Assessment 2: GDDA707 - Advanced Data Engineering

Project 2: Data Integration

Mira Torririt

GDDA 707

Lecturer: Dr Zarah Zandi

School of Technology
Graduate Diploma in Data Analytics (Level 7)

March 15, 2024

Section 1: Summary of the project

This assessment aims to show the integration process and make the data more accessible and reliable for business strategic planning and data analysis.

Most businesses have numerous business applications accessed by employees, such as CRM, online ordering systems, product inventory, logistics, and accounting databases. Data integration connects all these various sources in one storage.

Data integration is the process of combining data from different data sources into a unified form. This assessment started with data input going to the database (as a data source), which is then extracted, transformed, and loaded (ETL) generally to the data warehouse (as a unified storage).

In Part A, the two datasets obtained from the Kaggle website were placed in two cloud buckets using the dataflow for ETL to BigQuery as the destination. These data were cleaned before loading into the database to ensure data integrity and prepare them for future analysis. Dataflow was chosen as a methodology for ETL. During this process, many errors occurred due to connections and configurations, which will be discussed explicitly in the next section.

In part B, the datasets used were also downloaded from the Kaggle website; both are customer transaction data. This time, the methodology used to store it in the Cloud platform is the data ingestion pipeline. The datasets were initially stored in one GCP bucket and eventually transferred to the Hadoop Distributed File System (HDFS), which is widely used for big data processing and analysis. Just like in Part A, errors occurred during the application of the said methodology; one of these is the creation of the destination folder in HDFS.

Section 2: Task-Specific Description and Technical Details

Part A: ETL Operations

Task 1: Selection of Datasets and Tools

The scenario involves integrating two e-commerce sales databases into the cloud platform. Two datasets obtained from the Kaggle website were chosen for the scenario.

The first one is the Amazon Sales Report, which consists of the following data: index, order ID, date, status, fulfillment, sales channel, ship-service-level, style, SKU, category, size, ASIN, courier status, quantity, currency, amount, ship-city, ship-state, ship-postal-code, ship-country, promotion-ids, fulfilled-by, unnamed: 22. (The Devastator, (n.d.)).

The second dataset concerns a one-year business transaction for a UK-based shop. Since 2007, the shop has been selling online gifts and homewares for adults and children. Customers worldwide can order directly from their website. They also have small business owner customers who buy through their retail outlet channels. The dataset comprises more than 500k rows and eight columns for the transaction no., date, product no, product, price, quantity, customer no., and country (Ramos, G., (n.d)).

The datasets are also available in my github account: https://github.com/Myres16/Data-Analytics-Assessments/tree/main/707

The detailed tools used in the data integration are as follows:

Data Integration Tools		
Dataset	Amazon Sale Report	Sales Transactions v.4a
Data cleaning and uploading to the database	Python	Python
Database / Storage	MongoDB	GCP Bucket
ETL (Extract, Transform, and Load)	Dataflow	Dataflow
Data warehouse	Big Query	Big Query
Cloud platform	Google Cloud Platform	Google Cloud Platform

The reasons for using different storage are:

1. MongoDB has a flexible schema, making storing and retrieving data easier.

2. To access the CSV file directly, I manually uploaded it to the Google Cloud Platform (GCP) Bucket to utilize it.

3. To explore the MongoDB and GCP Bucket functions.

I chose GCP because the credits were available. To maximize its use, I chose the dataflow, which is free of charge in GCP and is a product of Google, which is responsible for the ETL process.

Here's the process to access the dataflow:

1. Go to the GCP navigation menu, scroll down, click MORE PRODUCTS, scroll down, look for ANALYTICS, and click Dataflow.

2. Go to Jobs and click + CREAT JOB FROM TEMPLATE

3. Fill out the Create job from the template

Important: The regional endpoint should match the source database's regional settings (for example, MongoDB Atlas) and the destination database's (for example, the Big Query). The dataflow template should match the source and destination databases (for example, MongoDB to BigQuery)

4. Once you have filled out the dataflow template, it will show you the required parameters to be filled out.

5. Click RUN JOB.

Task 2: Load and Pre-processing

I downloaded the datasets from the Kaggle website and imported them to Python using the pandas library and pd for pre-integration. read function.

- Amazon Sales Dataset

```
import pandas as pd
```

```
AmazonDF = pd.read_csv('Amazon Sale Report.csv')
AmazonDF
```

```
C:\Users\torri\AppData\Local\Temp\ipykernel_20064\1226751192.py:1: DtypeWarning: Columns (23) have mixed types. Specify dtype o
ption on import or set low_memory=False.
  AmazonDF = pd.read_csv('Amazon Sale Report.csv')
```

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	...	currency	Amount	ship-city	ship-sta
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	Set	...	INR	647.62	MUMBAI	MAHARASHTF
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	kurta	...	INR	406.00	BENGALURU	KARNATAI
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	kurta	...	INR	329.00	NAVI MUMBAI	MAHARASHTF
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	Western Dress	...	INR	753.33	PUDUCHERRY	PUDUCHERI
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	Top	...	INR	574.00	CHENNAI	TAMIL NAI
128970	128970	406-6001380-7673107	05-31-22	Shipped	Amazon	Amazon.in	Expedited	JNE3697	JNE3697-KR-XL	kurta	...	INR	517.00	HYDERABAD	TELANGAI
128971	128971	402-9551604-7544318	05-31-22	Shipped	Amazon	Amazon.in	Expedited	SET401	SET401-KR-NP-M	Set	...	INR	999.00	GURUGRAM	HARYAI
128972	128972	407-9547469-3152358	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0157	J0157-DR-XXL	Western Dress	...	INR	690.00	HYDERABAD	TELANGAI
128973	128973	402-6184140-0545956	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0012	J0012-SKD-XS	Set	...	INR	1199.00	Halol	Guja
128974	128974	408-7436540-8728312	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0003	J0003-SET-S	Set	...	INR	696.00	Raipur	CHHATTISGAI

128975 rows × 24 columns

Data was cleaned to prepare it for integration and future analysis. To facilitate, I used the isnull function to look for null values and the fillna function to replace the columns with null values. I saved the cleaned CSV file, as this is the one to be imported into the database.

```
AmazonDF.isnull().sum()
```

```
index                   0
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Style                   0
SKU                     0
Category                0
Size                    0
ASIN                    0
Courier Status       6872
Qty                     0
currency             7795
Amount               7795
ship-city              33
ship-state             33
ship-postal-code       33
ship-country           33
promotion-ids       49153
B2B                     0
fulfilled-by        89698
Unnamed: 22         49050
dtype: int64
```

```
AmazonDF['Unnamed: 22'].fillna('TRUE', inplace=True)
```

```
AmazonDF['Courier Status'].fillna('Cancelled', inplace=True)
```

```
AmazonDF['currency'].fillna('INR', inplace=True)
```

```
AmazonDF['Amount'].fillna(0.00, inplace=True)
```

```
AmazonDF['fulfilled-by'].fillna('FedEx', inplace=True)
```

```
AmazonDF['ship-city'].fillna('SOUTH', inplace=True)
```

```
AmazonDF['ship-state'].fillna('DELHI', inplace=True)
```

```
AmazonDF['ship-postal-code'].fillna('110015', inplace=True)
```

```
AmazonDF['ship-country'].fillna('IN', inplace=True)
```

```
AmazonDF['promotion-ids'].fillna('NotAvailable', inplace=True)
```

```
AmazonDF.isnull().sum()
```

```
index               0
Order ID            0
Date                0
Status              0
Fulfilment          0
Sales Channel       0
ship-service-level  0
Style               0
SKU                 0
Category            0
Size                0
ASIN                0
Courier Status      0
Qty                 0
currency            0
Amount              0
ship-city           0
ship-state          0
ship-postal-code    0
ship-country        0
promotion-ids       0
B2B                 0
fulfilled-by        0
Unnamed: 22         0
dtype: int64
```

```
#Saving to a new CSV file
AmazonDF.to_csv('AmazonDF.csv')
```

- Sales Transaction Dataset

```
import pandas as pd
```

```
SalesTransactionDF=pd.read_csv('Sales Transaction v.4a.csv')
SalesTransactionDF
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
0	581482	12/9/2019	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom
1	581475	12/9/2019	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom
2	581475	12/9/2019	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom
3	581475	12/9/2019	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom
4	581475	12/9/2019	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom
...	...	...	...	...	...	...	...	...
536345	C536548	12/1/2018	22168	Organiser Wood Antique White	18.96	-2	12472.0	Germany
536346	C536548	12/1/2018	21218	Red Spotty Biscuit Tin	14.09	-3	12472.0	Germany
536347	C536548	12/1/2018	20957	Porcelain Hanging Bell Small	11.74	-1	12472.0	Germany
536348	C536548	12/1/2018	22580	Advent Calendar Gingham Sack	16.35	-4	12472.0	Germany
536349	C536548	12/1/2018	22767	Triple Photo Frame Cornice	20.45	-2	12472.0	Germany

536350 rows × 8 columns

```
SalesTransactionDF.isnull().sum()
```

```
TransactionNo     0
Date              0
ProductNo         0
ProductName       0
Price             0
Quantity          0
CustomerNo       55
Country           0
dtype: int64
```

```
SalesTransactionDF['CustomerNo'].fillna('12835', inplace=True)
```

```
SalesTransactionDF.isnull().sum()
```

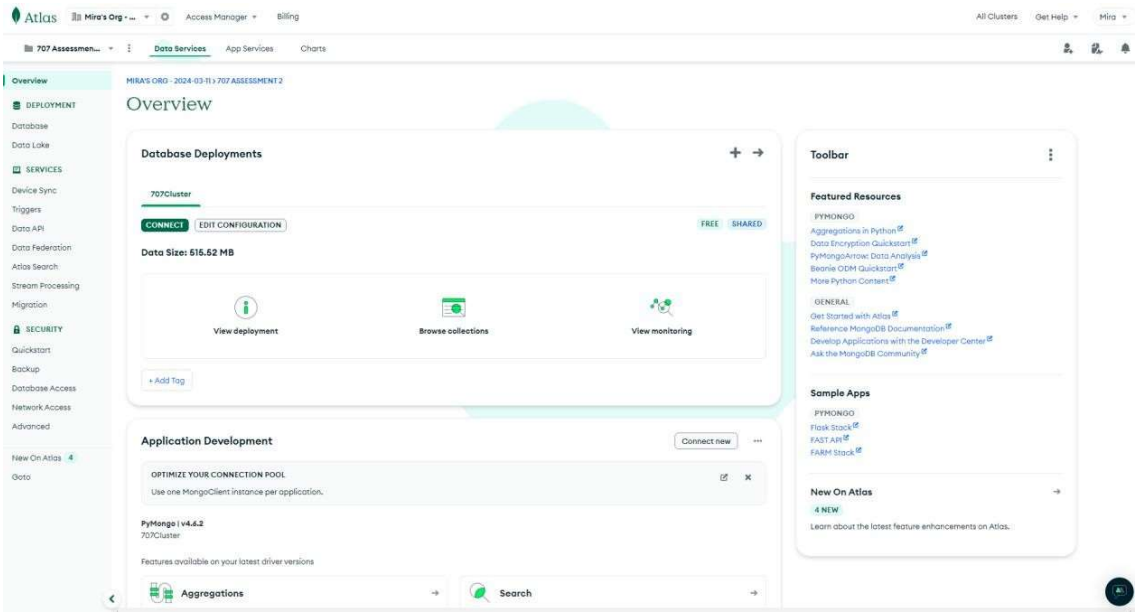
```
TransactionNo    0
Date             0
ProductNo        0
ProductName      0
Price            0
Quantity         0
CustomerNo       0
Country          0
dtype: int64
```

