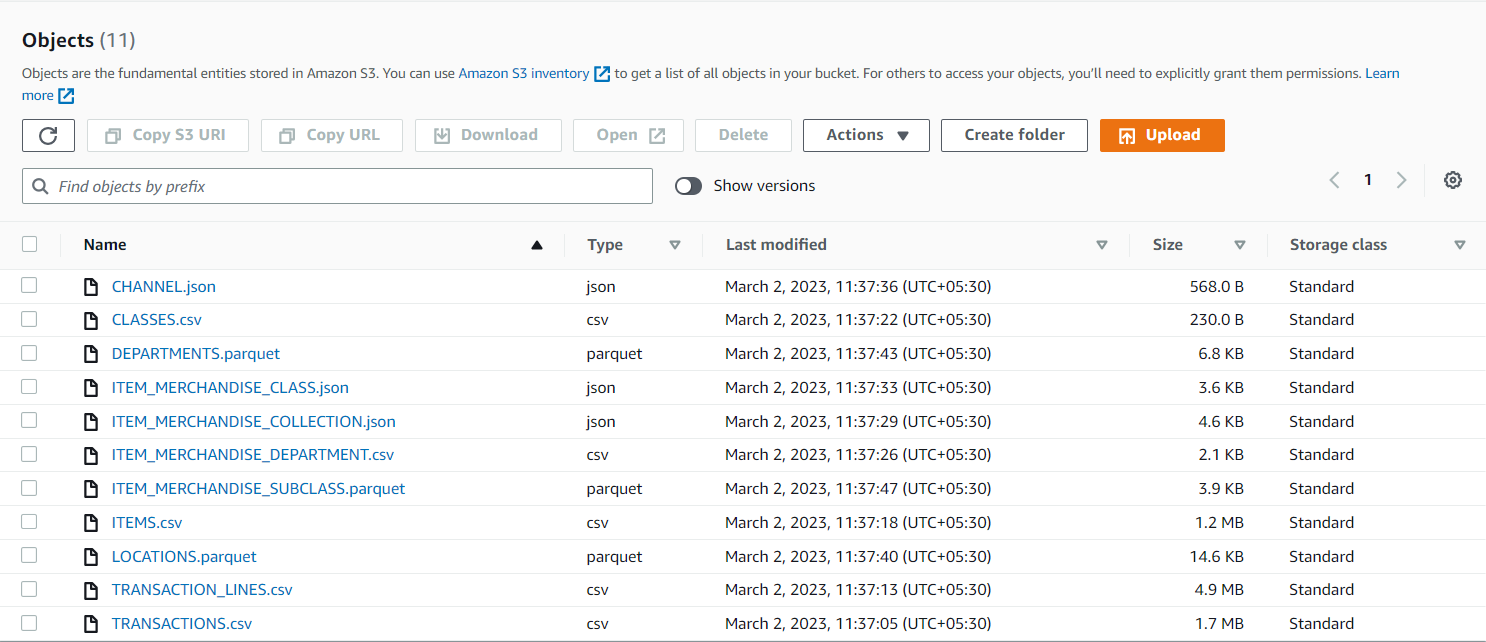
**Code File:**



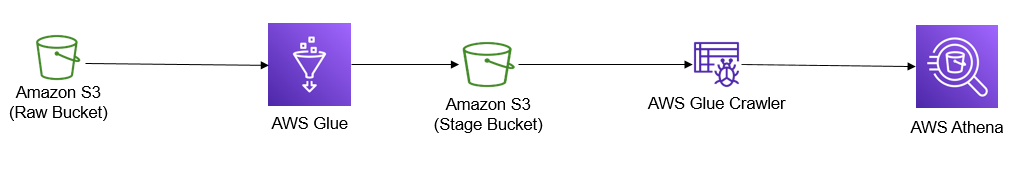
****

After running the current job we ca see the number of files loaded into s3 location

S3 Location: s3://kpi-batch4/DurgaPrasad/raw\_data/



**Stage-2: Raw layer to stage layer**

****

**Stage Layer Requirements:**

1. Load the data from raw files in to the stage bucket.

    Transformations:

    1. Apply de-dup logic and load the distinct set of rows into stage bucket.

    2. Add audit columns like process\_date to all files.

2. Create stage tables using Glue Crawler/Manually by following the Stage Model tab.

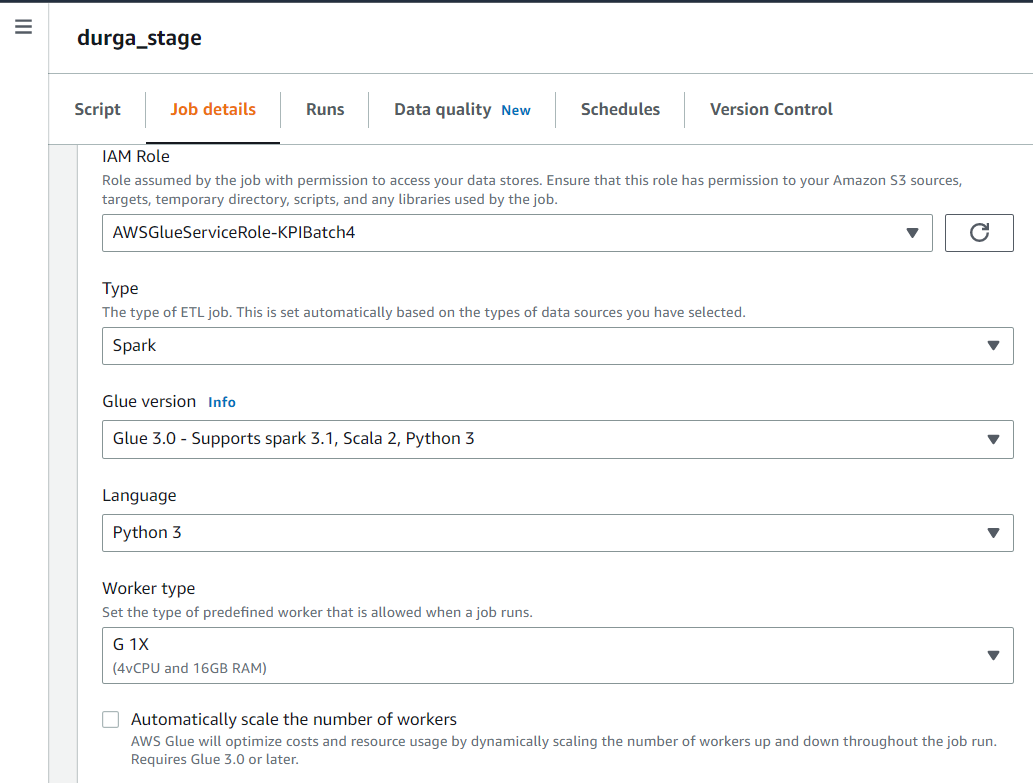
**Resources:**

Glue, S3, Athena, IAM Role.

* Create a new glue job as we did earlier.

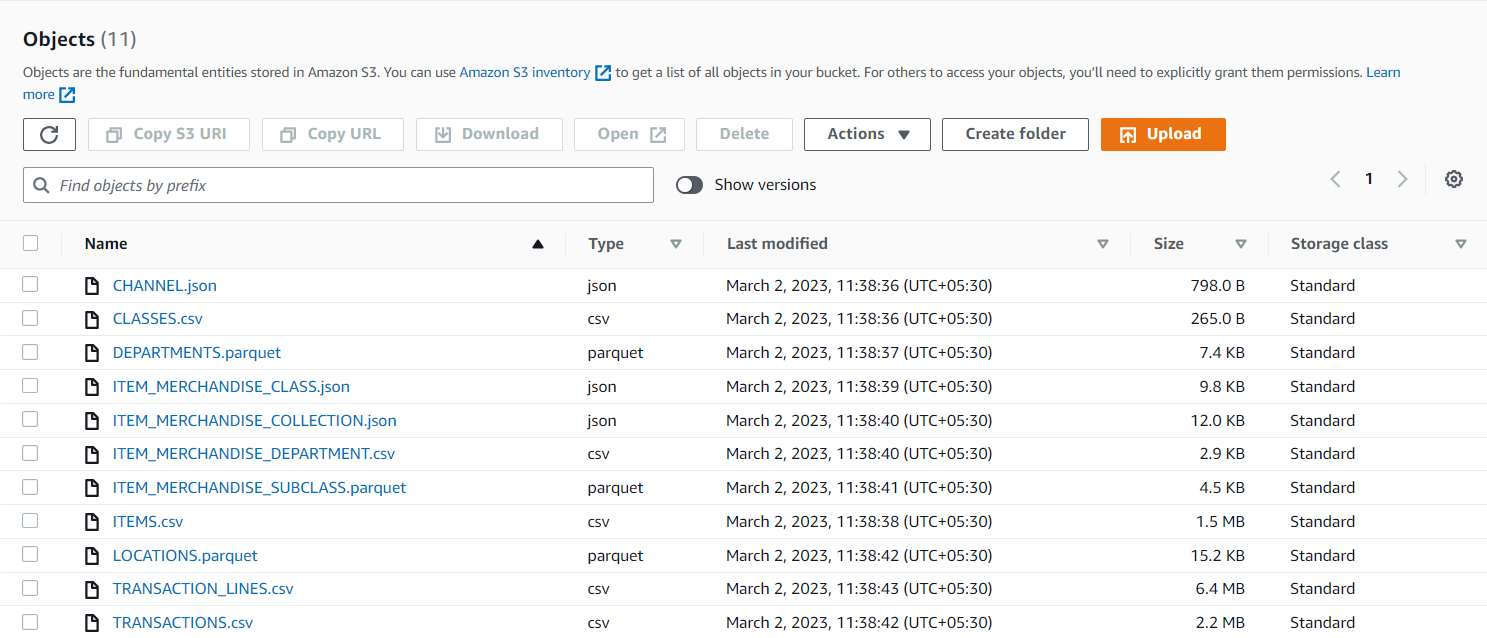
**Code File:**



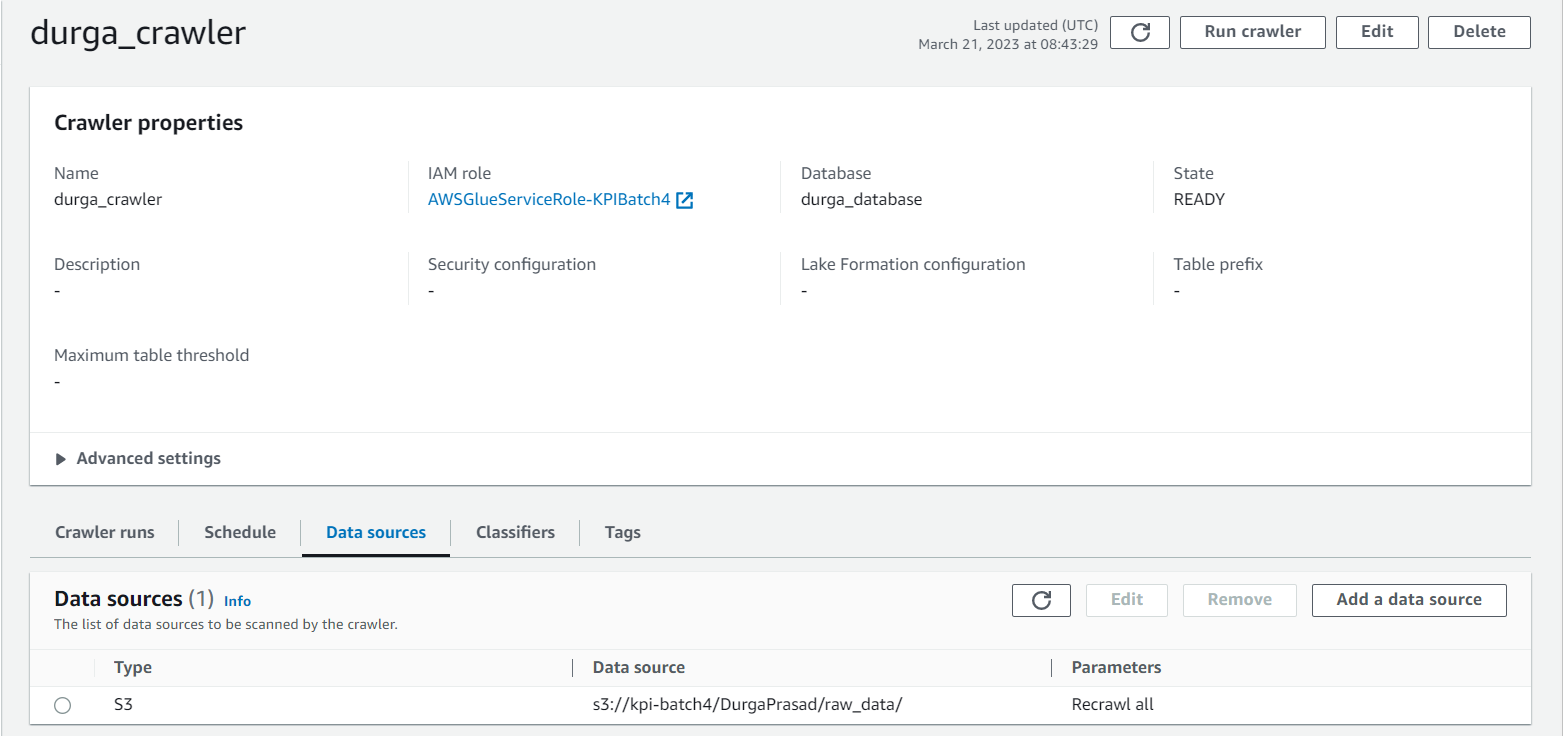
****

After running the current job we ca see the number of files loaded into s3 location

S3 Location: s3://kpi-batch4/DurgaPrasad/stagefolder/



**Creating stage table using AWS Glue Crawler:**

****

**Task 3: Stage layer to serve layer**

**To do things in serve layer**

1. Create DDL for target tables in the warehouse following the Target Model tab.

2. Load data into target tables by applying the below transformations on stage tables/files.

    1. Surrogate key "KPI\_DW\_SKEY" needs to be added in the dimension table.

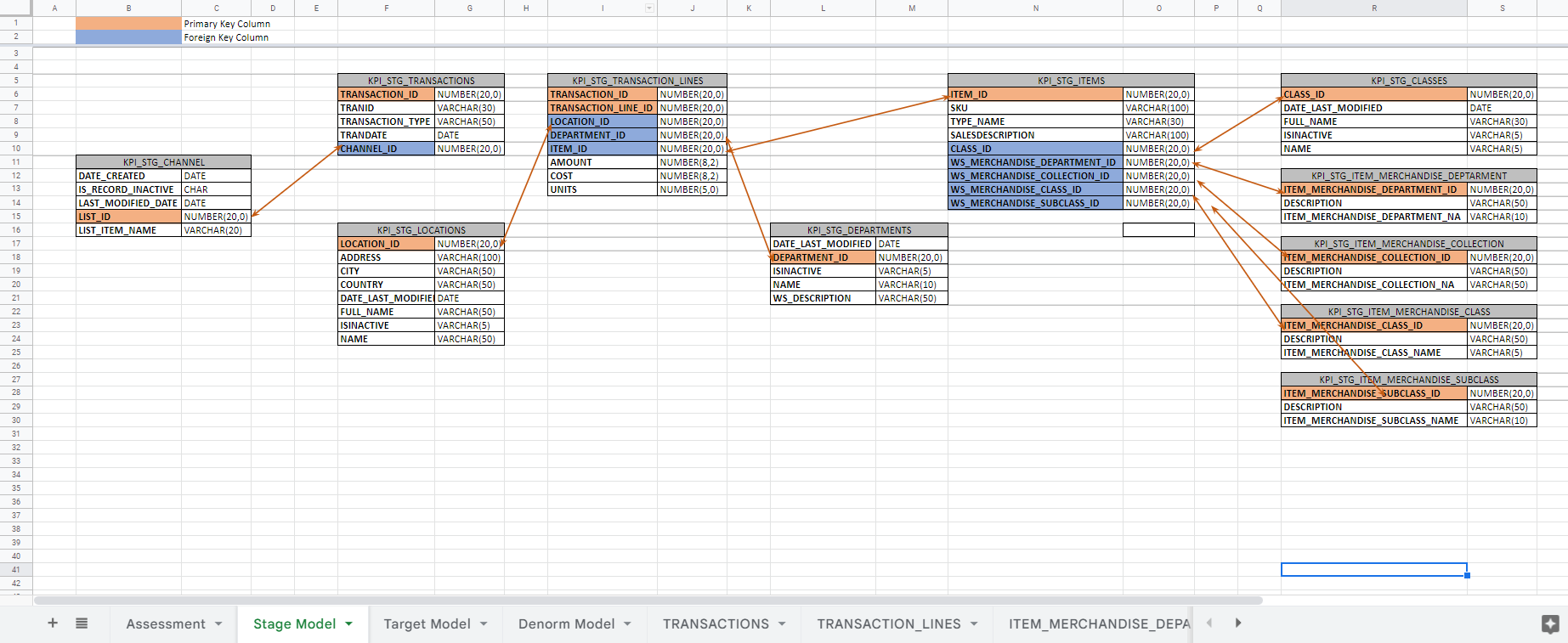
    2. Add audit columns like process\_date to all the tables.

    3. Add an additional column to the item table "brand\_name". Data is the same as the "name" column.

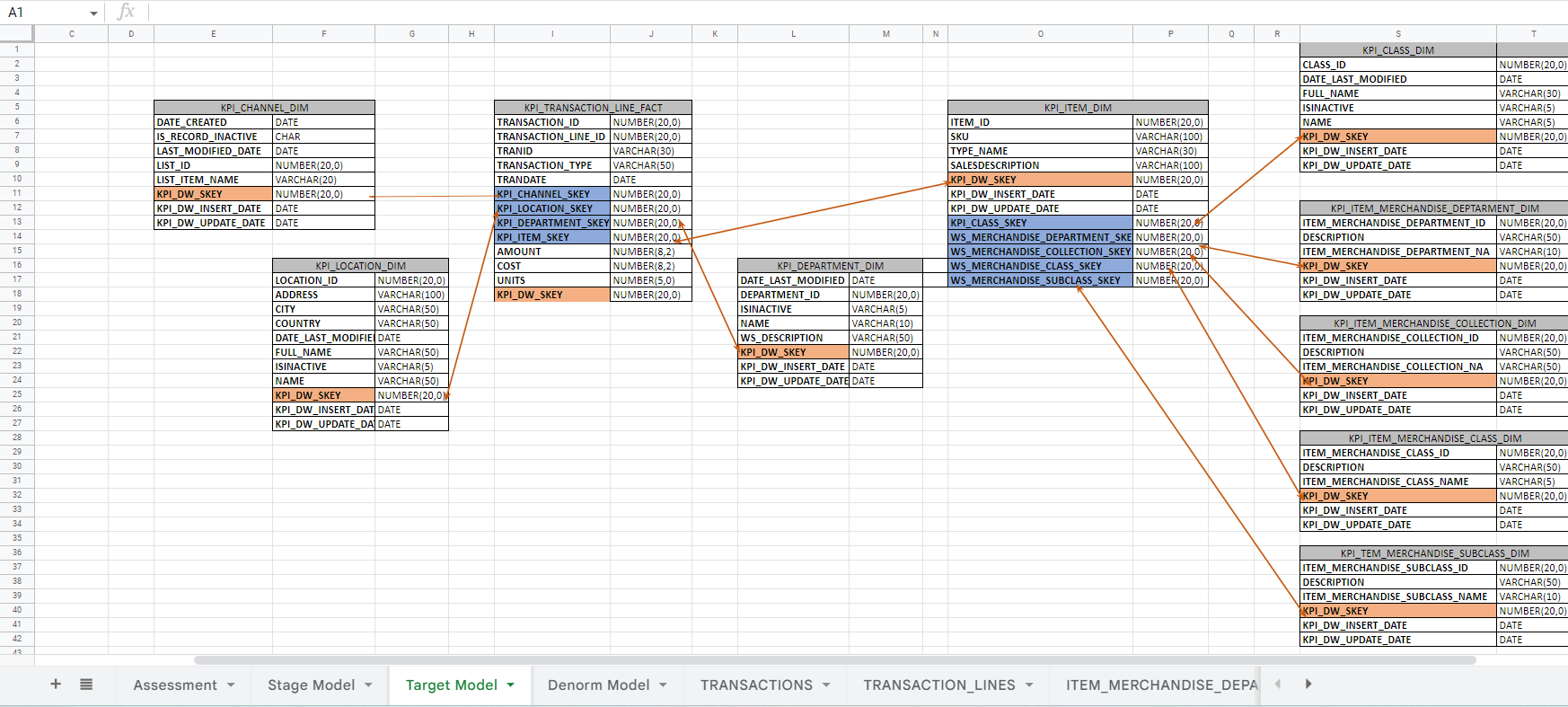
    4. Identify the relationships between each table and add foreign keys to the fact tables.

Before creating the DDL statements in snowflake we need to do some transformations before loading into the snowflake warehouse.

**Stage Layer:**

****

**Target layer:**

****

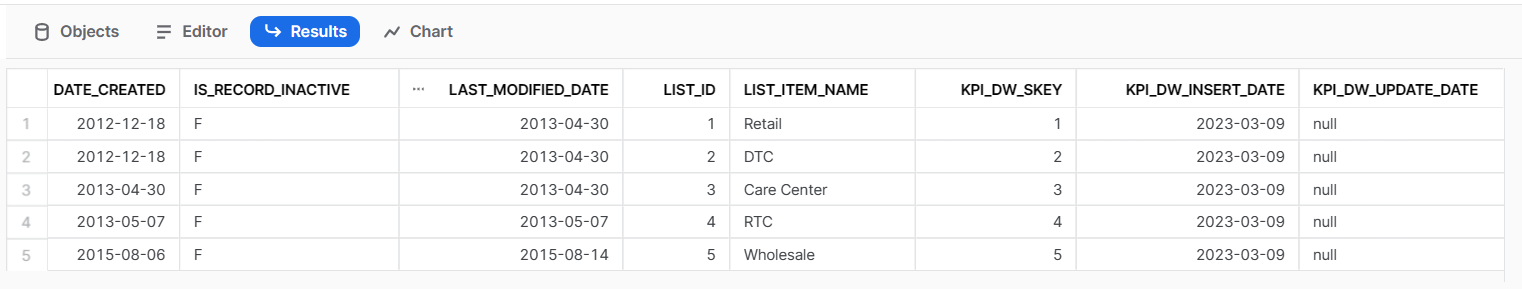
**Code File:**



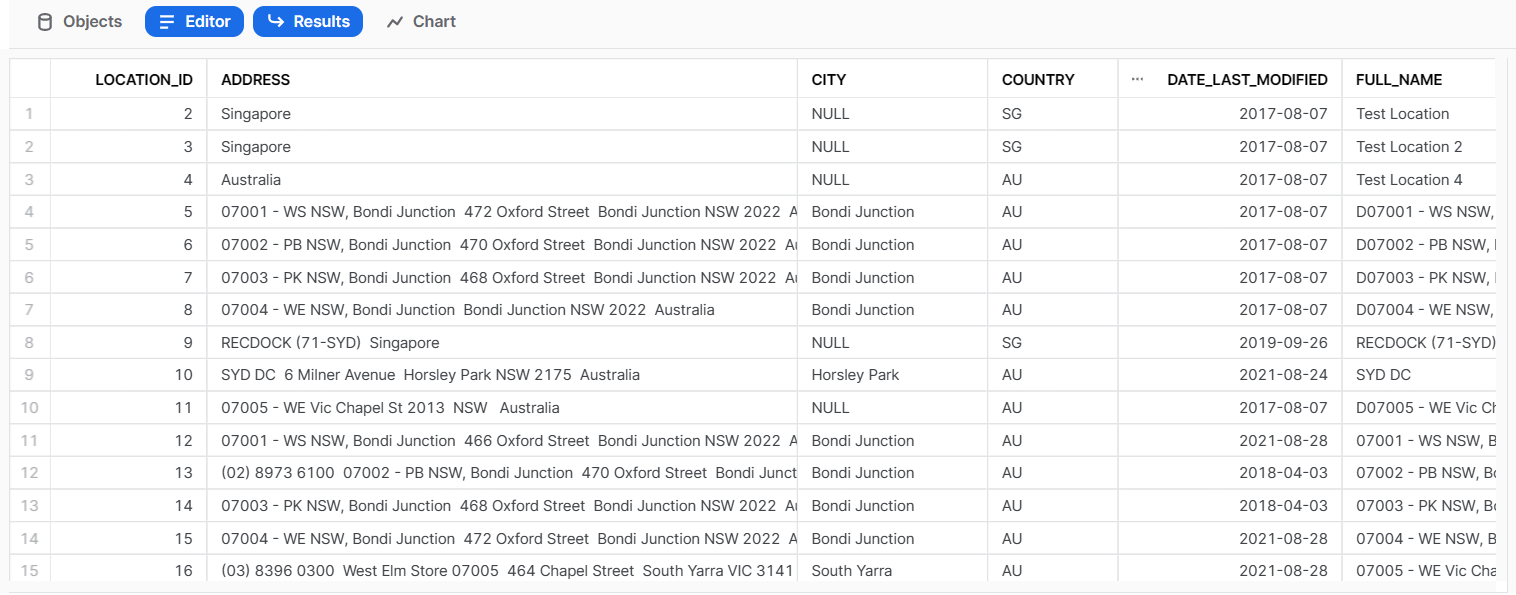
After running the current job we ca see the number of files loaded into our target snowflake database.