```
SalesTransactionDF.to_csv('Cleaned_SalesTransactionDF.csv')
```

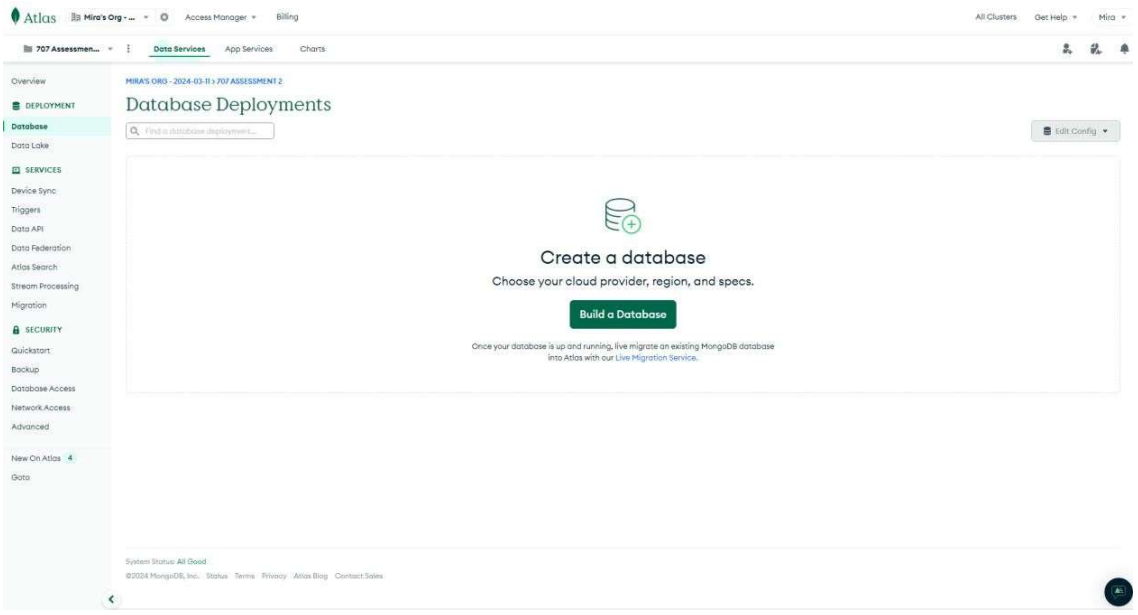
Task 3: ETL Operations and Integration

Amazon Sales Dataset – From Python to MongoDB

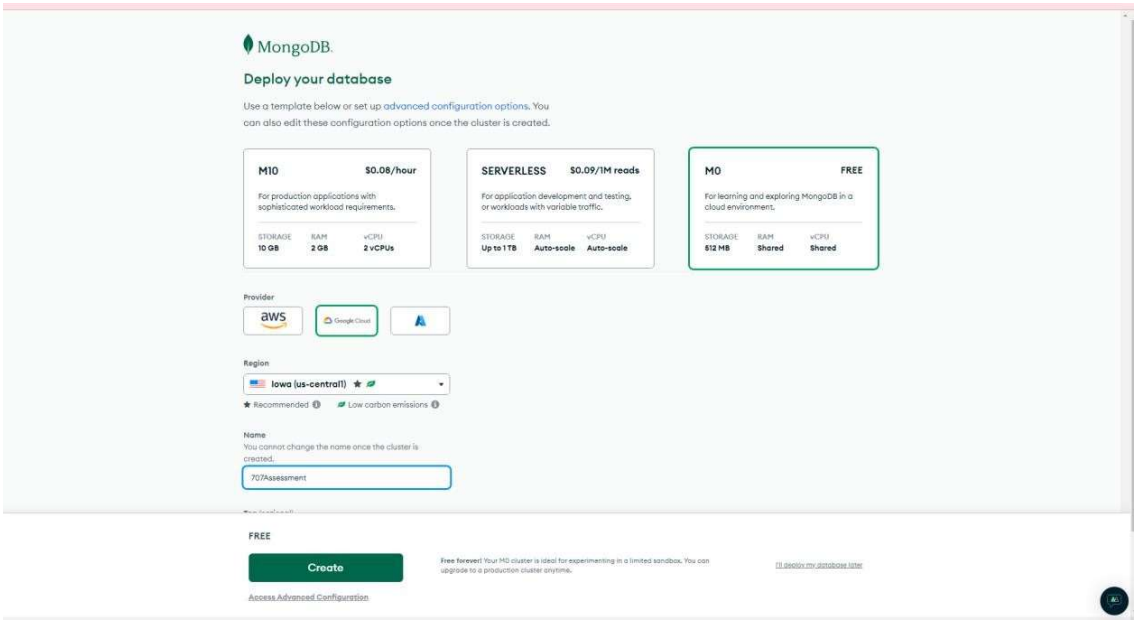
- To set up MongoDB Atlas, go to the Overview page.



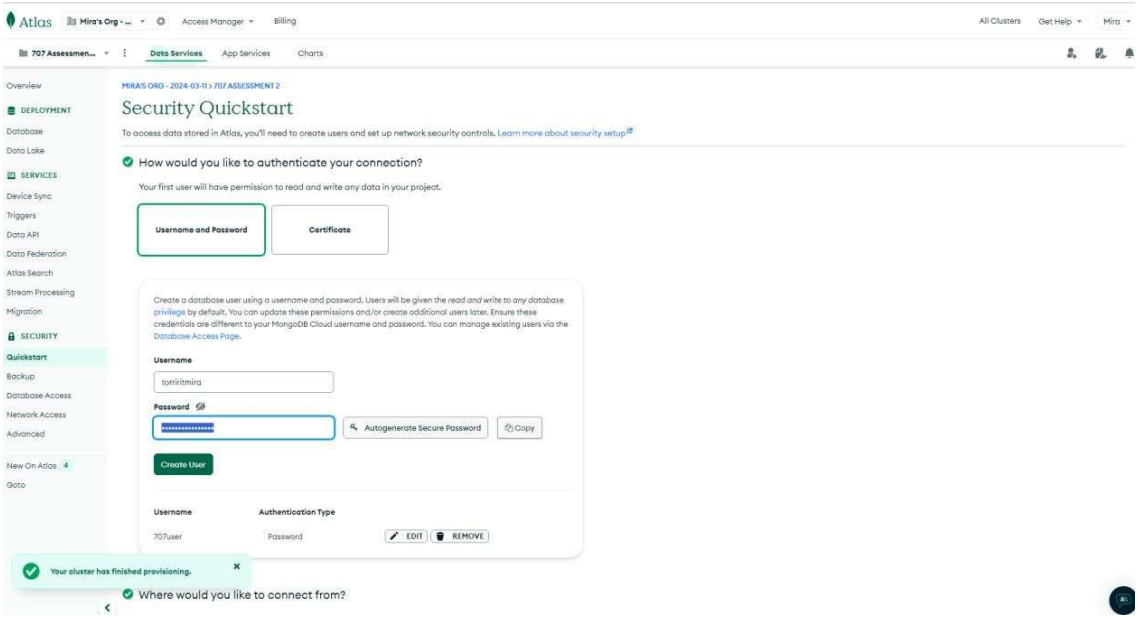
- Create a cluster - click database under deployment, then click Build a Database

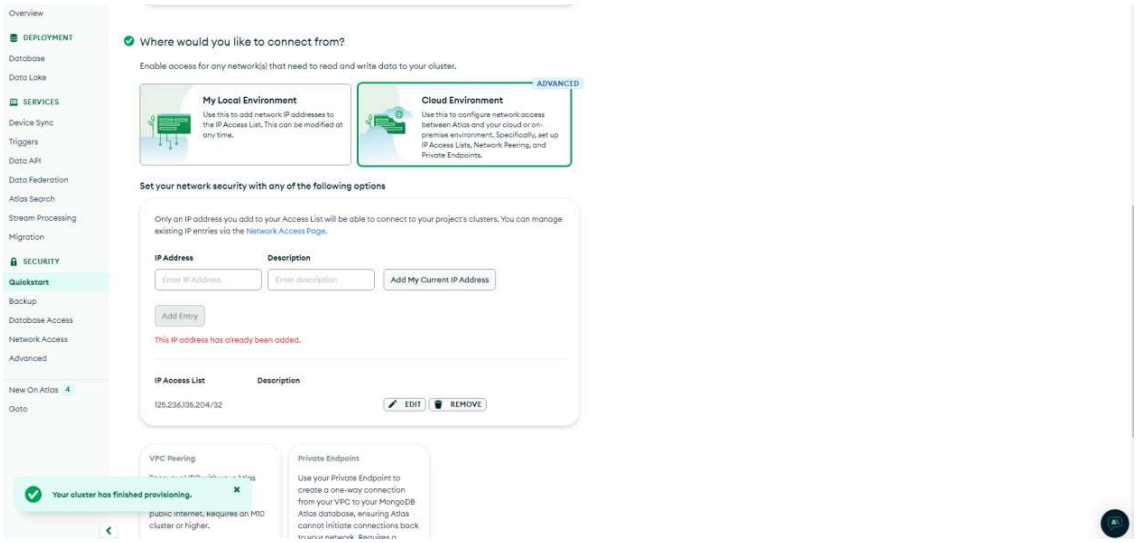


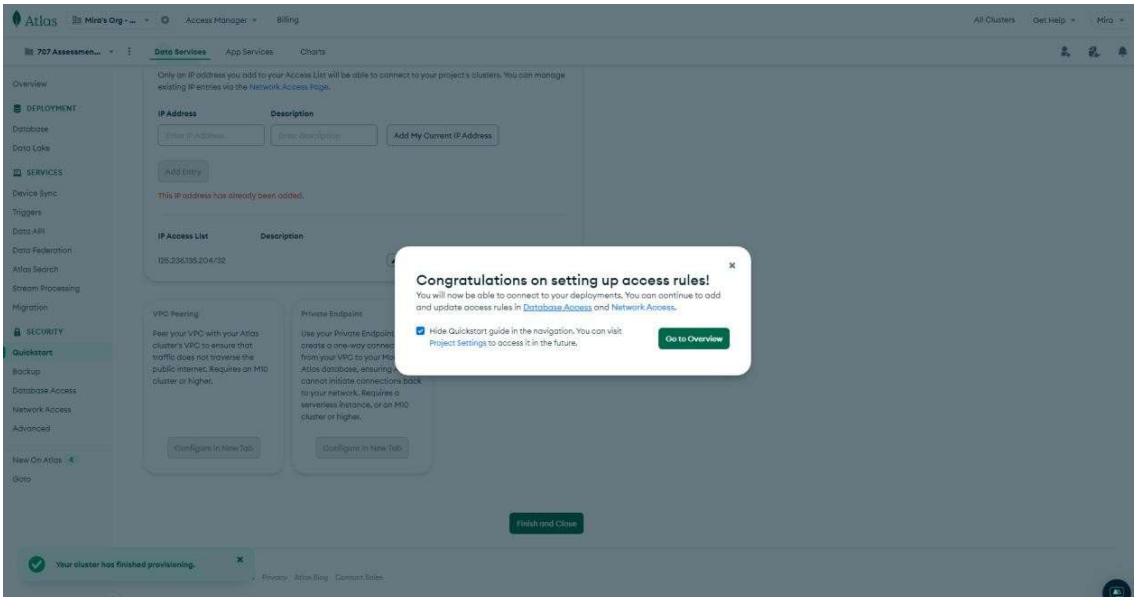
- Create cluster: cluster name created – 707Assessment



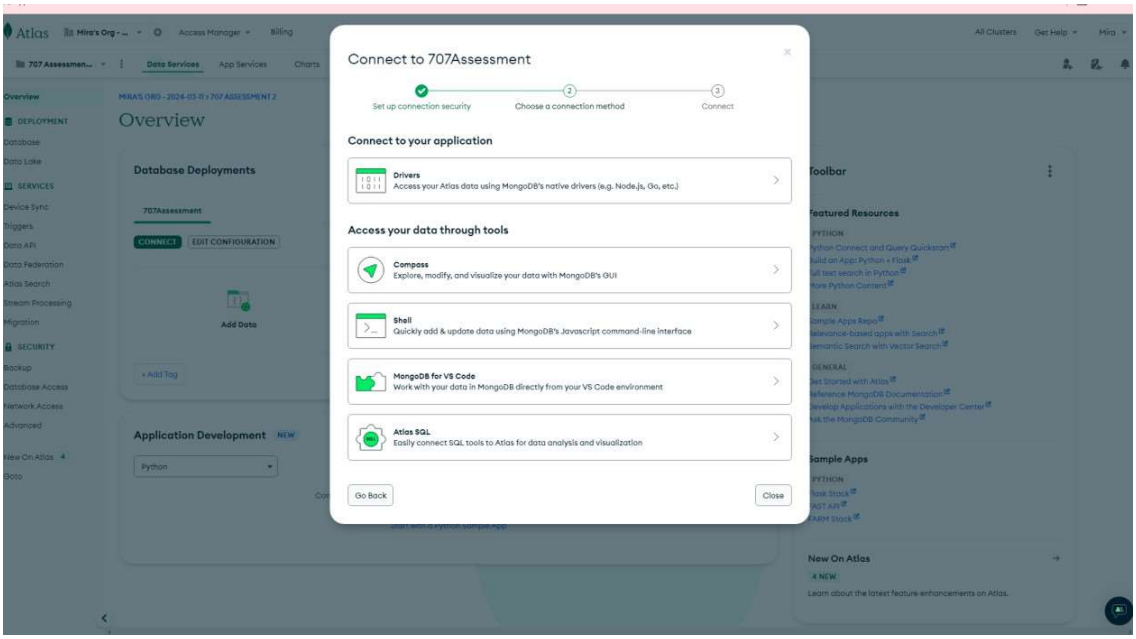
- Create user ID



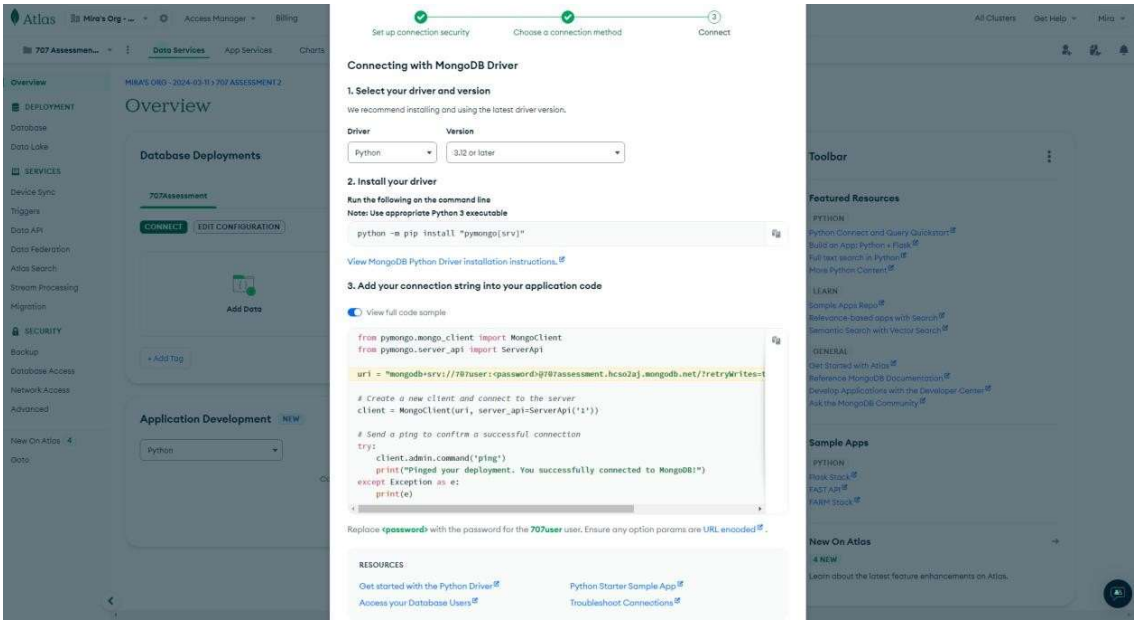

Overview
DEPLOYMENT
Database
Data Lake
SERVICES
Device Sync
Triggers
Data API
Data Federation
Atlas Search
Stream Processing
Migration
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
New On Atlas 4
Goto
Where would you like to connect from?
Enable access for any network(s) that need to read and write data to your cluster.
My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.
ADVANCED
Cloud Environment
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.
Set your network security with any of the following options
Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.
IP Address
Description
Add My Current IP Address
Add Entry
This IP address has already been added.
IP Access List
Description
125.236.136.204/32
EDIT
REMOVE
VPC Peering
Private Endpoint
Use your Private Endpoint to create a one-way connection from your VPC to your MongoDB Atlas database, ensuring Atlas cannot initiate connections back
Your cluster has finished provisioning.
public internet. Requires an M10 cluster or higher.


Atlas
Access Manager
Billing
All Clusters
Get Help
Mira
707 Assessmen...
Data Services
App Services
Charts
Overview
DEPLOYMENT
Database
Data Lake
SERVICES
Device Sync
Triggers
Data API
Data Federation
Atlas Search
Stream Processing
Migration
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
New On Atlas 4
Goto
Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.
IP Address
Description
Add My Current IP Address
Add Entry
This IP address has already been added.
IP Access List
Description
125.236.136.204/32
VPC Peering
Peer your VPC with your Atlas cluster's VPC to ensure that traffic does not traverse the public internet. Requires an M10 cluster or higher.
Private Endpoint
Use your Private Endpoint to create a one-way connection from your VPC to your MongoDB Atlas database, ensuring Atlas cannot initiate connections back to your network. Requires a serverless instance, or an M10 cluster or higher.
Configure in New Tab
Configure in New Tab
Congratulations on setting up access rules!
You will now be able to connect to your deployments. You can continue to add and update access rules in Database Access and Network Access.
Hide Quickstart guide in the navigation. You can visit Project Settings to access it in the future.
Go to Overview
Finish and Close
Your cluster has finished provisioning.
Privacy
Atlas Blog
Contact Sales

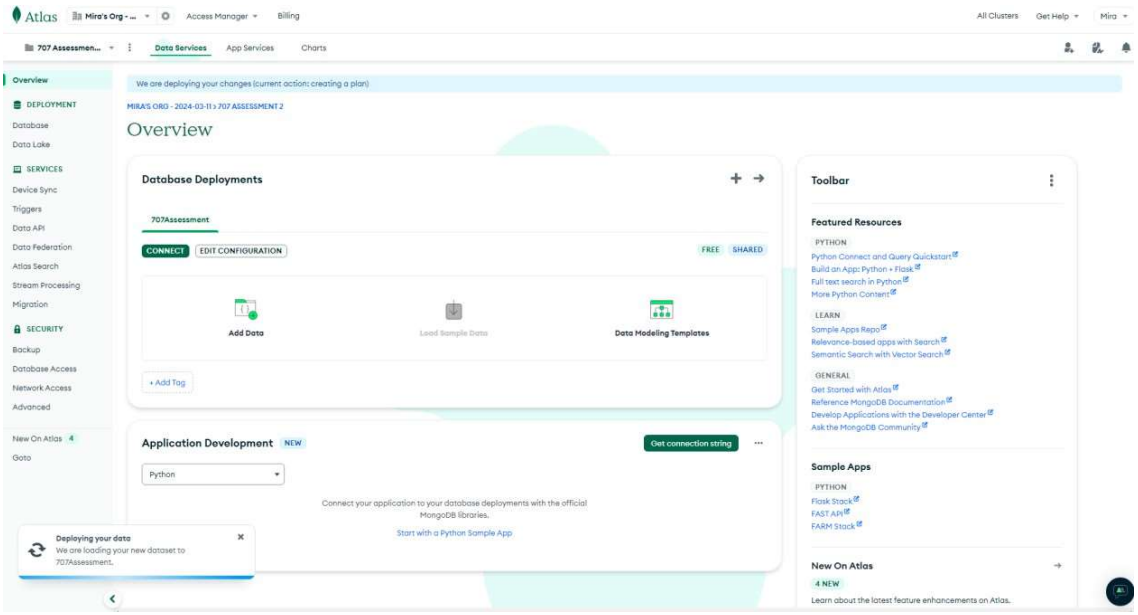
- Setting up a connection to the database. Click drivers.



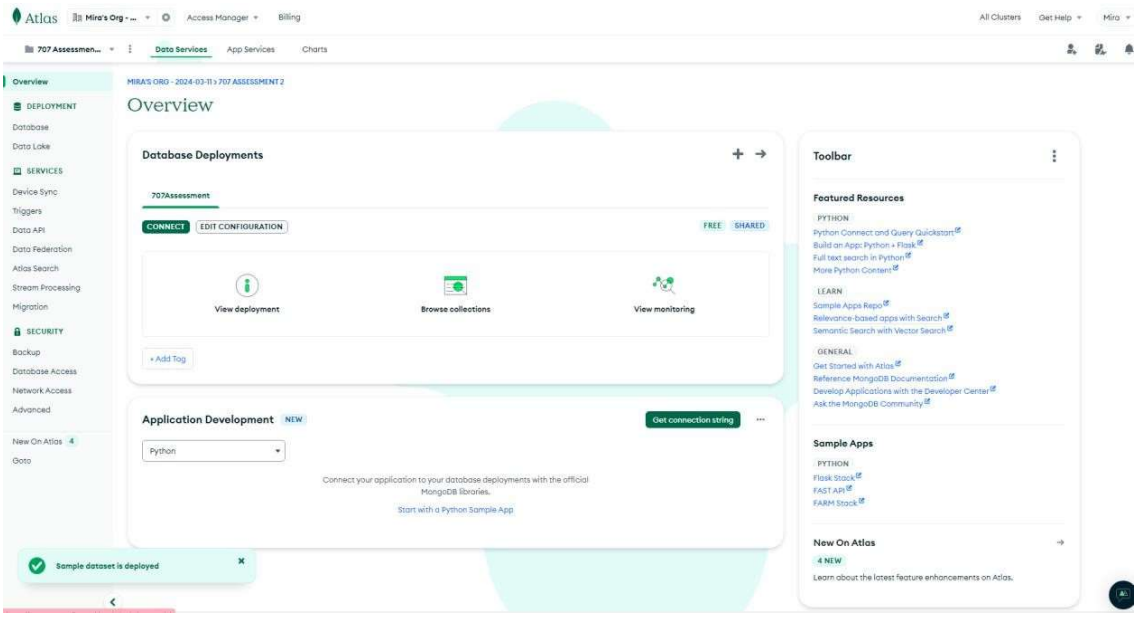
- Make sure the MongoDB is set to Python Ver. 3.12 later.



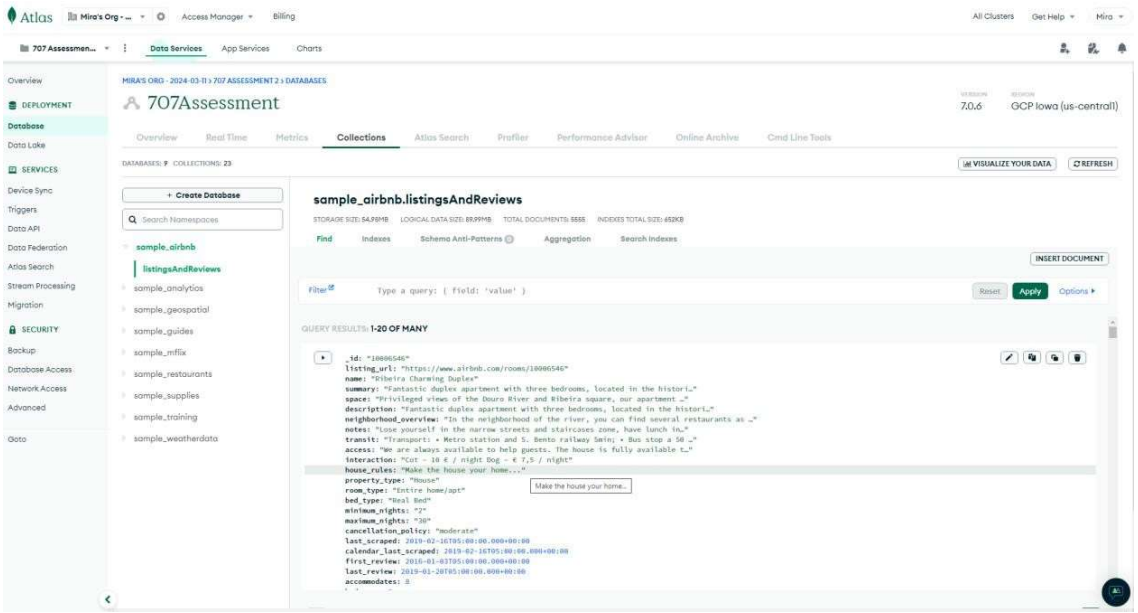
- Click Load Sample Data to browse the database collection.



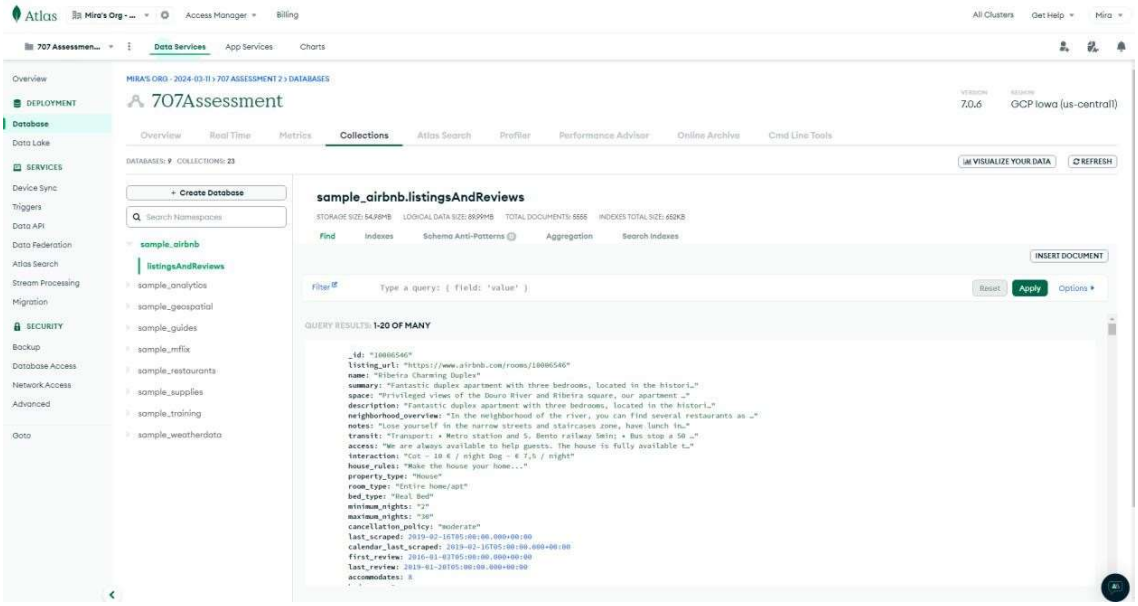
- Once deployed, click Browse Collection.



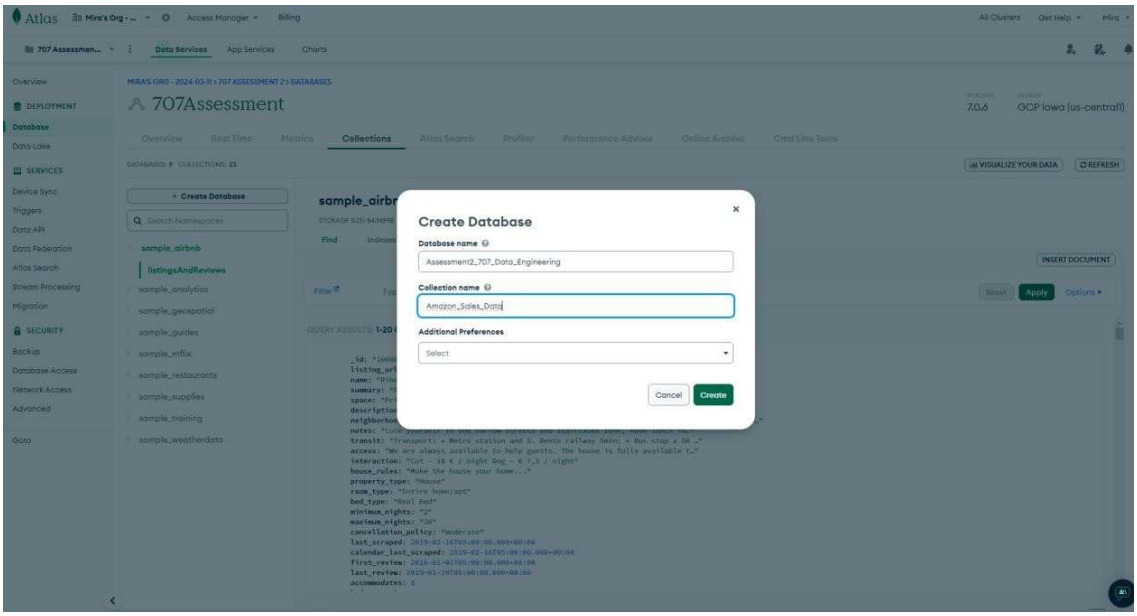
- Displaying sample MongoDB generated collection.



- Create a database and collection. Click + Create Database.



- Name the database and collection. Click Create.



- Set up a connection from Python to MongoDB Atlas.

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

def get_mongodbclient():

    userpassword = 'qxZAHRkXprBqiAUy'
    uri = 'mongodb+srv://707user:u4NG9f4sCtn2mPhD@707assessment.hcso2aj.mongodb.net/?retryWrites=true&w=majority&appName=707Asses

    # Create a new client and connect to the server
    client = MongoClient(uri, server_api=ServerApi('1'))

    # Send a ping to confirm a successful connection
    try:
        client.admin.command('ping')
        print("Pinged your deployment. You successfully connected to MongoDB!")
        return client
    except Exception as e:
        print(e)
        return None


# Connect to MongoDB Atlas
client = get_mongodbclient()
# Initiate an instance of the database
db = client['Assessment2_707_Data_Engineering']
```