Snips of following table’s data in snowflake after running target glue job

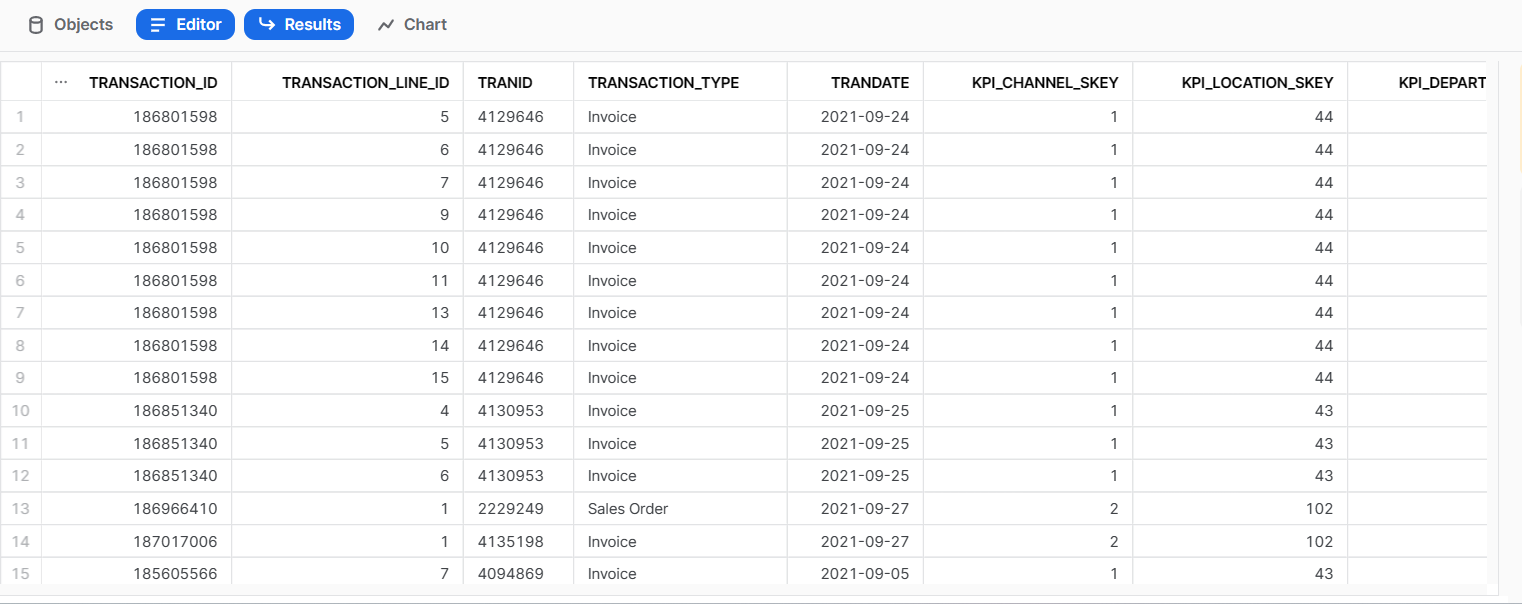
**KPI\_CHANNEL\_DIM**



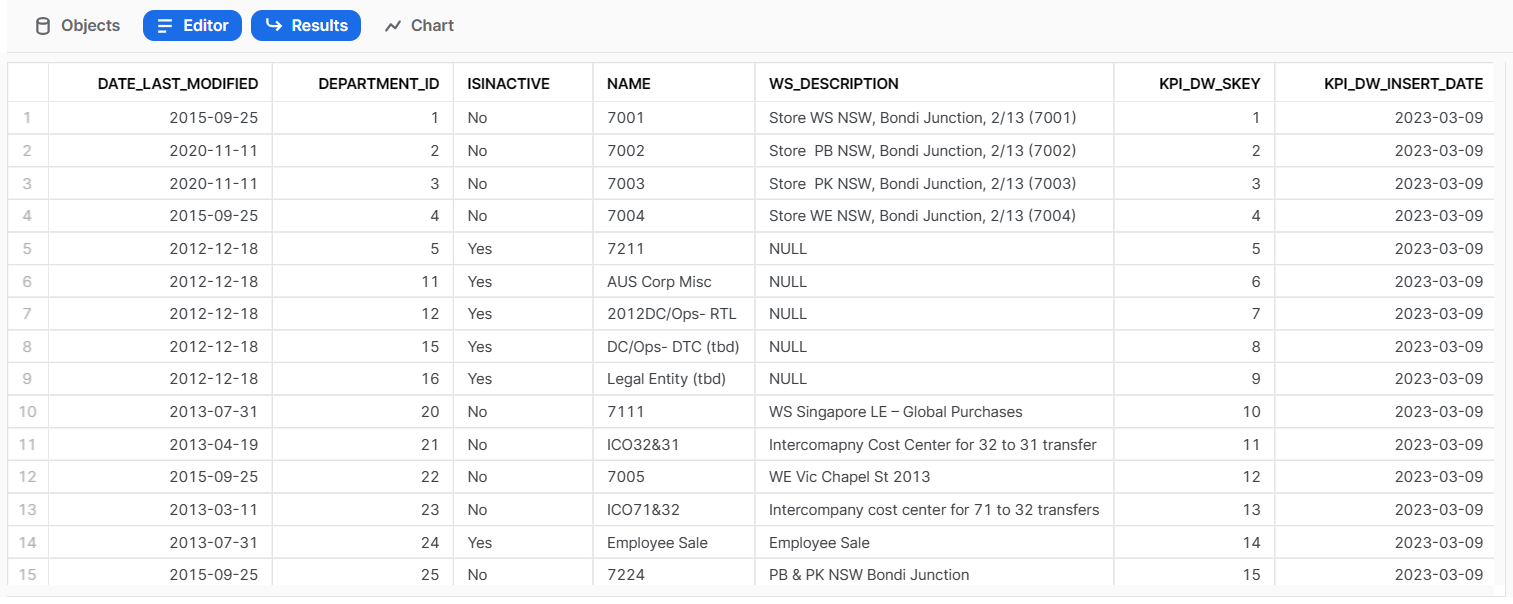
**KPI\_LOCATION\_DIM**



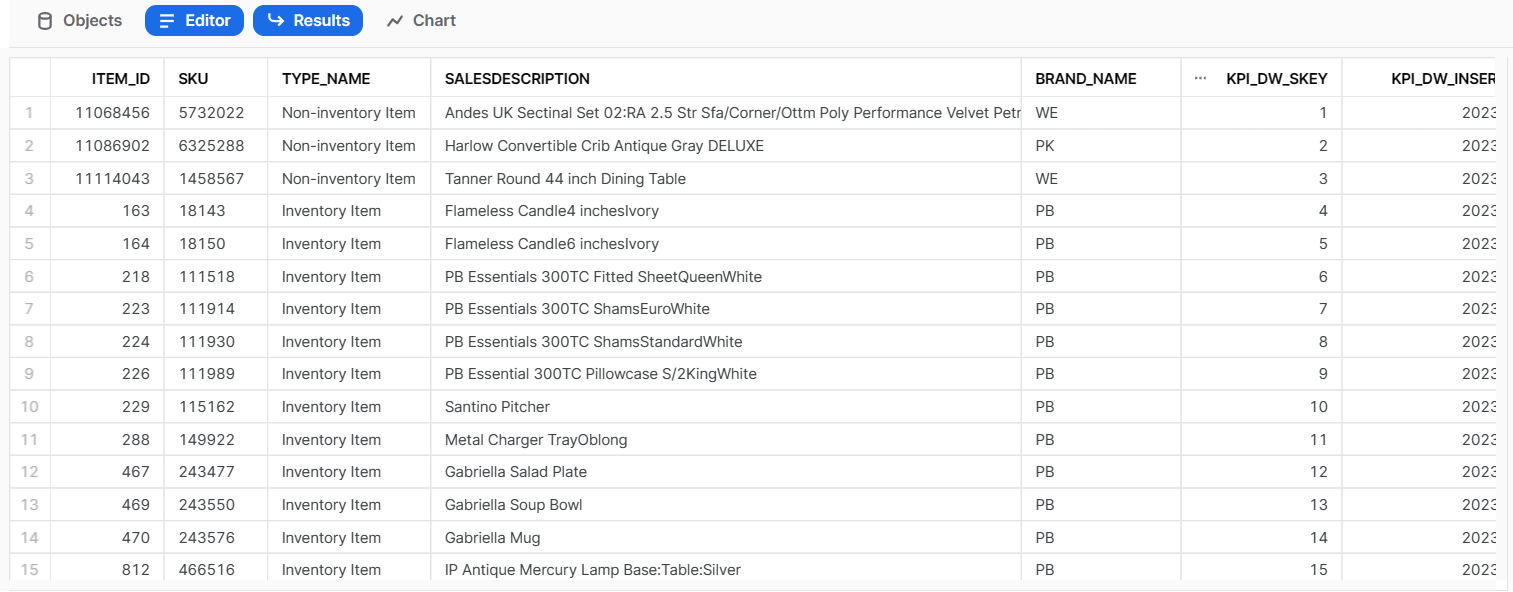
**KPI\_TRANSACTION\_LINE\_FACT**

****

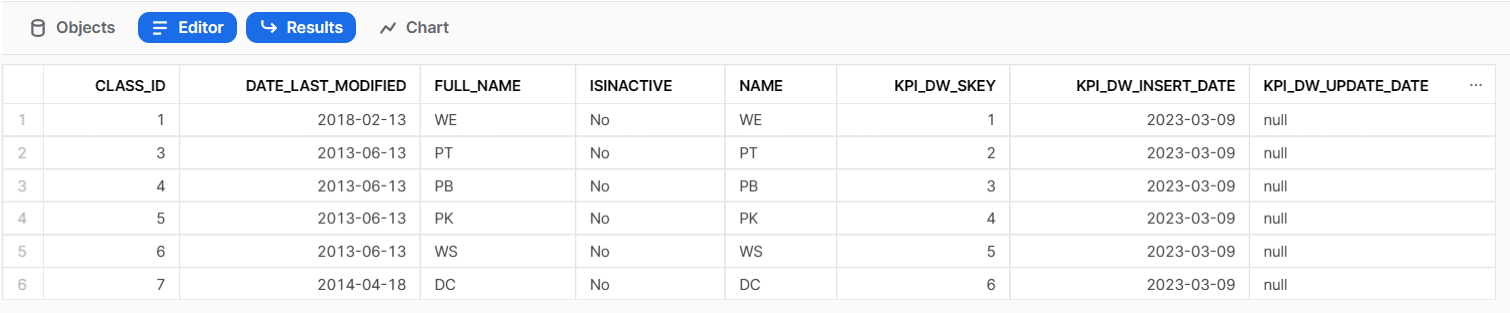
**KPI\_DEPARTMENT\_DIM**

****

**KPI\_ITEM\_DIM**

****

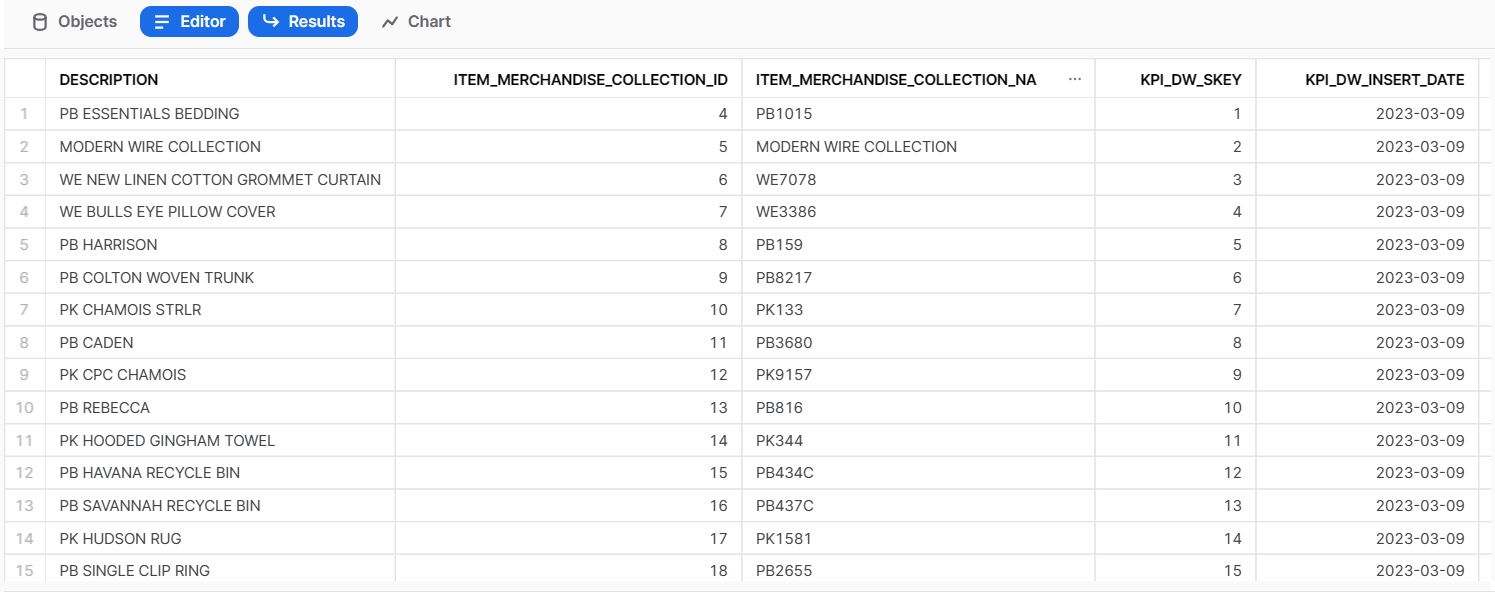
**KPI\_CLASS\_DIM**

****

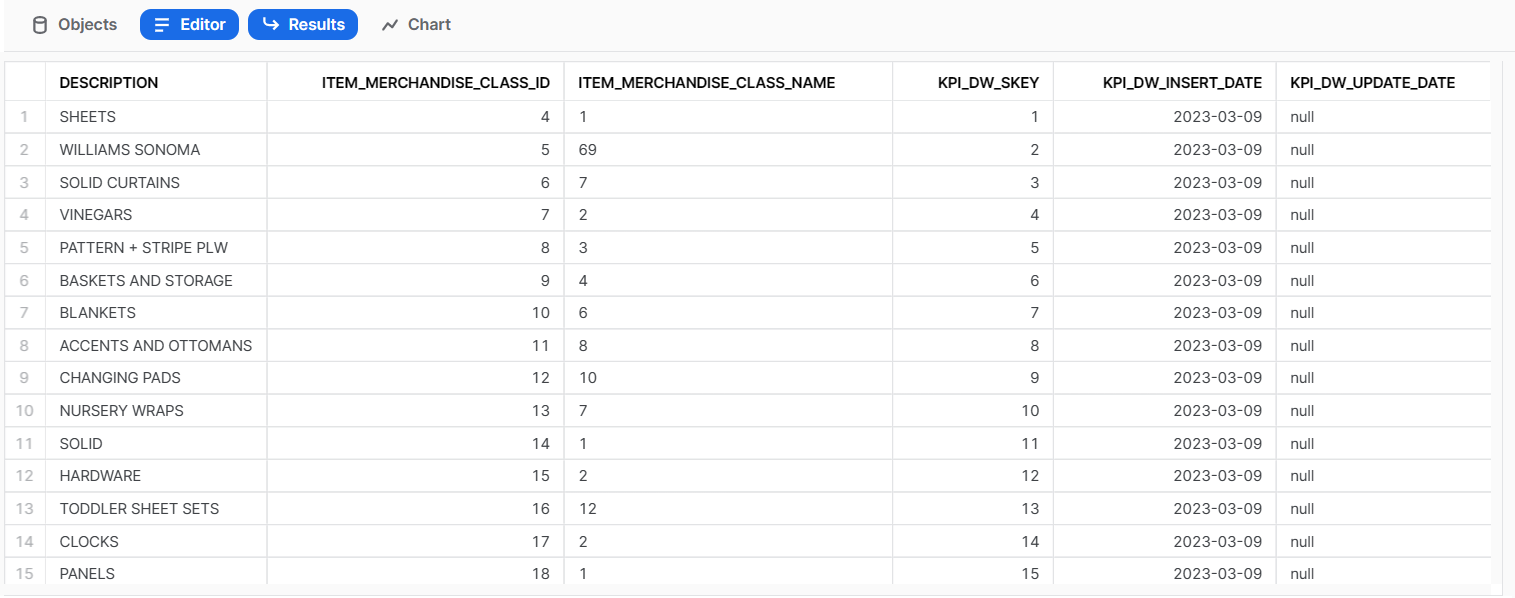
**KPI\_ITEM\_MERCHANDISE\_DEPTARMENT\_DIM**

****

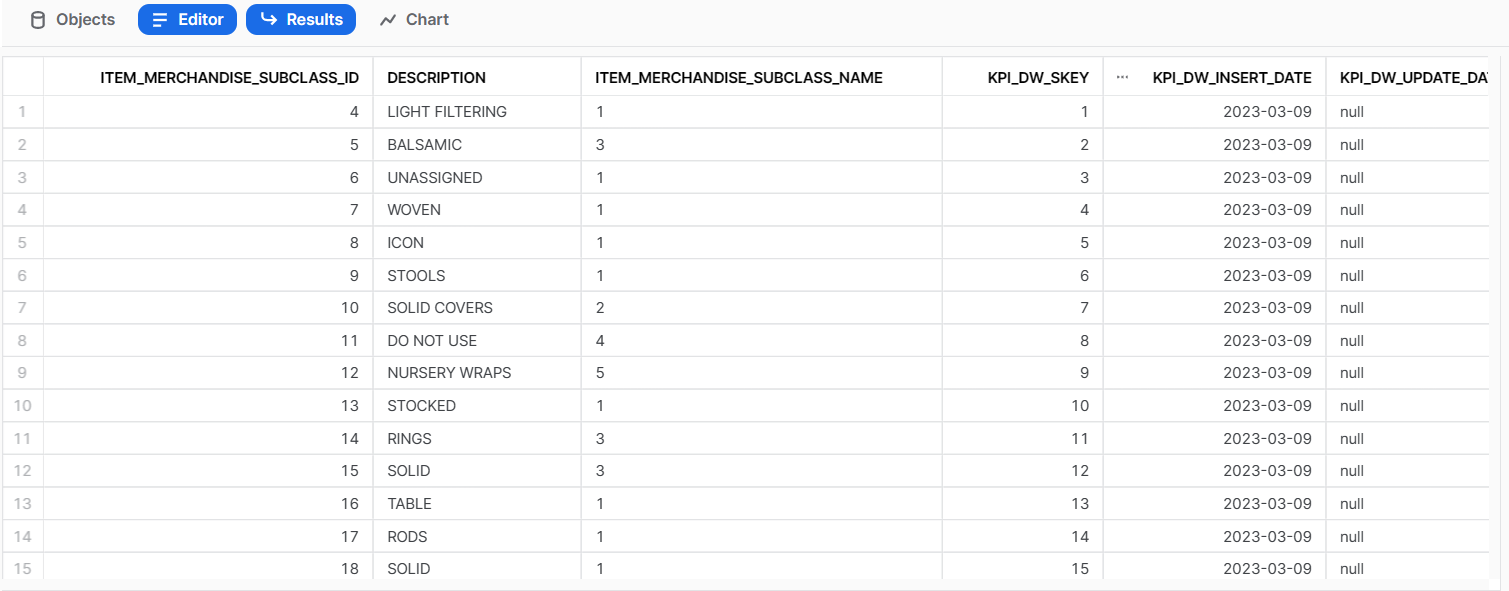
**KPI\_ITEM\_MERCHANDISE\_COLLECTION\_DIM**

****

**KPI\_ITEM\_MERCHANDISE\_CLASS\_DIM**

****

**KPI\_TEM\_MERCHANDISE\_SUBCLASS\_DIM**

****

**Operations in snowflake database:**

**Relationship of primary key and foreign key in database**



**Query tasks performed in snowflake database**

1. **Find the Top 5 and Bottom 5 Items based on the Demand Amount values in a single query**

SELECT KPI\_ITEM\_SKEY AS TOP\_ITEM, SUM (AMOUNT)

FROM KPI\_TRANSACTION\_LINE\_FACT

WHERE TRANSACTION\_TYPE='Sales Order'

GROUP BY KPI\_ITEM\_SKEY

HAVING SUM (AMOUNT) IS NOT NULL

ORDER BY 2 DESC

SELECT KPI\_ITEM\_SKEY AS BOTTOM\_ITEM, SUM (AMOUNT)

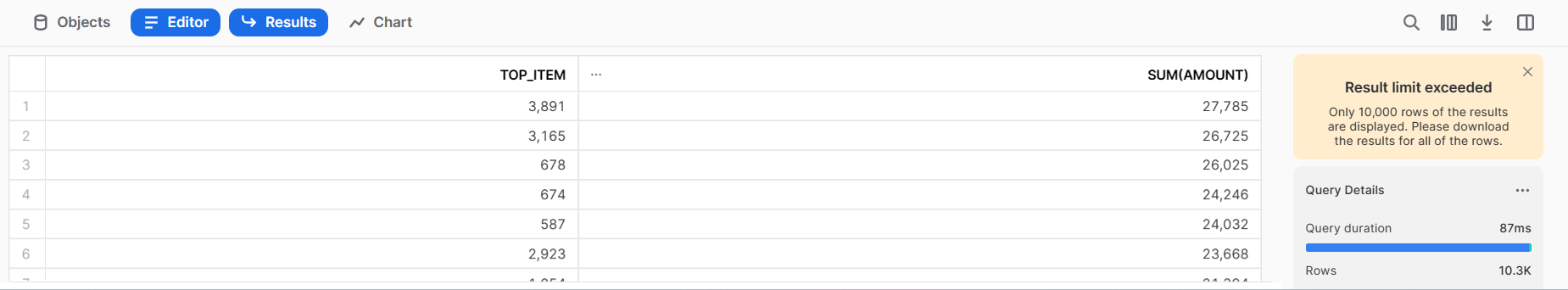
FROM KPI\_TRANSACTION\_LINE\_FACT

WHERE TRANSACTION\_TYPE='Sales Order'

GROUP BY KPI\_ITEM\_SKEY

HAVING SUM (AMOUNT) IS NOT NULL

ORDER BY 2

****

1. **Which Department has the highest Demand and Sales Amount**

SELECT T2.NAME, SUM (AMOUNT) AS HIGHEST\_DEMAND\_AMOUNT

FROM KPI\_TRANSACTION\_LINE\_FACT T1 INNER JOIN KPI\_DEPARTMENT\_DIM T2

ON T1.KPI\_DEPARTMENT\_SKEY = T2.KPI\_DW\_SKEY

WHERE TRANSACTION\_TYPE='Sales Order'

GROUP BY T2.NAME

ORDER BY 2 DESC

LIMIT 1;

SELECT T2.NAME, SUM (AMOUNT) AS HIGHEST\_SALES\_AMOUNT

FROM KPI\_TRANSACTION\_LINE\_FACT T1 INNER JOIN KPI\_DEPARTMENT\_DIM T2

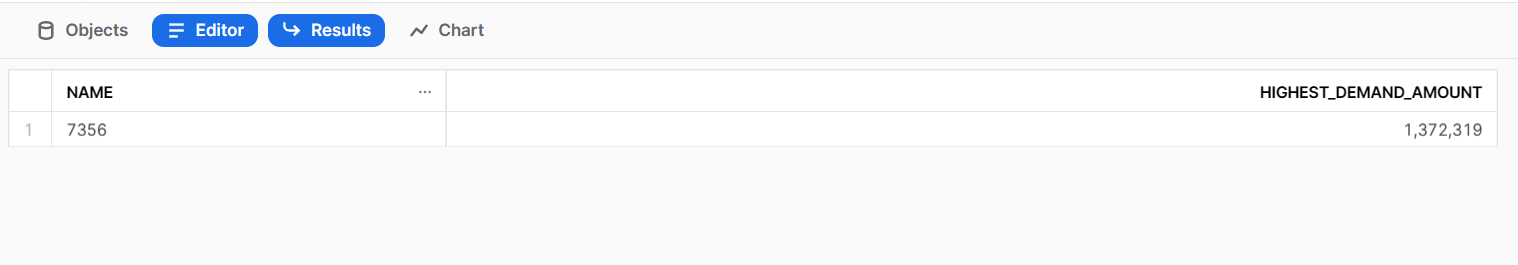
ON T1.KPI\_DEPARTMENT\_SKEY = T2.KPI\_DW\_SKEY

WHERE TRANSACTION\_TYPE='Invoice'

GROUP BY T2.NAME

ORDER BY 2 DESC

LIMIT 1;

****

1. **Find the Demand Profit and Sales Profit for each ITEM. Populate ITEM.NAME**

SELECT T2.BRAND\_NAME,

TRANSACTION\_TYPE,

CASE

WHEN TRANSACTION\_TYPE='Sales Order' THEN ROUND (((SUM (AMOUNT) +SUM (COST))/SUM (AMOUNT))\*100)