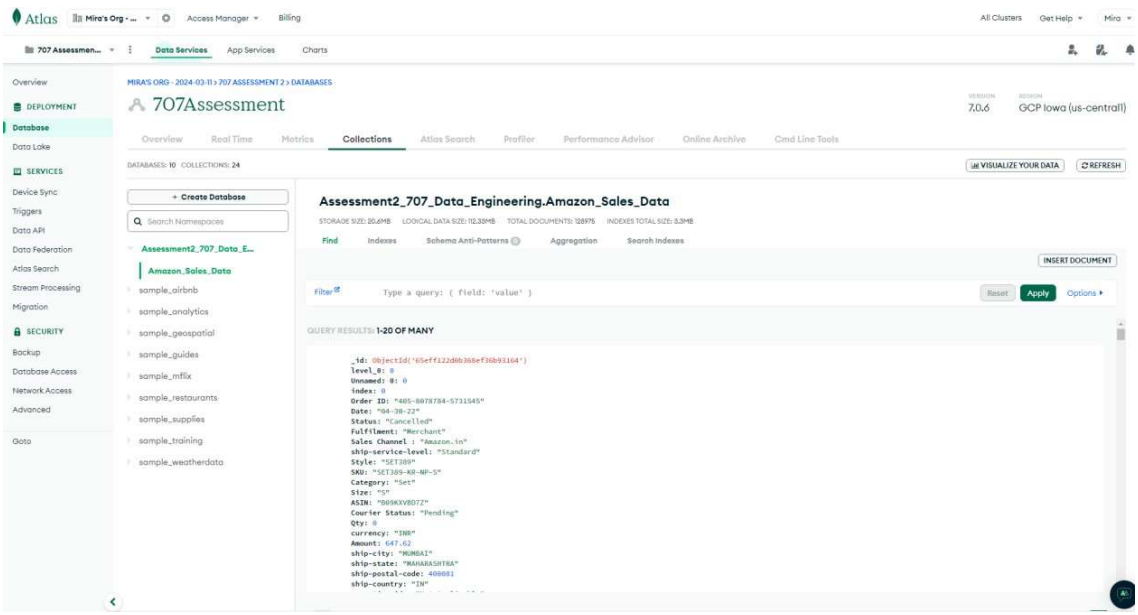
```
Pinged your deployment. You successfully connected to MongoDB!
```