ELSE 0

END AS Demand\_Profit,

CASE

WHEN TRANSACTION\_TYPE='Invoice' THEN ROUND (((SUM (AMOUNT) +SUM (COST))/SUM (AMOUNT))\*100)

ELSE 0

END AS Sales\_Profit

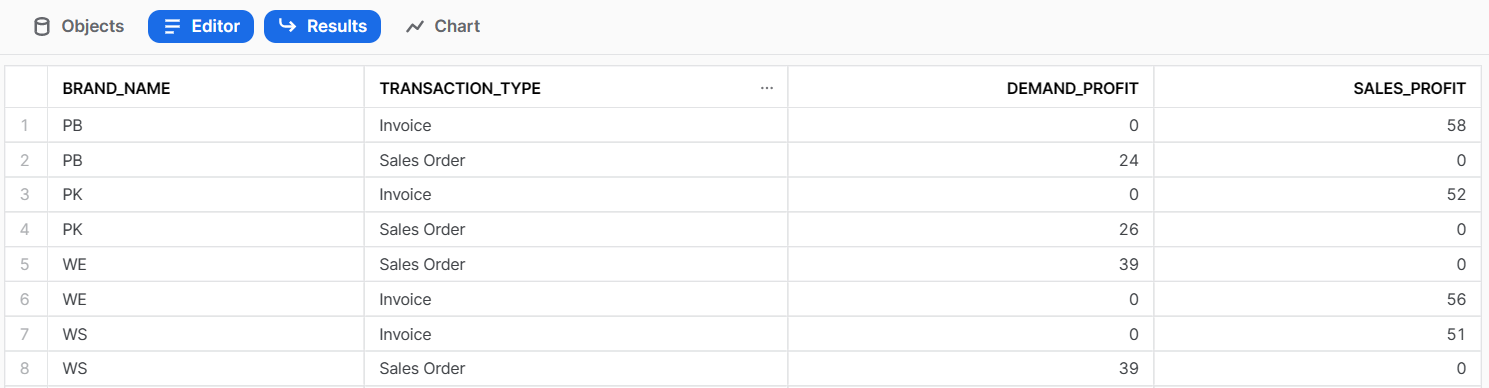
FROM KPI\_TRANSACTION\_LINE\_FACT T1 INNER JOIN KPI\_ITEM\_DIM T2

ON T1.KPI\_ITEM\_SKEY = T2.KPI\_DW\_SKEY

GROUP BY T2.BRAND\_NAME, TRANSACTION\_TYPE

HAVING SUM (AMOUNT)>0

ORDER BY T2.BRAND\_NAME;

****

1. **Populate top 10 LOCATIONS based on number of Demand Transactions using Analytical functions**

SELECT T2.FULL\_NAME AS LOCATION,

COUNT (\*)

FROM KPI\_TRANSACTION\_LINE\_FACT T1 INNER JOIN KPI\_LOCATION\_DIM T2

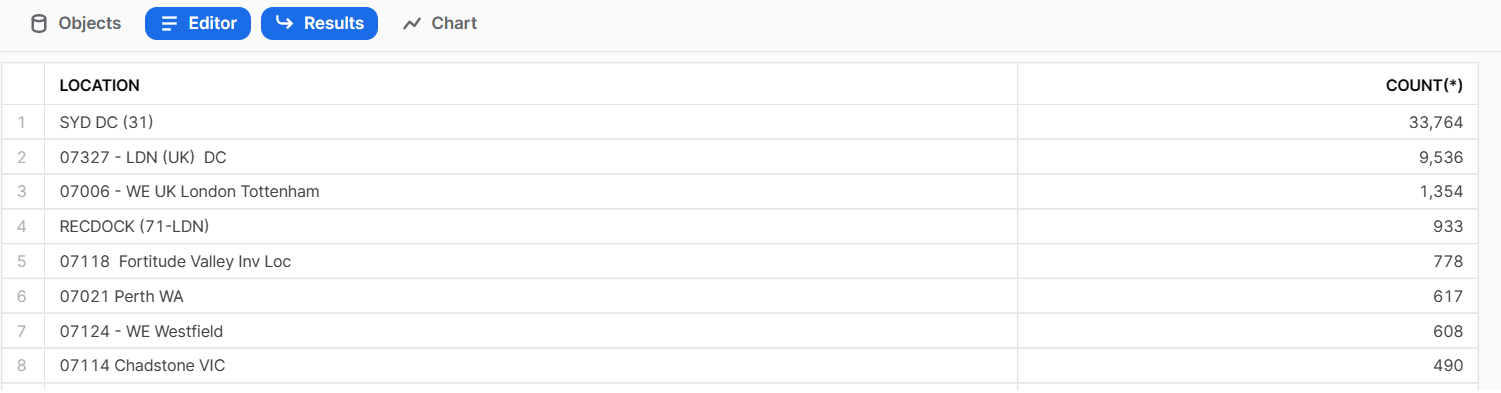
ON T1.KPI\_LOCATION\_SKEY = T2.KPI\_DW\_SKEY

WHERE T1.TRANSACTION\_TYPE='Sales Order'

GROUP BY T2.FULL\_NAME

ORDER BY 2 DESC

LIMIT 10;

****

1. **Find Demand Amount, Demand Units, Sales Amount and Sales Units for each Channel**

SELECT KPI\_CHANNEL\_SKEY,

TRANSACTION\_TYPE,

CASE

WHEN TRANSACTION\_TYPE='Sales Order' THEN SUM (AMOUNT)

ELSE 0

END AS Demand\_Amount,

CASE

WHEN TRANSACTION\_TYPE='Sales Order' THEN SUM (UNITS)

ELSE 0

END AS Demand\_Units,

CASE

WHEN TRANSACTION\_TYPE='Invoice' THEN SUM (AMOUNT)

ELSE 0

END AS Sales\_Amount,

CASE

WHEN TRANSACTION\_TYPE='Invoice' THEN SUM (UNITS)

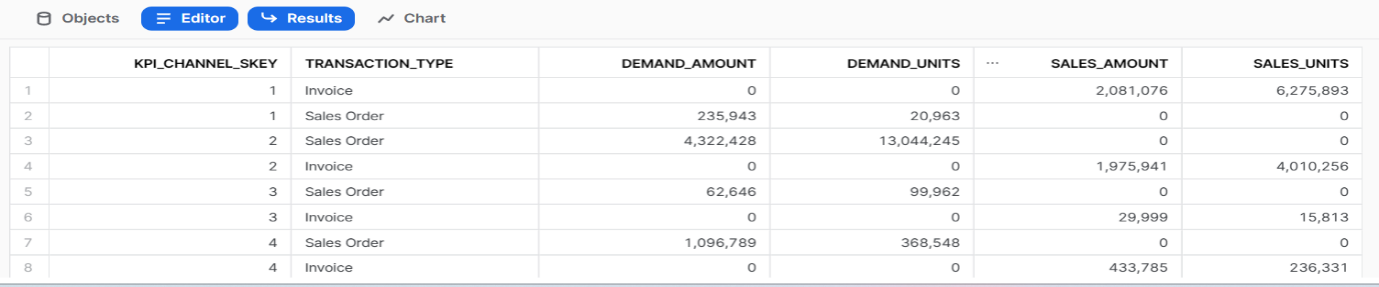
ELSE 0

END AS Sales\_Units

FROM KPI\_TRANSACTION\_LINE\_FACT

GROUP BY KPI\_CHANNEL\_SKEY, TRANSACTION\_TYPE

ORDER BY KPI\_CHANNEL\_SKEY;

****

**Task 4: Target table to denorm table**

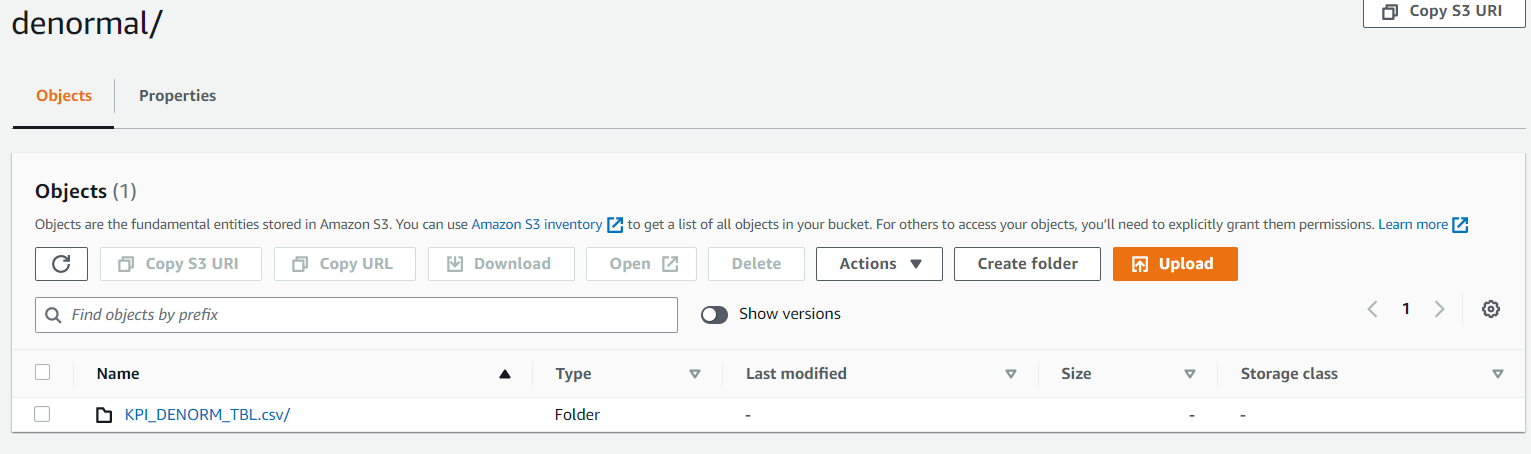
Now we have to create denormal table including all 10 tables we have to create one table with few column mentioned in the internal\_data we have to perform many joins operation like this given in the script below:

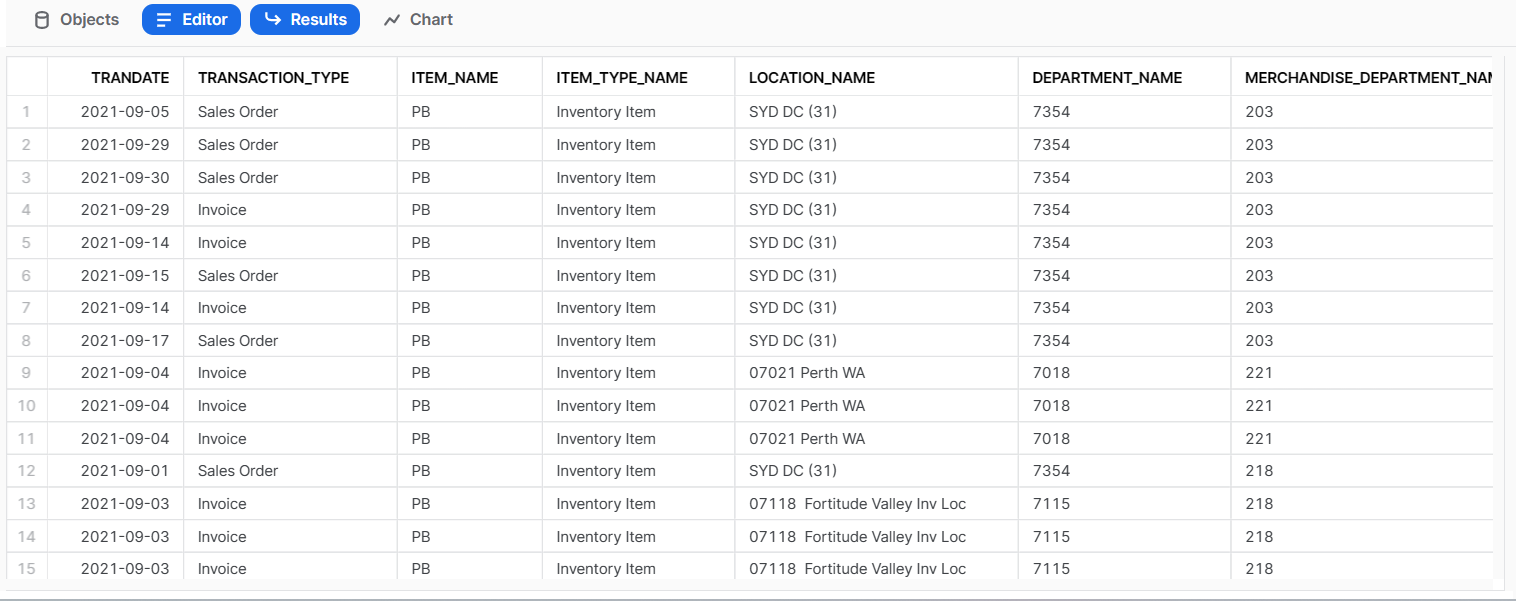
**Code file:**



After running the current job we ca see the data loaded into s3 location and snowflake DB