- Extracting the data from Python to MongoDB

```
# Load your dataset (e.g., from a CSV file)
# Source: https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data
data = pd.read_csv('AmazonDF.csv')
# Due to limitation in Mongodb Atlas, I limit the data to 100,000 rows
# data = data.head(200000)

# Access amazon_sales_report collection
collection = db['Amazon_Sales_Data']

# Reset the index if needed
data.reset_index(inplace=True)

# Set batch size due to timeout issue
batch_size = 10000

# Insert batches into MongoDB
for i in range(0, len(data), batch_size):
    batch = data.iloc[i:i + batch_size].to_dict('records')
    collection.insert_many(batch)


print(f"Inserted {len(data)} records into MongoDB Atlas.")
```

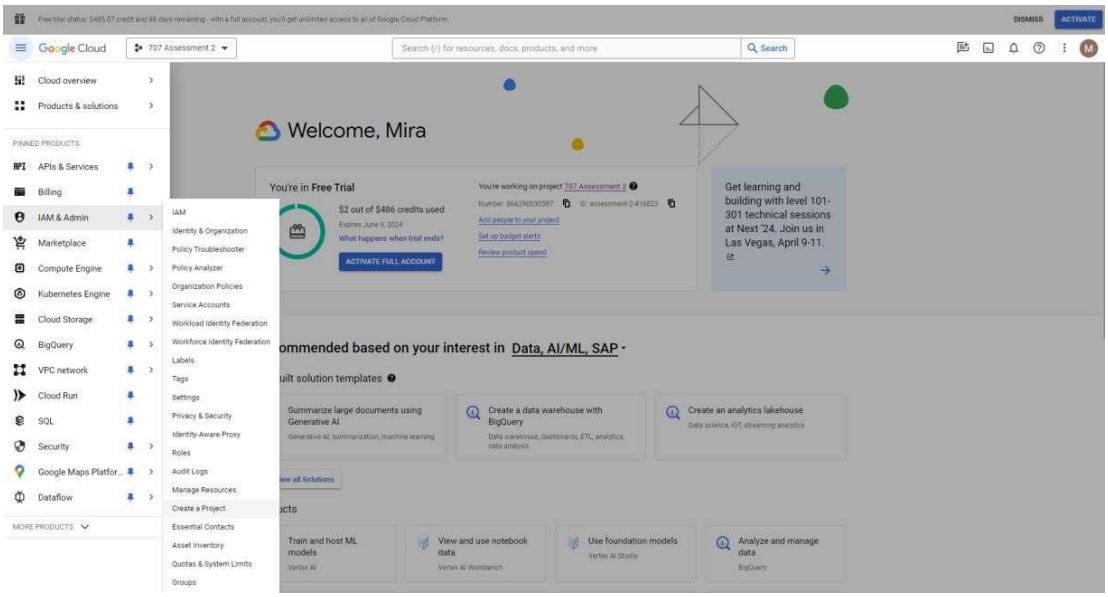
```
Inserted 128975 records into MongoDB Atlas.
```

- Checking in MongoDB. Go to the Collections tab.

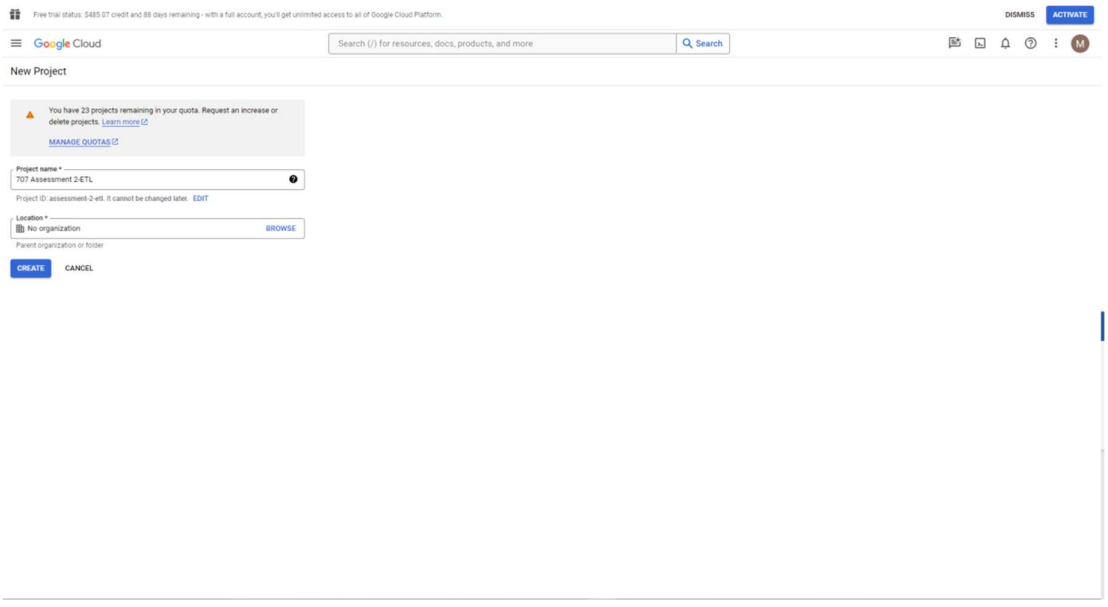


MongoDB Atlas to BigQuery

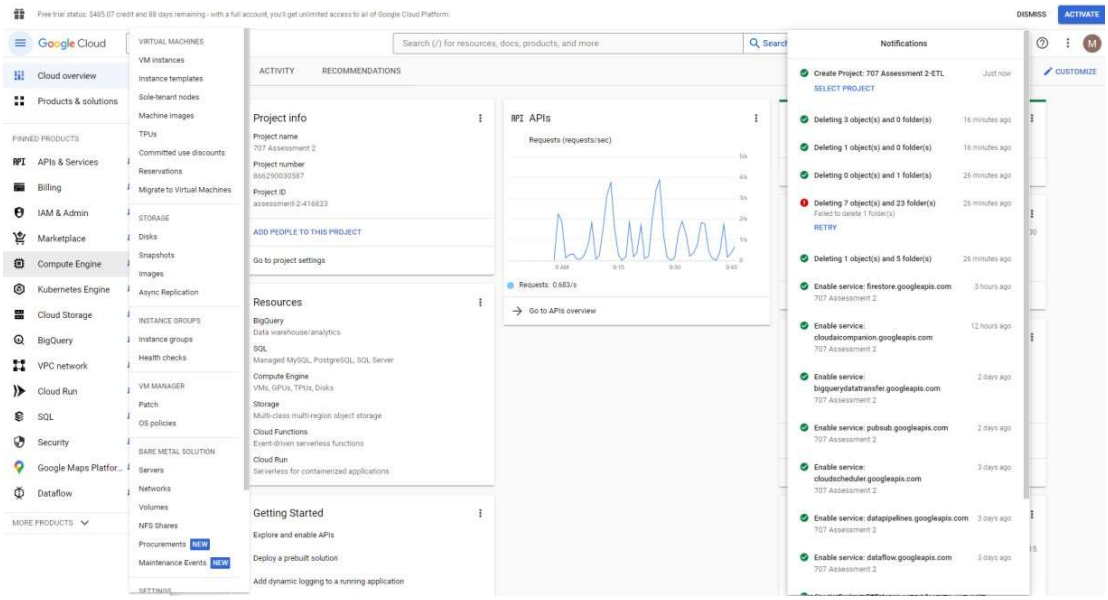
- Create a new project in GCP – in the navigation menu, go to IAM & Admin and click Create a Project.