S3 Location: s3://kpi-batch4/DurgaPrasad/denormal/

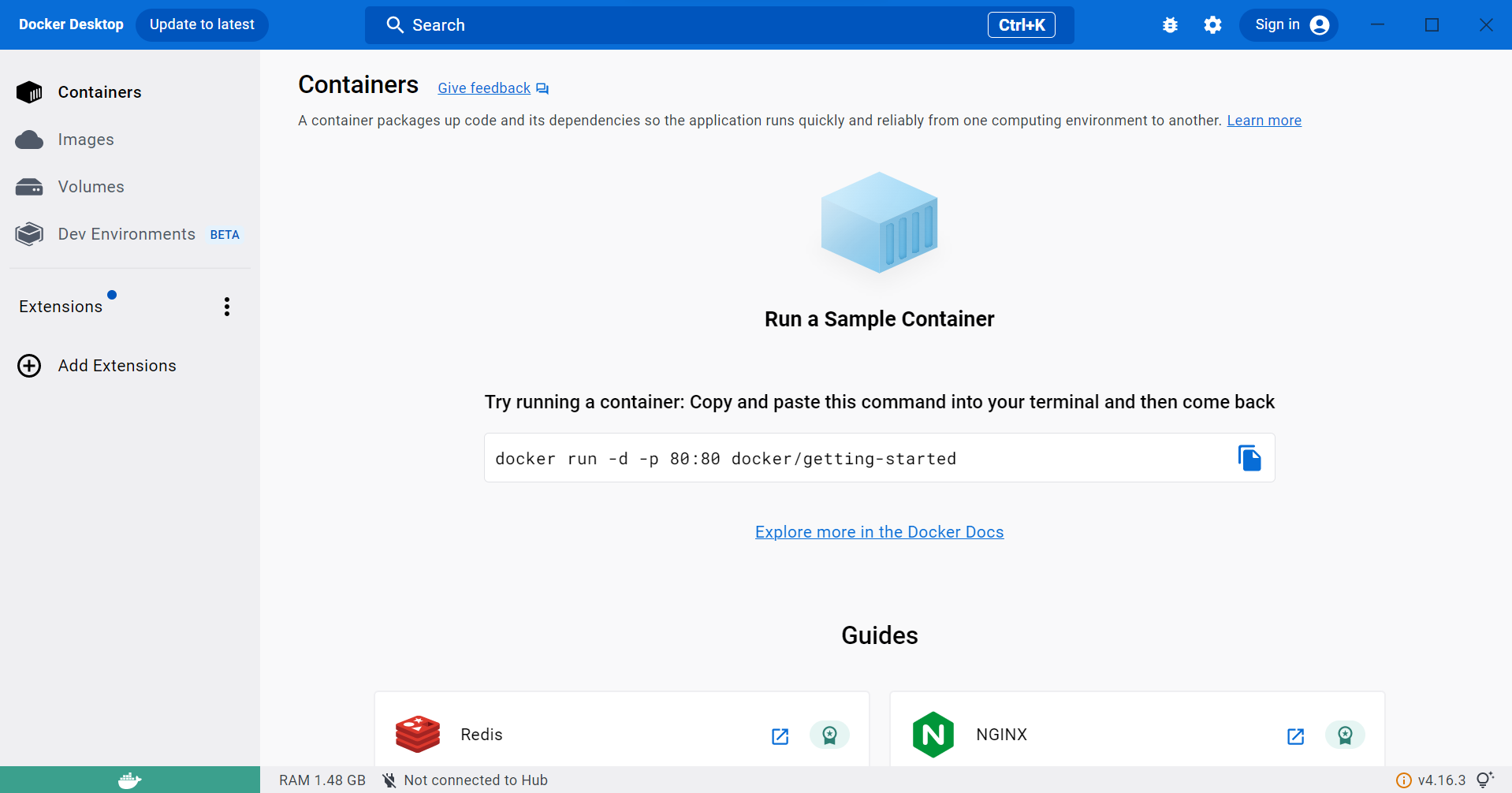


**KPI\_DENORM\_TBL**

**Task 5: Airflow Orchestration**

We run Airflow in our local system for that we use Docker

Download and install Docker desktop

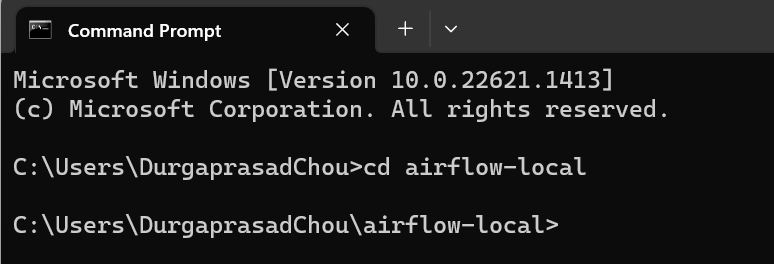


Open Cmd and Create a directory named Airflow-local

Command: mkdir airflow-local

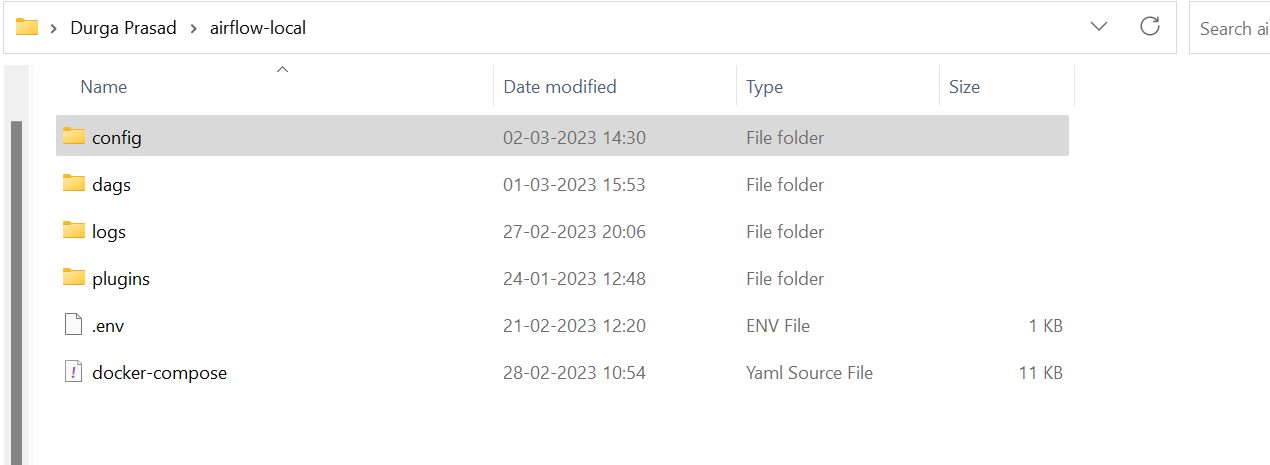
Then open airflow-local directory from cmd

Command: cd airflow-local



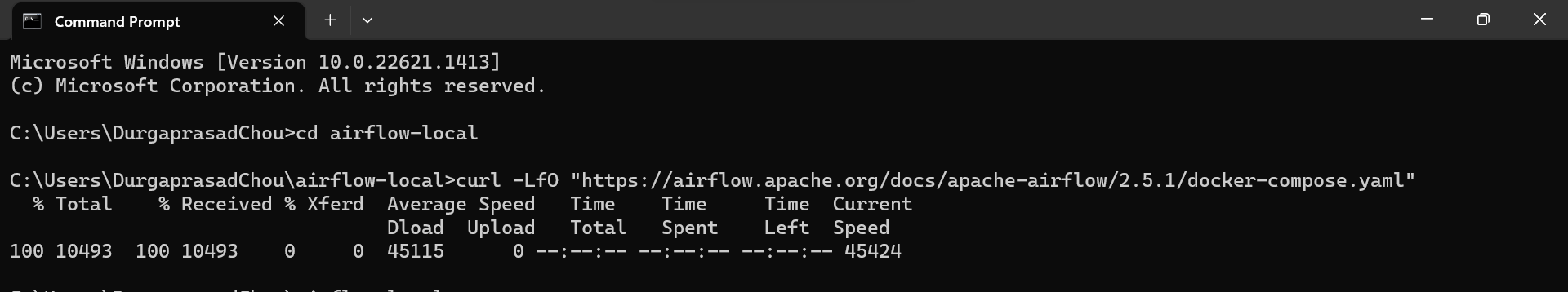
**C: Drive**>>**Users**>> <**YourName**> >> **airflow-local**

Create dags, plugins, logs and config folders inside airflow-local directory



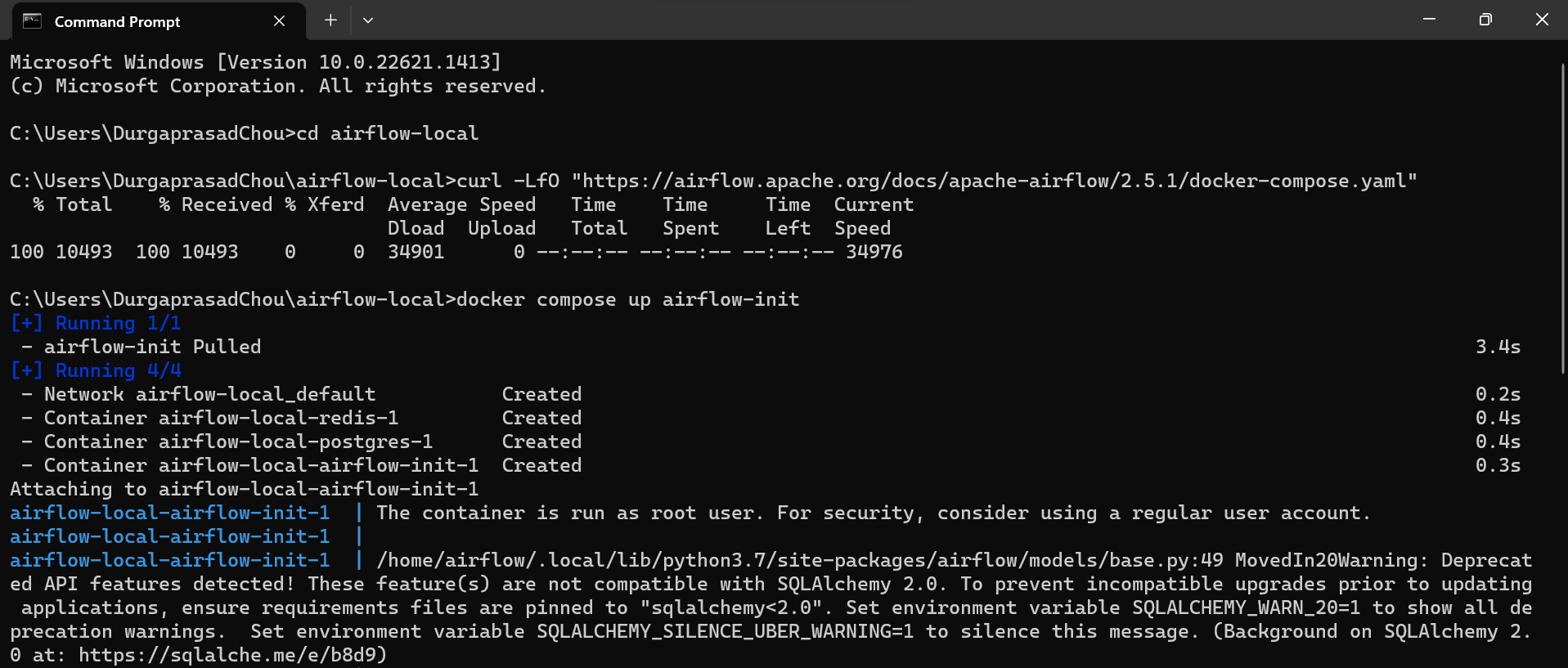
Run the following command to get the airflow ymal file.

Cmd: curl -LfO “https://airflow.apache.org/docs/apache-airflow/2.5.1/docker-compose.yaml”



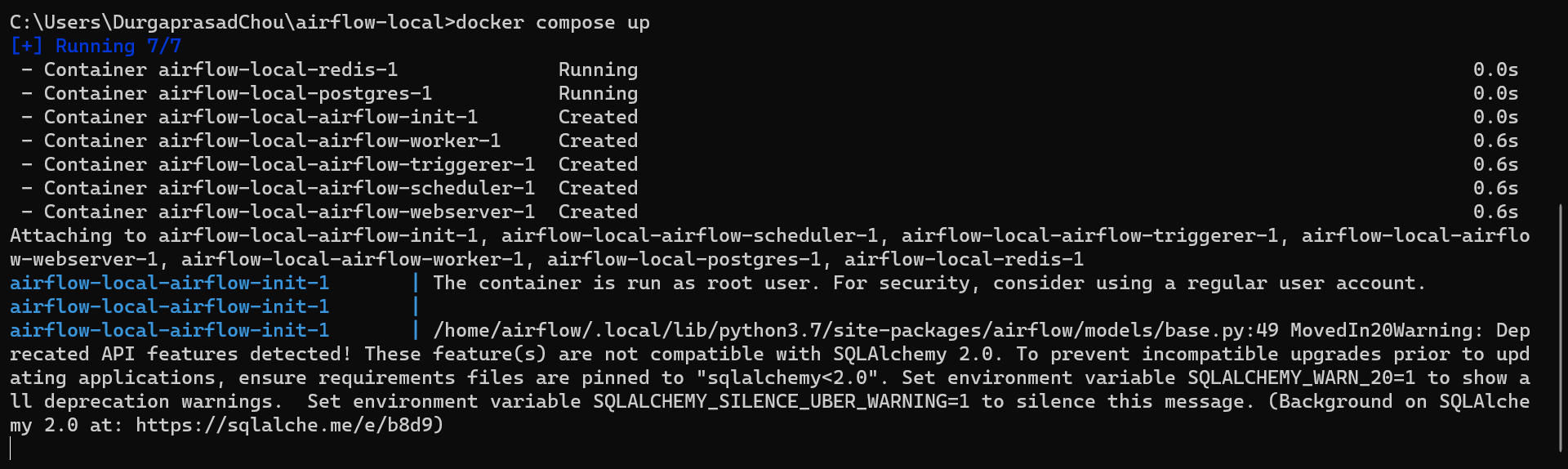
Run the command to download the Docker images for airflow.

Cmd: docker compose up airflow-init

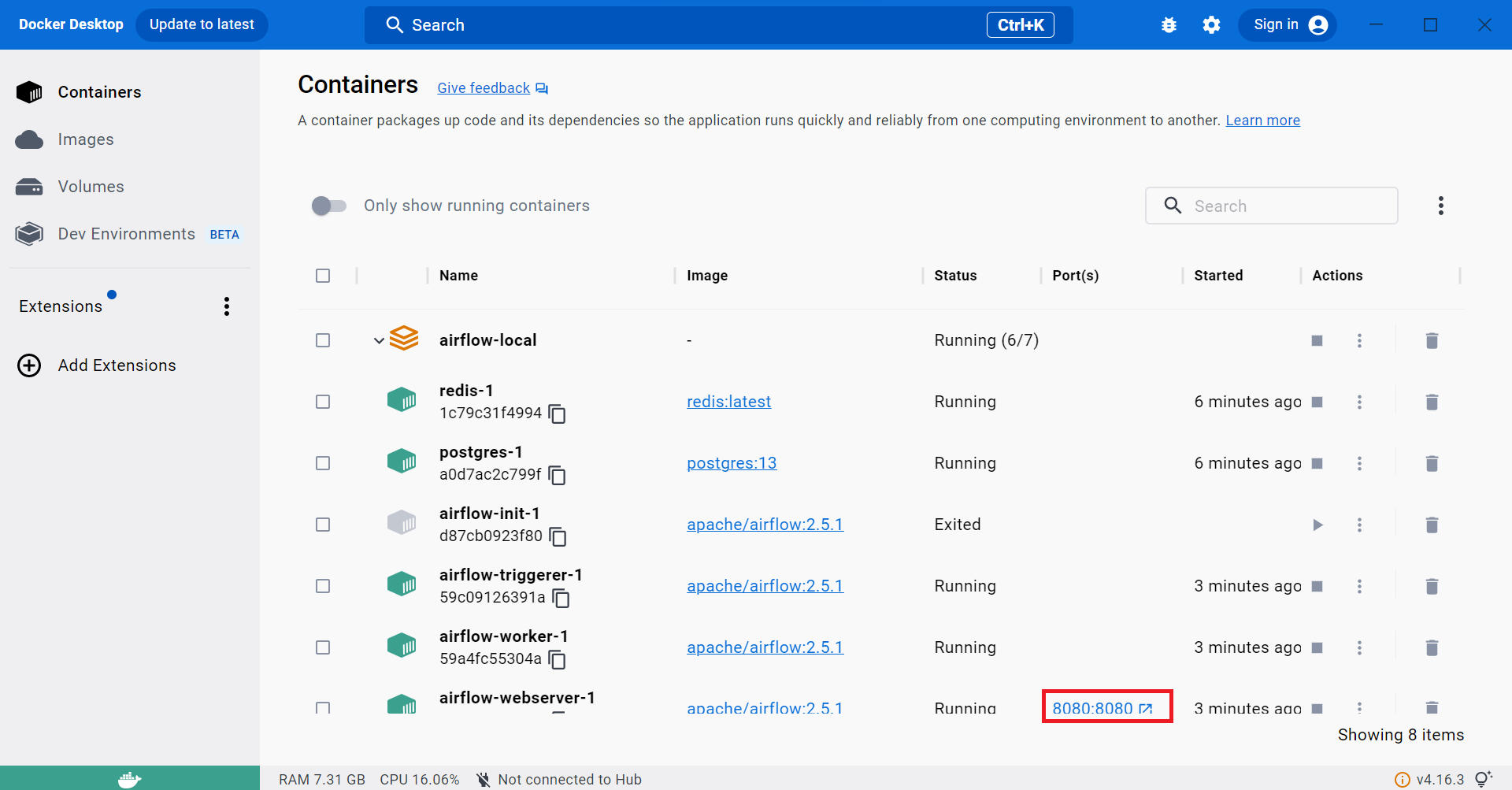


Run the following command to start the airflow in Docker

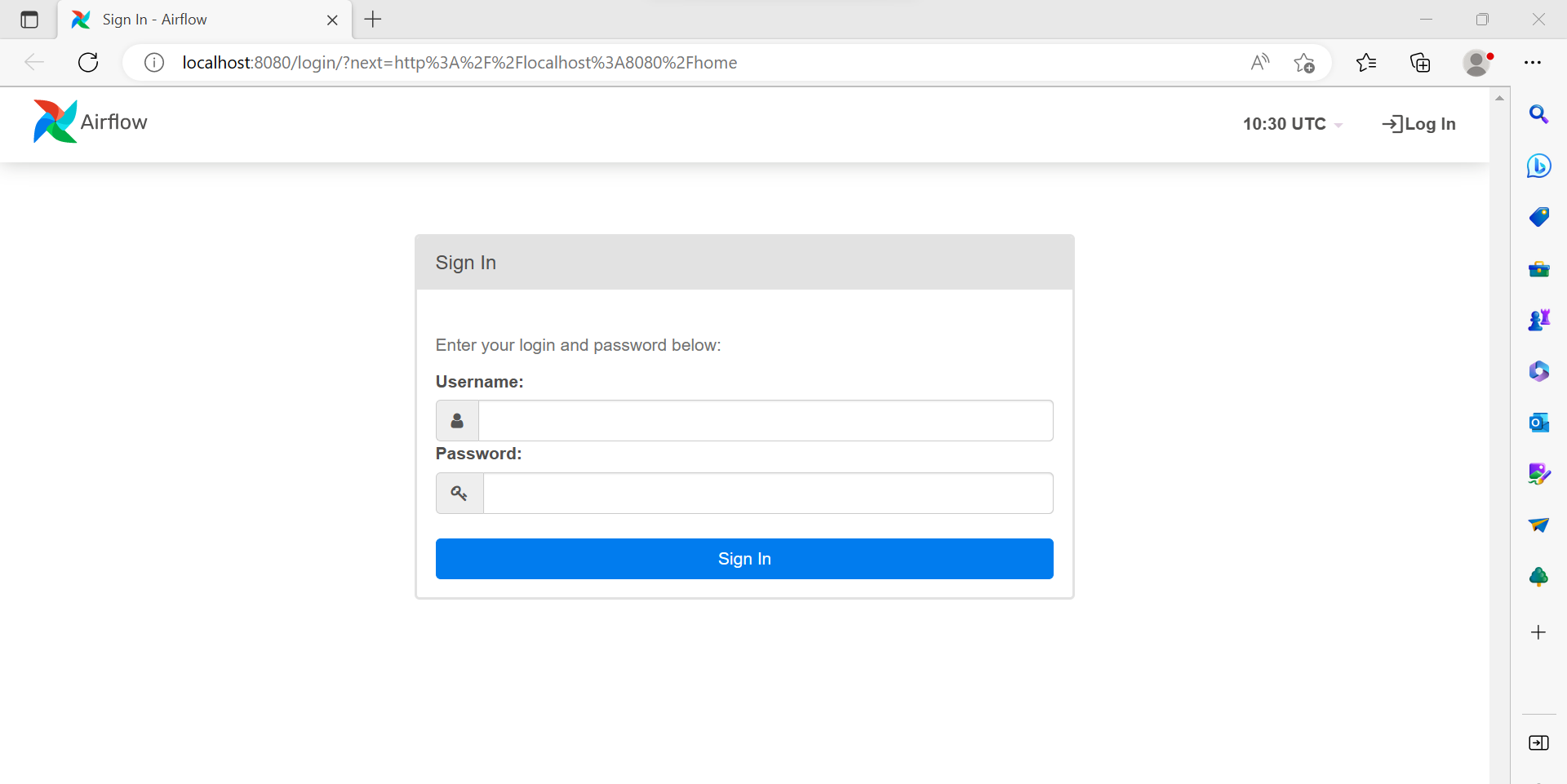
Cmd: docker compose up



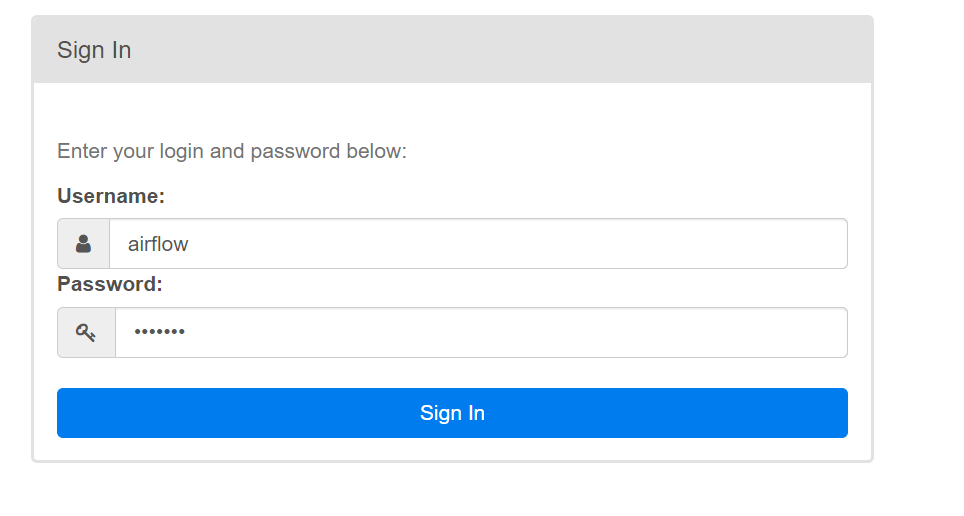
Check the Docker desktop containers whether airflow-local services are running