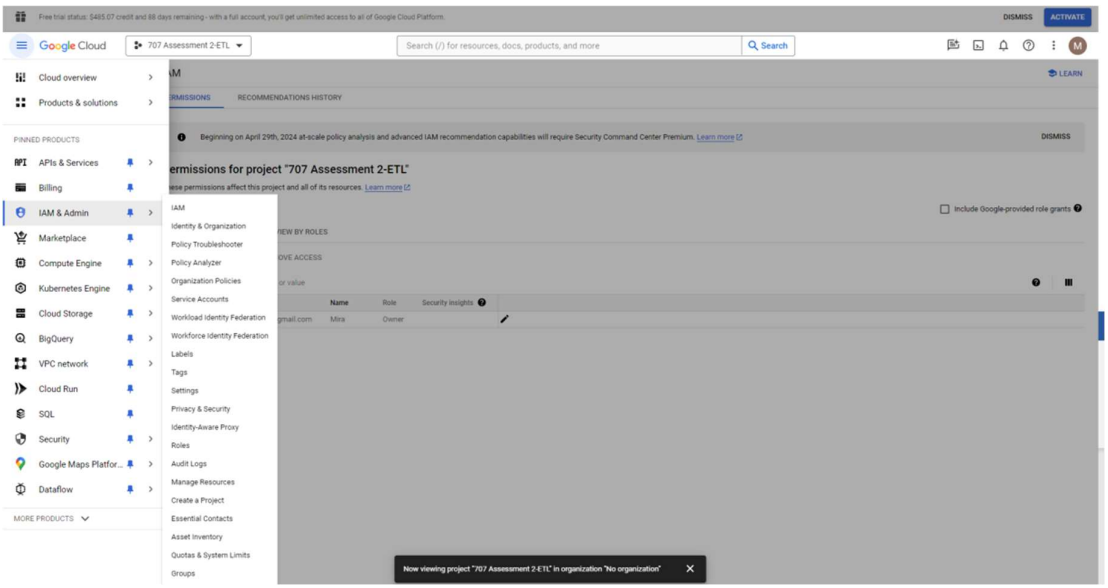
- Fill out the Project name and Location and click CREATE



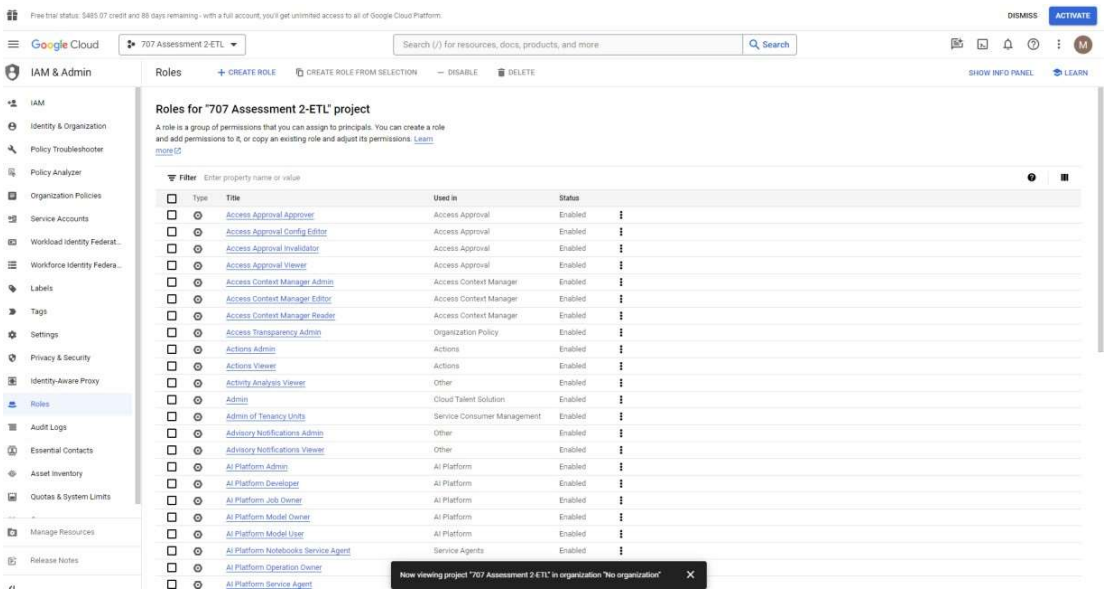
A notification will pop out and will be seen on the right side of the page. Click SELECT PROJECT.



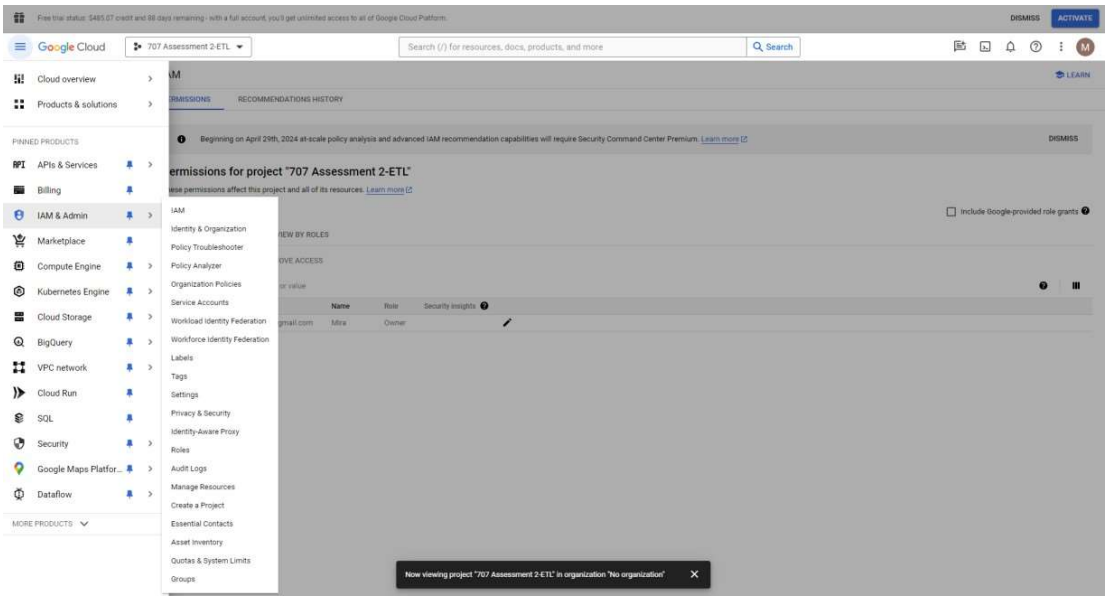
- Create a custom role – Go to IAM & Admin and click Roles



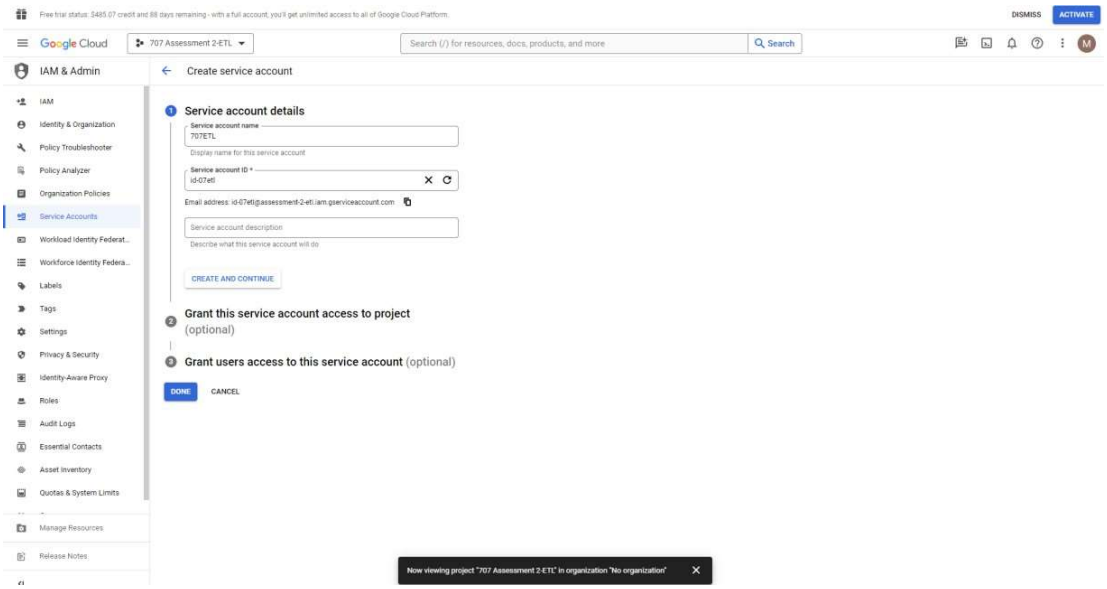
- Click + CREATE ROLE

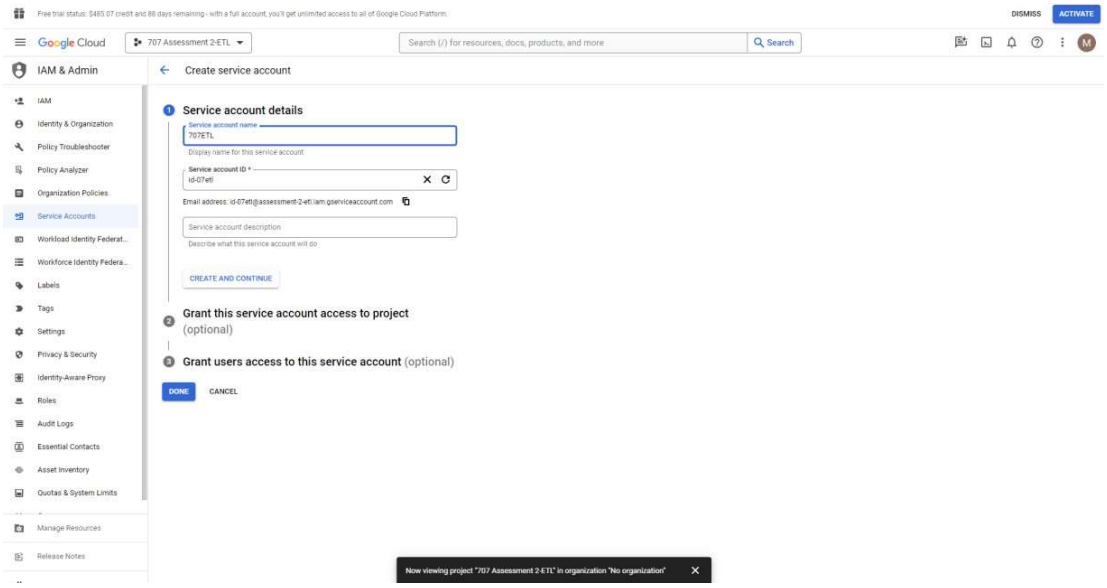


- Create a service account for the project. Go to IAM & Admin and click on Service Accounts.