Open <http://localhost:8080/> port for webUI



Username and password is default airflow login with it

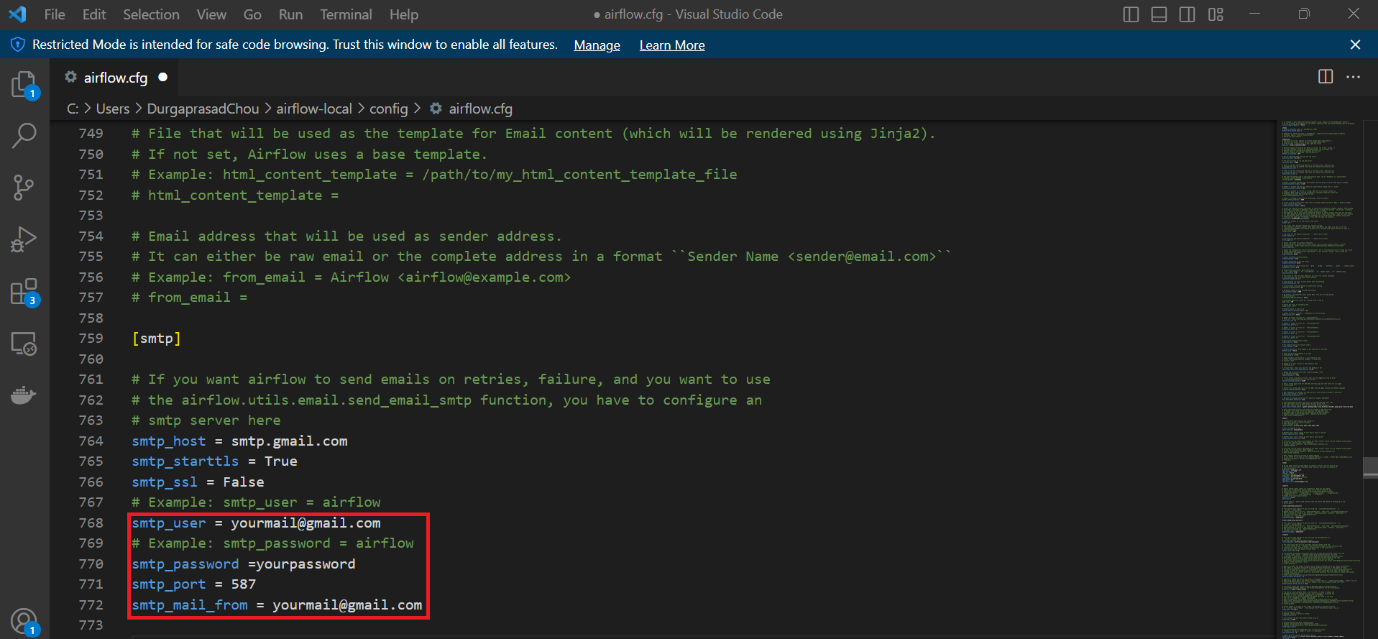


Under **config file** >> **airflow.cfg**, click on it and search for **SMTP** connection, change here: **smtp\_host=smtp.gmail.com**

**smtp\_user=<your\_email>**

**smtp\_password=<your\_password>**

Note: smtp\_password can be derived by enabling **two-step authentication** from your Gmail, and getting the **App Passwords** from that page itself

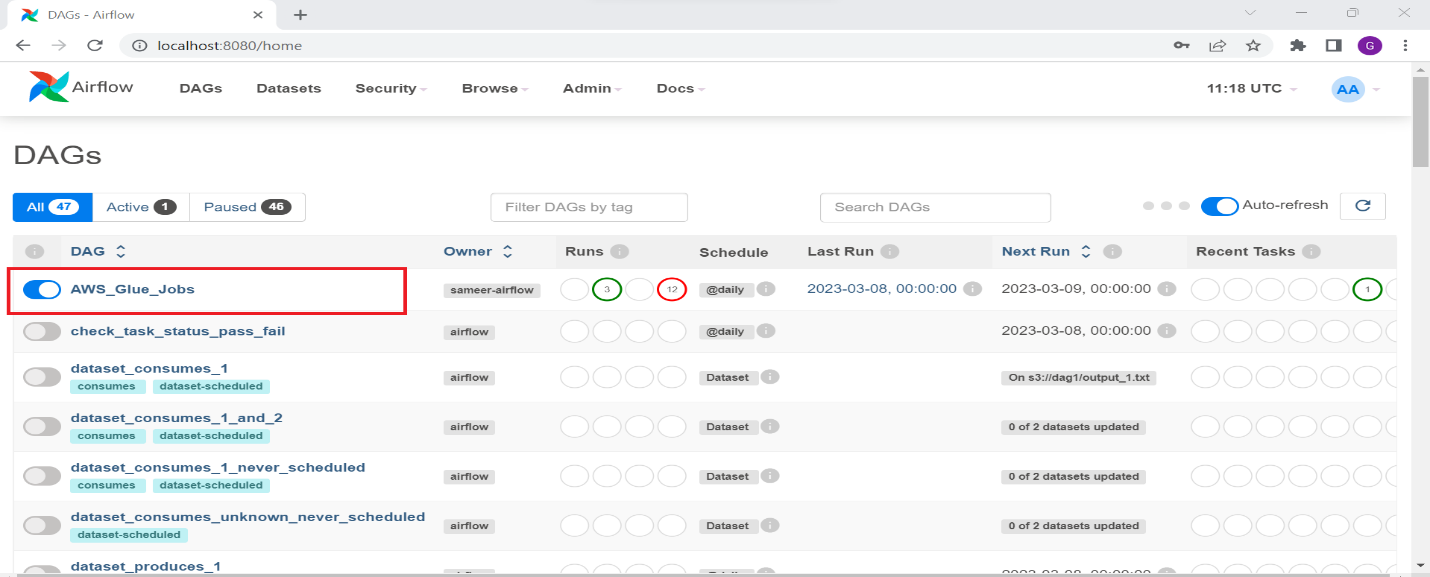


Under **dags** folder create a new file and write your desired dag code and save it a with **.py** extension.

**Dag File:**

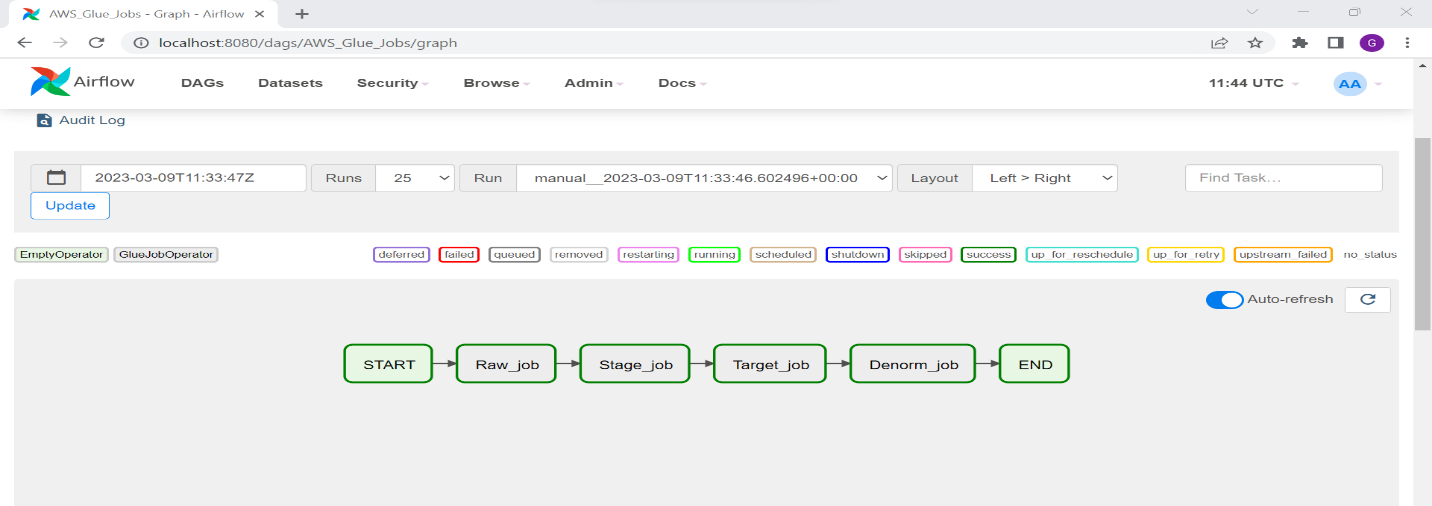


Now you can save the .py dag files in dags folder and it will reflect in airflow web UI and then you can run the dags.



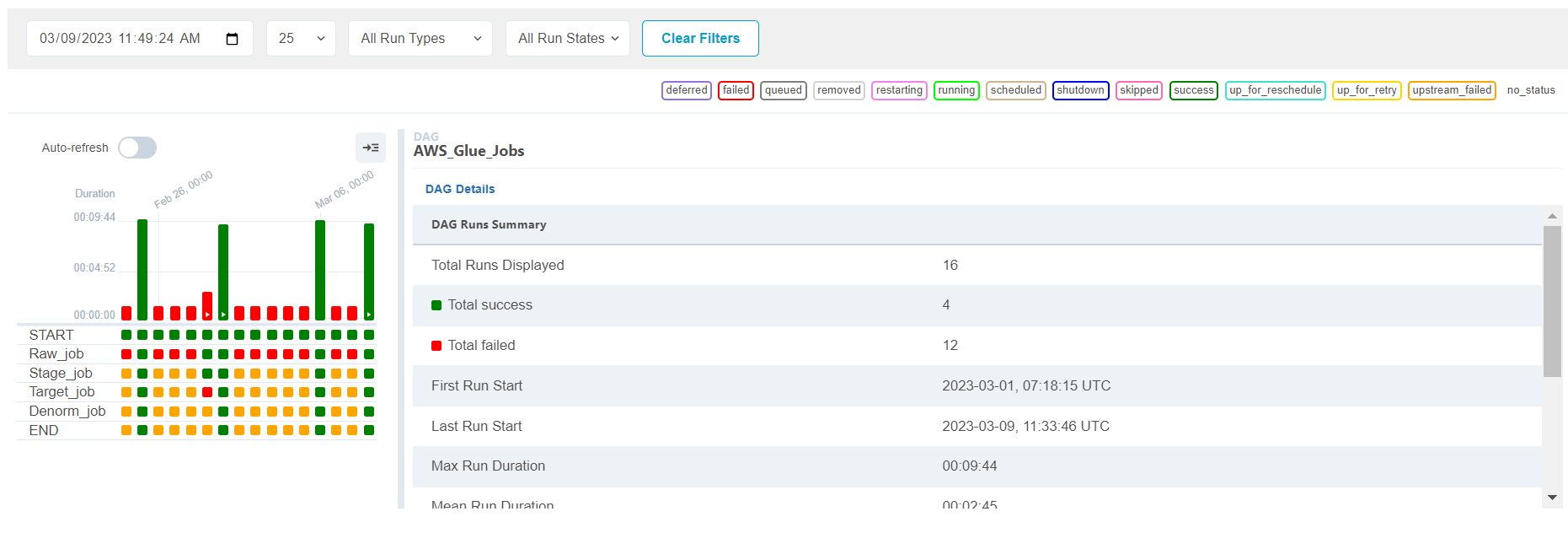
Trigger Dag which you want to run and wait for it to be completed

GRAPH:

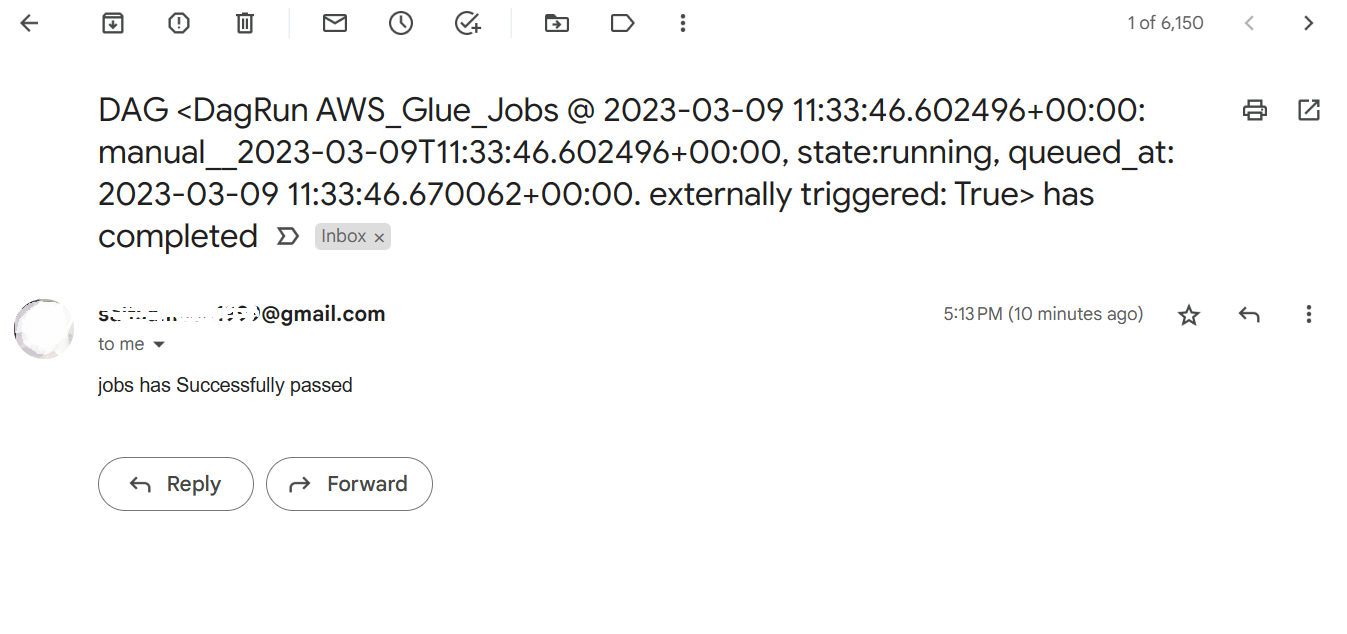


After the DAG Run

GRID:



After successful dag creation you will get a notification in mail.



If a dag file fails we will get a mail notification like



**THE END**