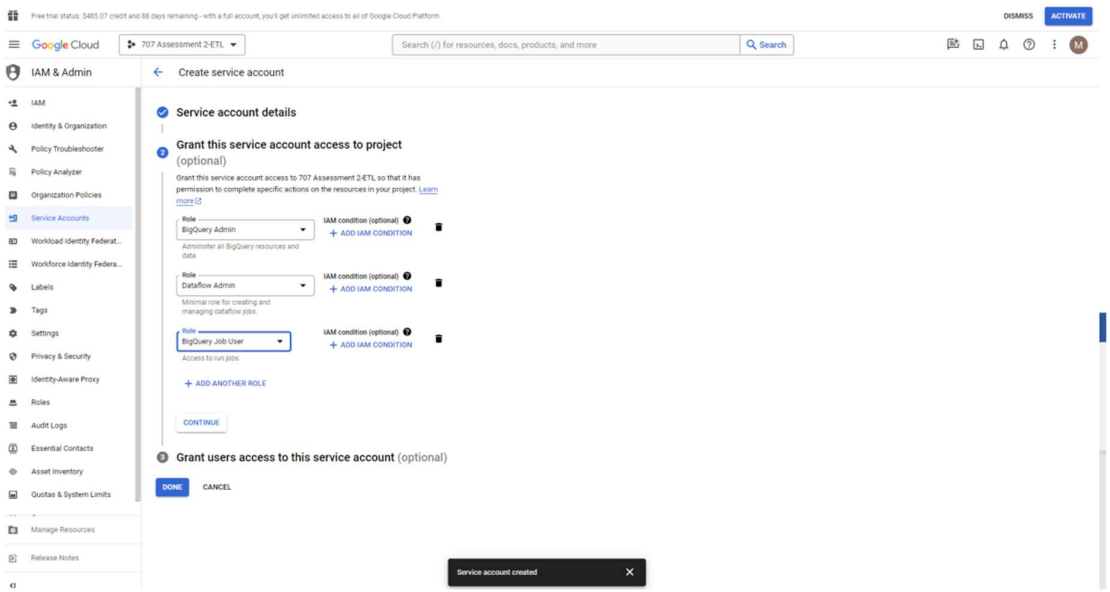


- Click +CREATE SERVICE ACCOUNT; provide the Service account name and click CREATE AND CONTINUE.

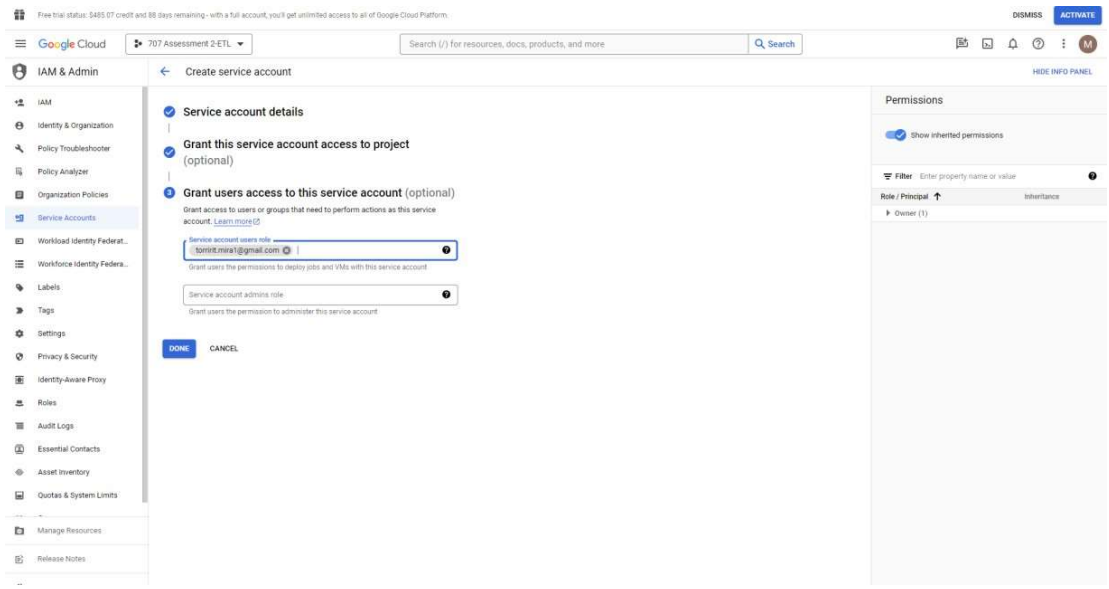




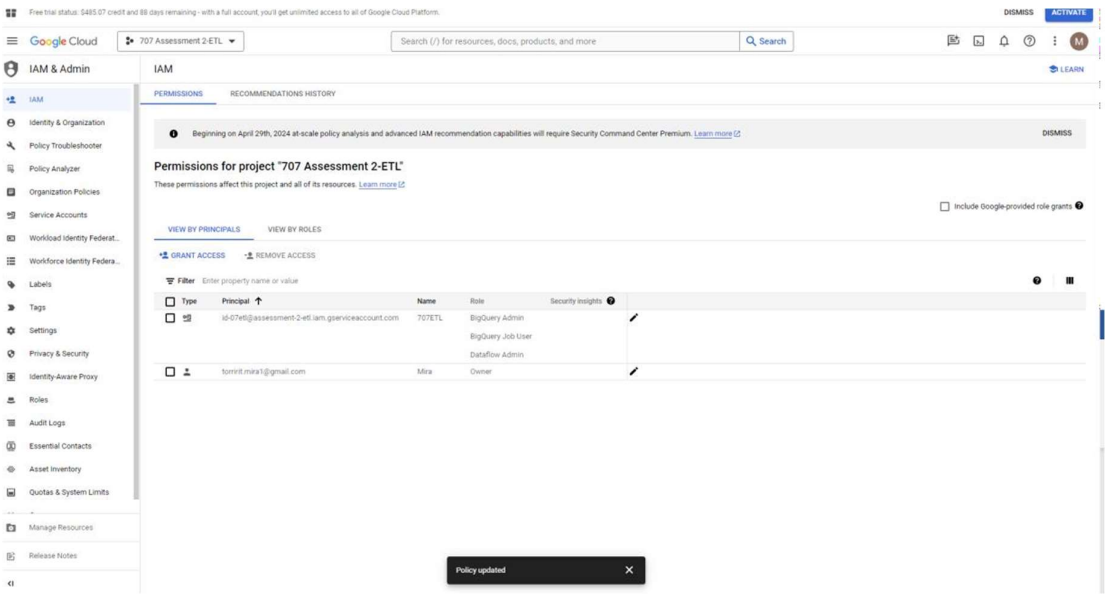
- Select three roles and click CONTINUE



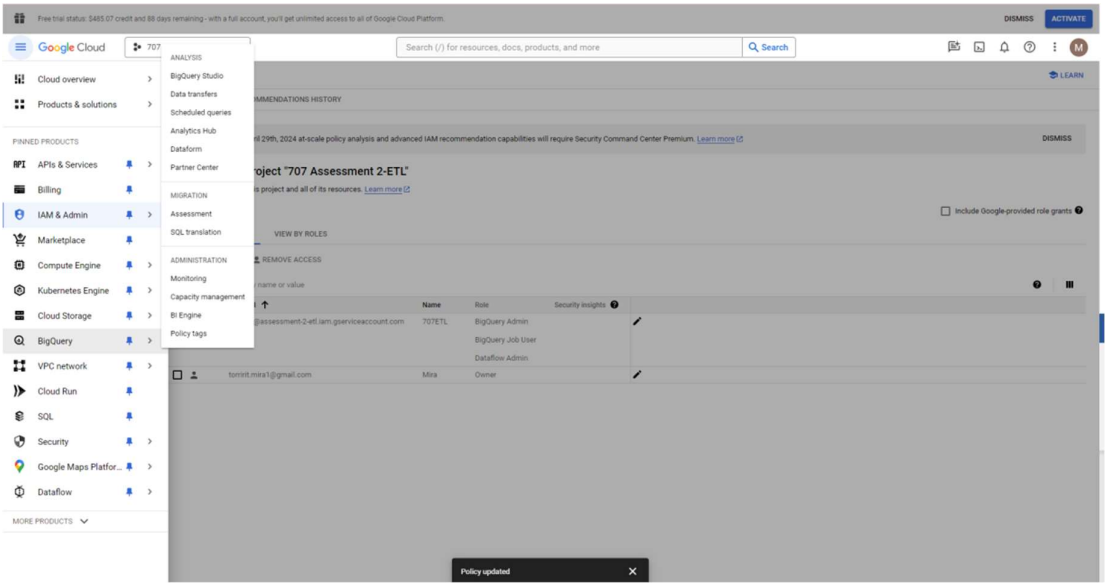
- Provide the email address of the user and click DONE.



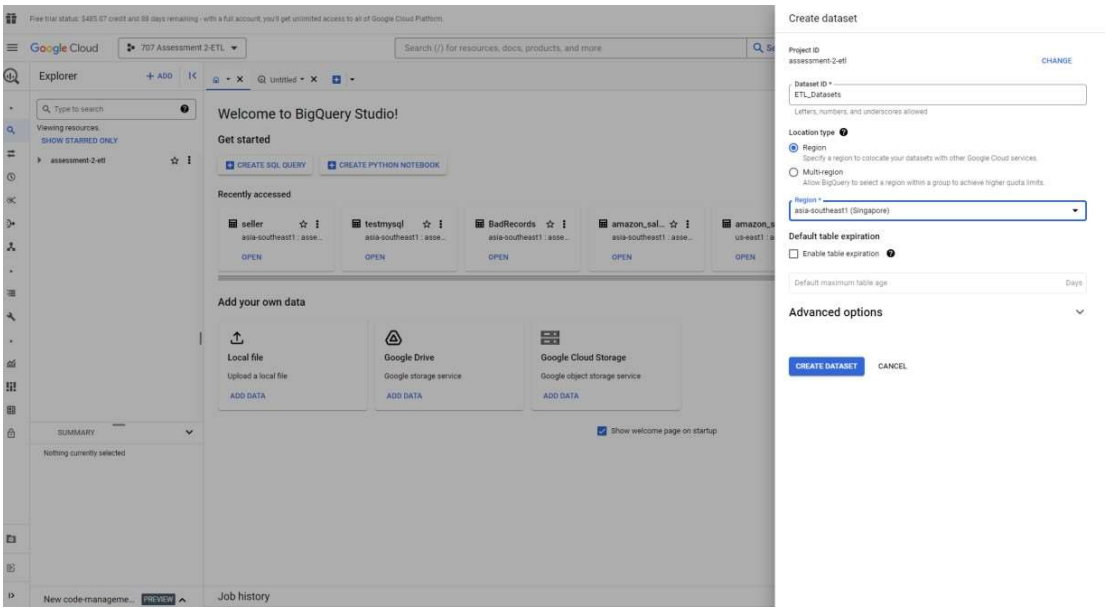
- Once the service account is created, go to IAM & Admin and click IAM to verify the new user.

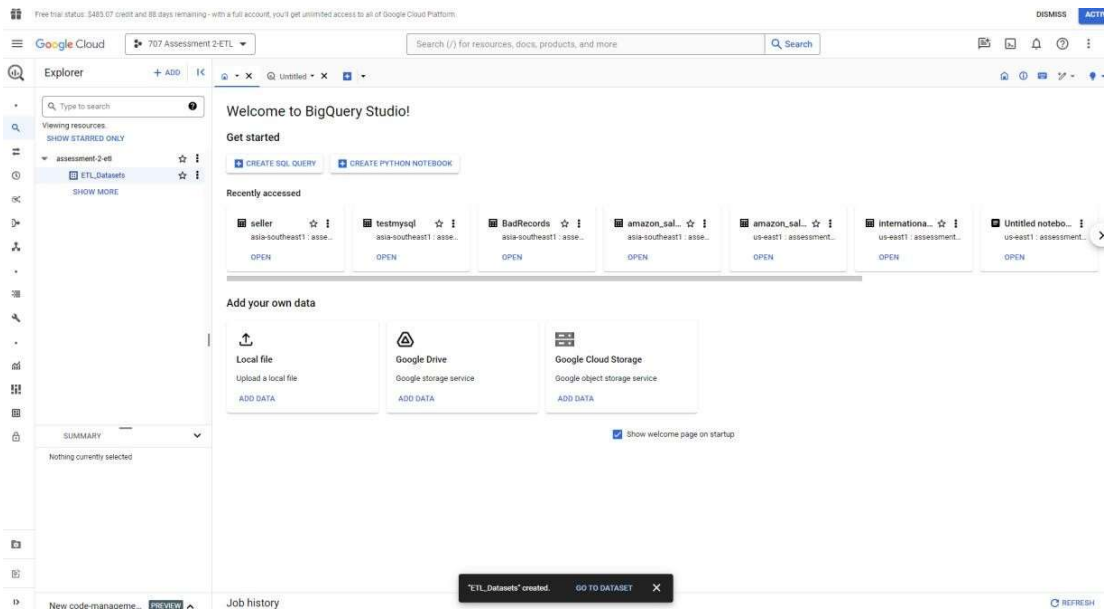


- Create a BigQuery Table – go to BigQuery and click BigQuery Studio

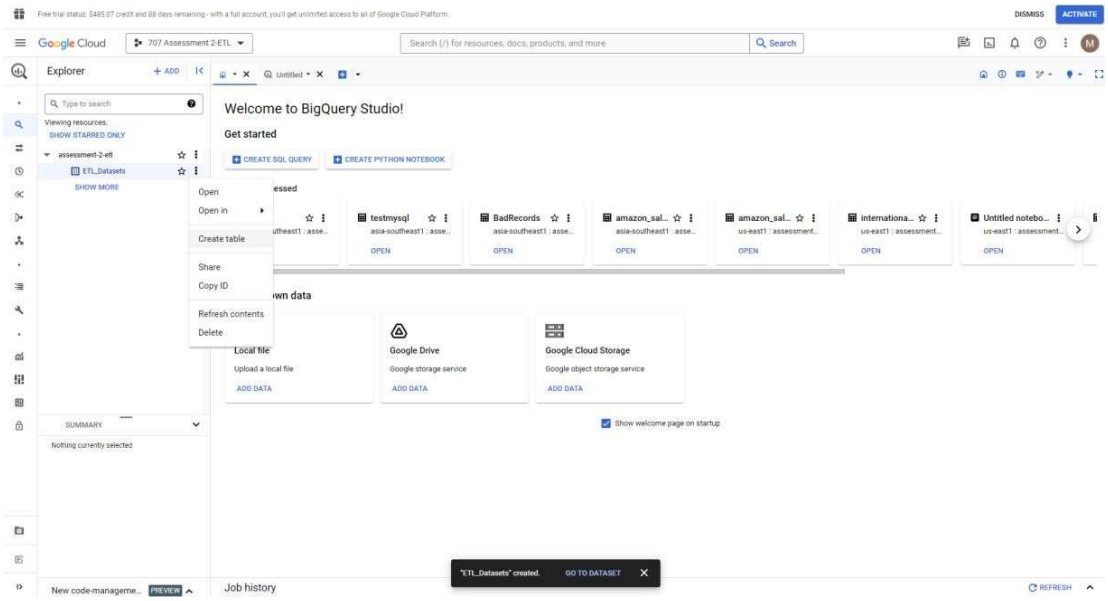


- Click the database assessment-2-etl and create dataset. Create a Dataset ID. Choose the location type and Region. I chose the Southeast (Singapore) region as this is the region I put in my MongoDB then click CREATE DATASET.

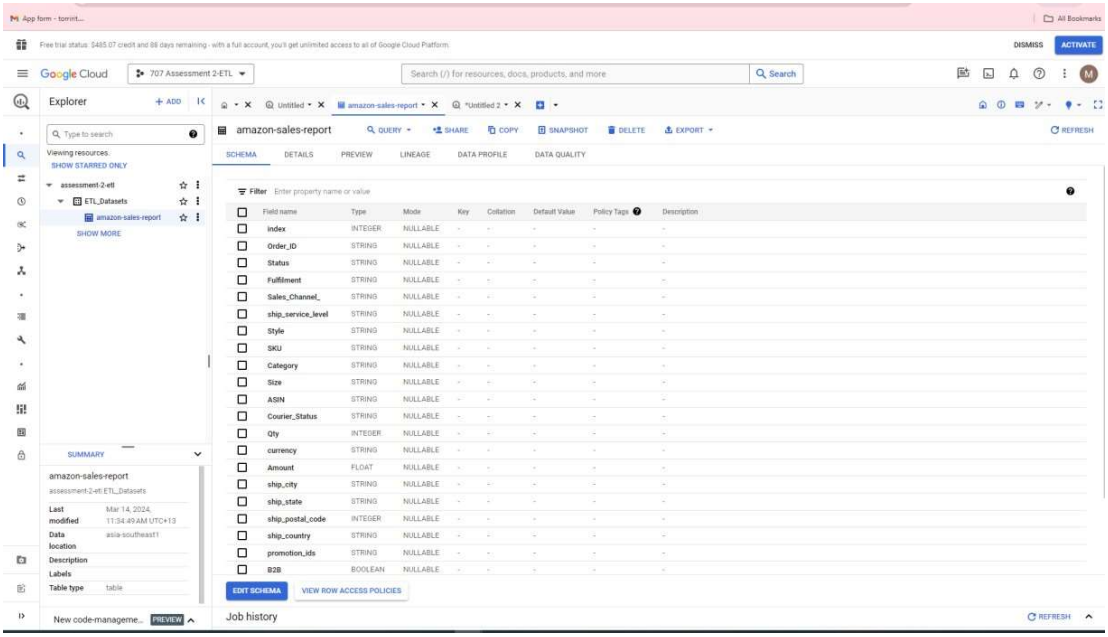




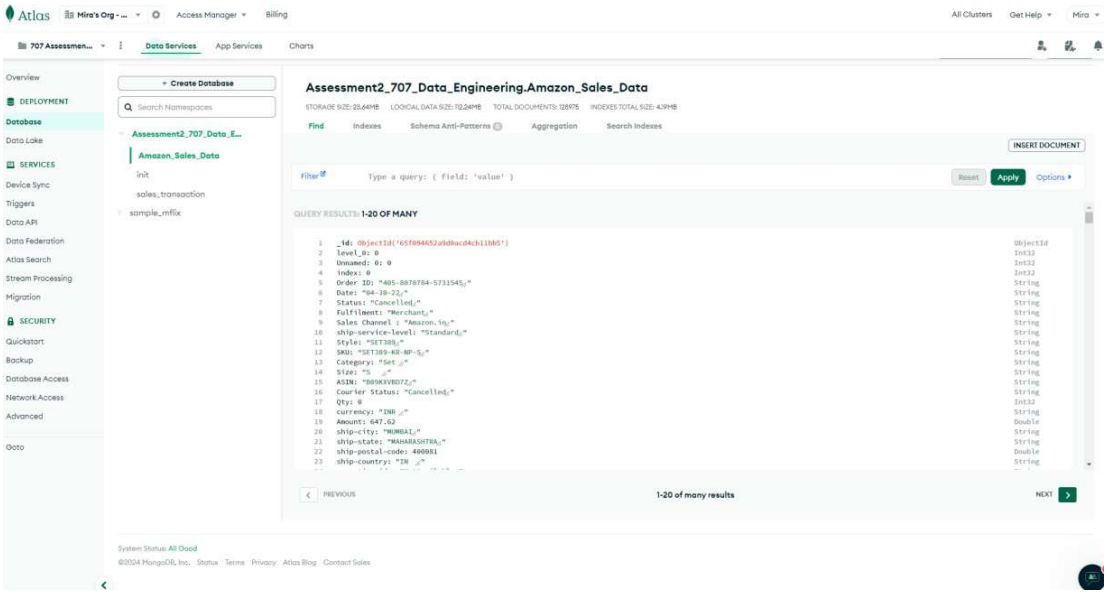
- From your created dataset table, click the three dots beside it and click Create table. Fill out the needed data (with reference to the MongoDB collection).



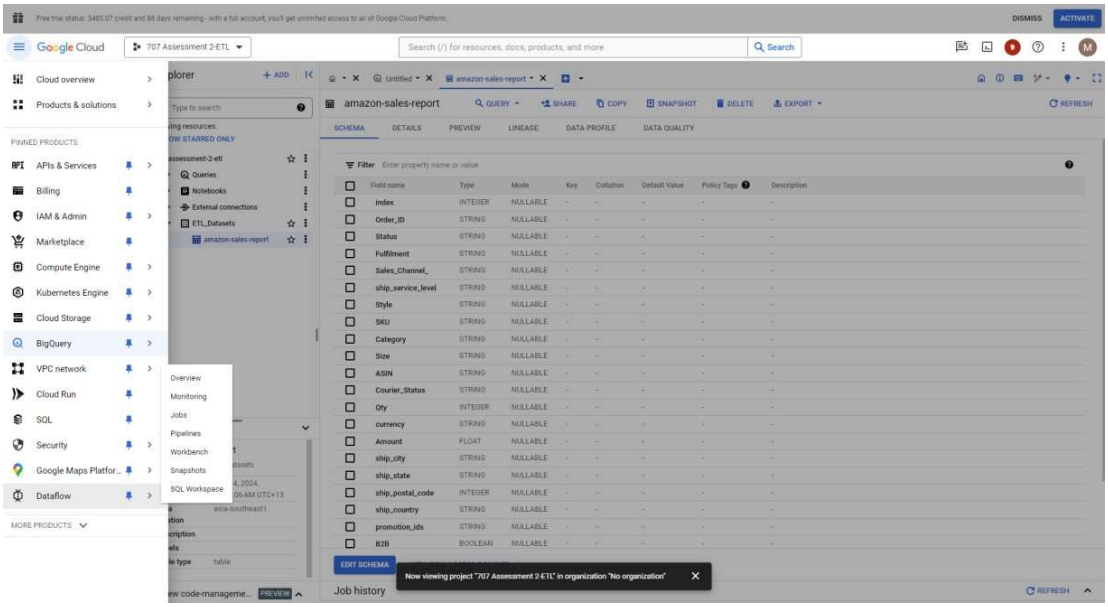
- Provide the needed table name (should match with the table names in MongoDB Atlas)



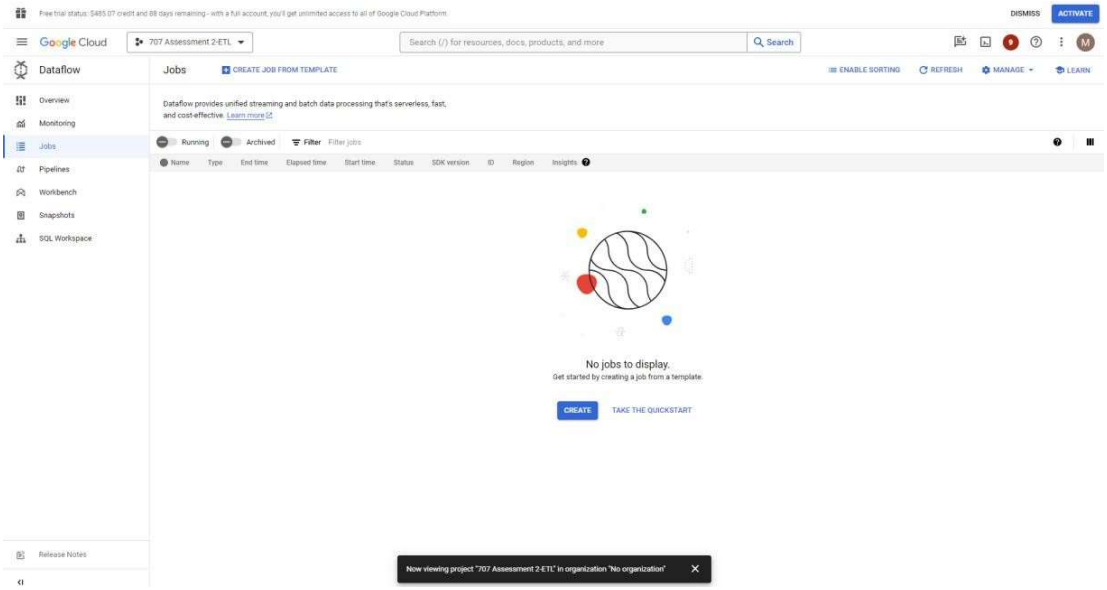
MongoDB Atlas Table (for reference)



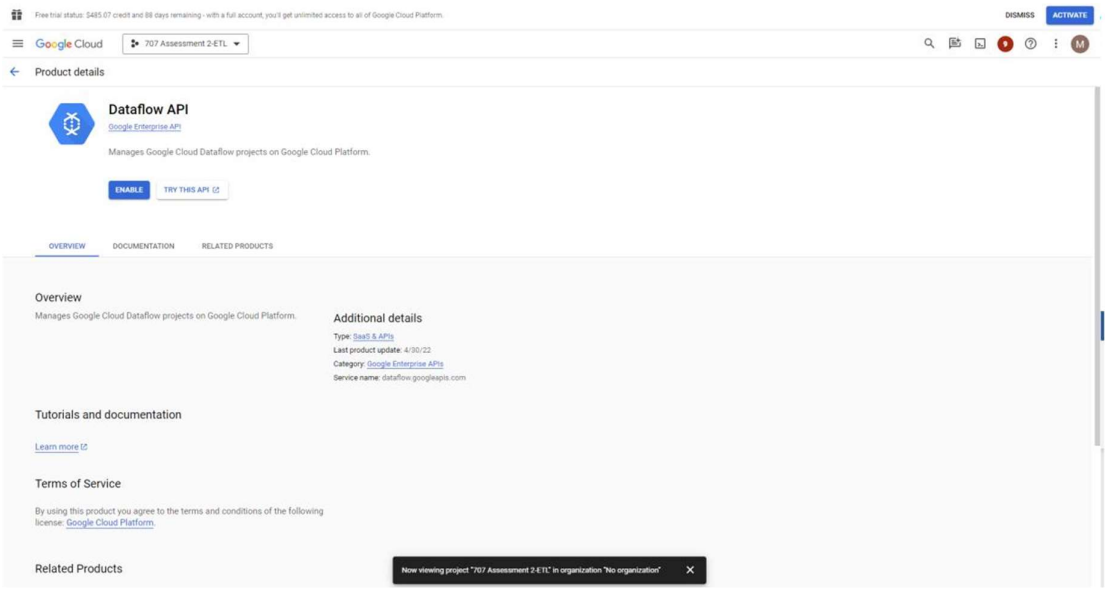
- Once the table is created, you may create a Dataflow job. Go to Dataflow and click on Jobs.



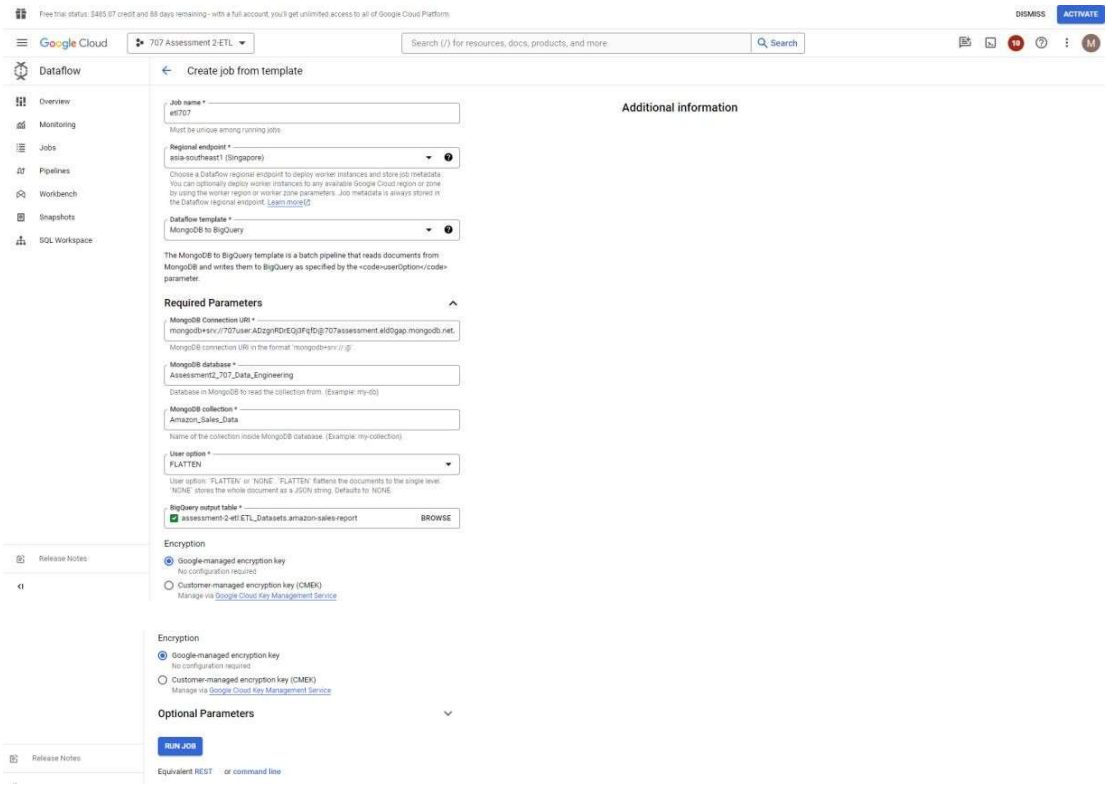
- Click +CREATE JOB FROM TEMPLATE



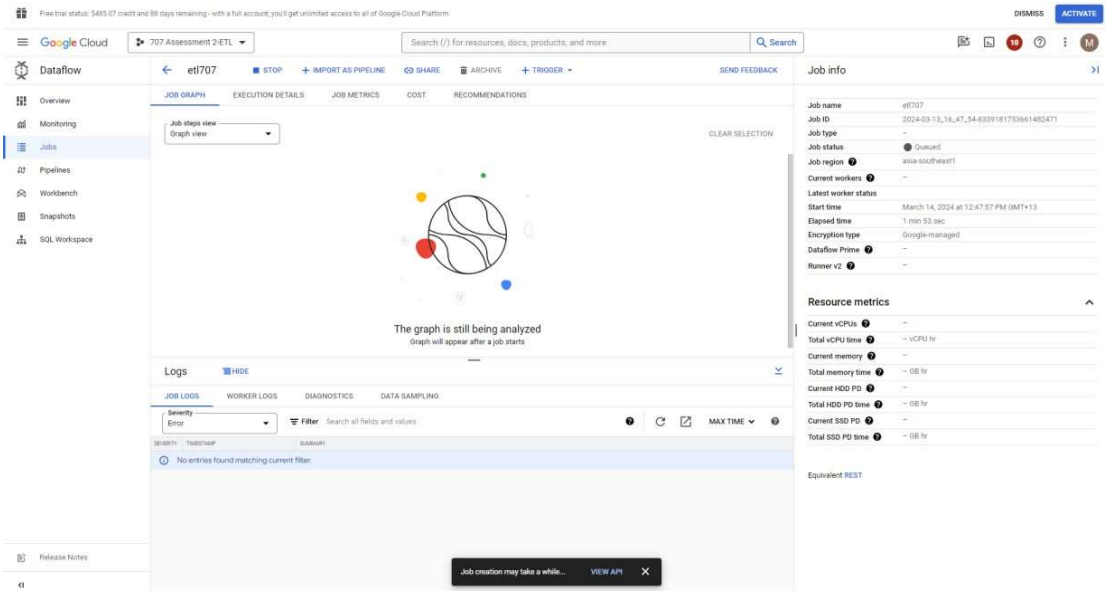
- Click ENABLE Dataflow API



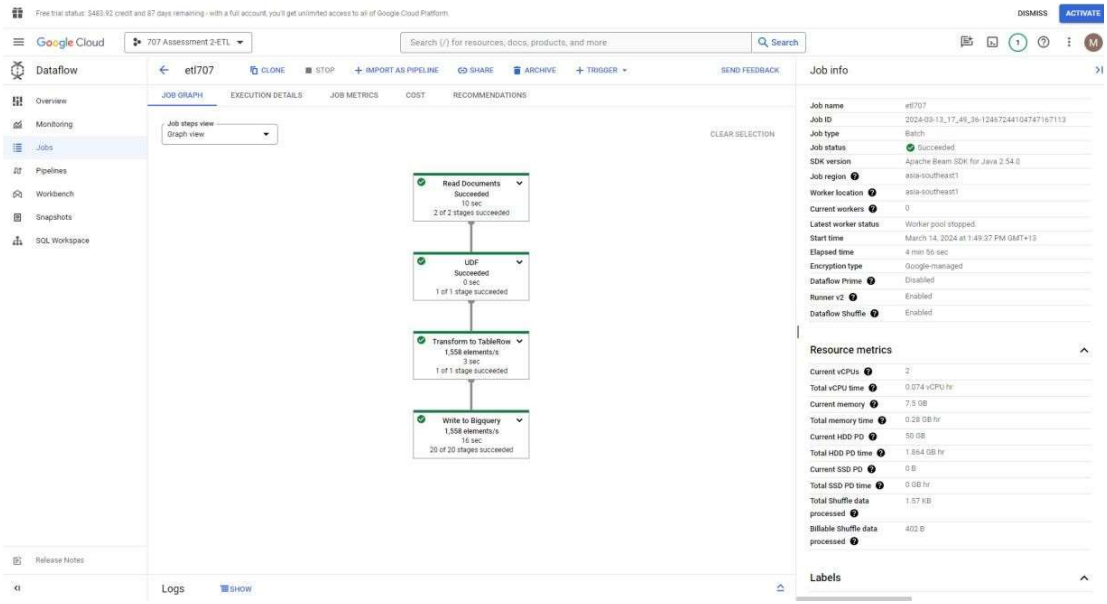
- Create a job from a template



Job creation may take a while.



- Successful ETL process.



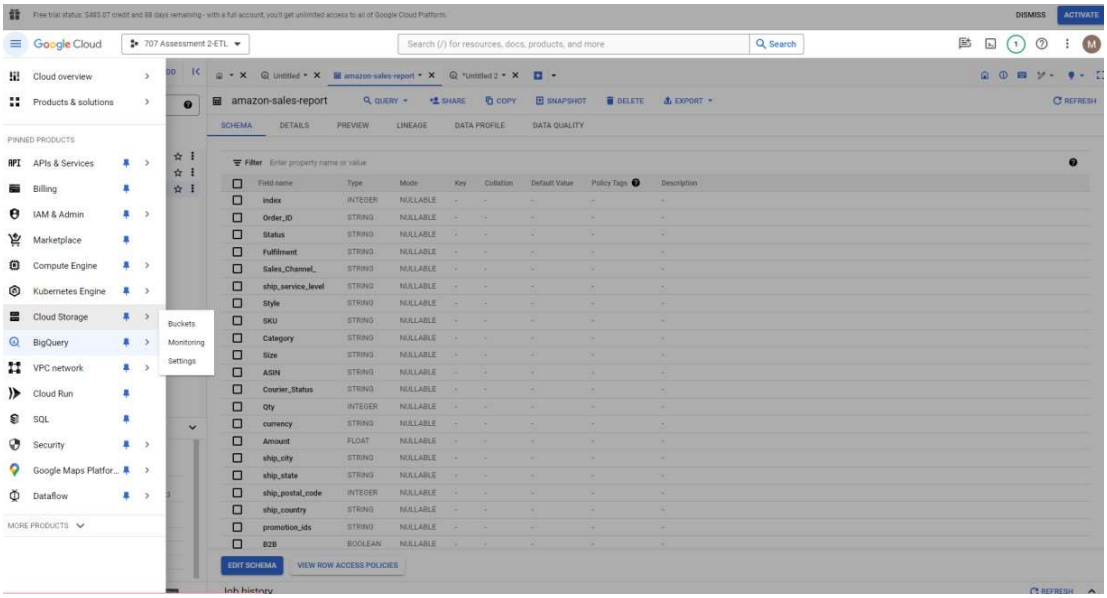
The same process should be used for the second dataset, but this time, I used the GCP bucket as a data source instead of MongoDB Atlas.

Sales Transaction Dataset – From Python to GCP Bucket

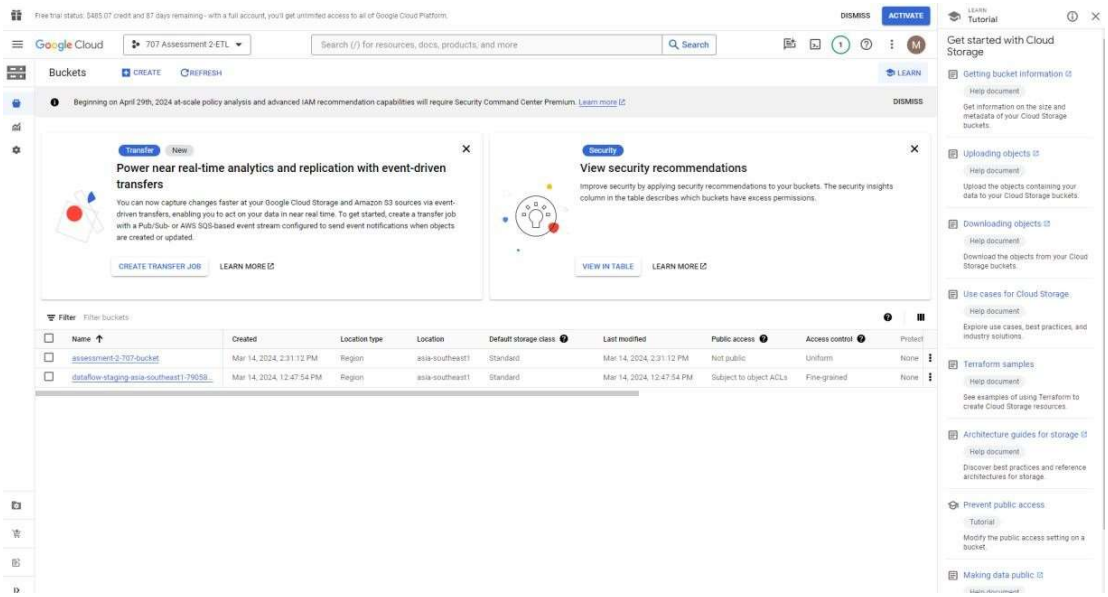
- Import storage library in Python to import the cleaned dataset to GCP Bucket.

```
from google.cloud import storage

# Load service account credentials
credentials_path = 'assessment-2-etl-95c16a484157.json'
storage_client = storage.Client.from_service_account_json(credentials_path)

# Bucket name
bucket_name = 'assessment2_bucket'
# Get the bucket
bucket = storage_client.get_bucket(bucket_name)

# Specify the remote filename in the bucket
blob = bucket.blob('Cleaned_SalesTransactionDF.csv')

# Upload the local file
blob.upload_from_filename('./Cleaned_SalesTransactionDF.csv')
```

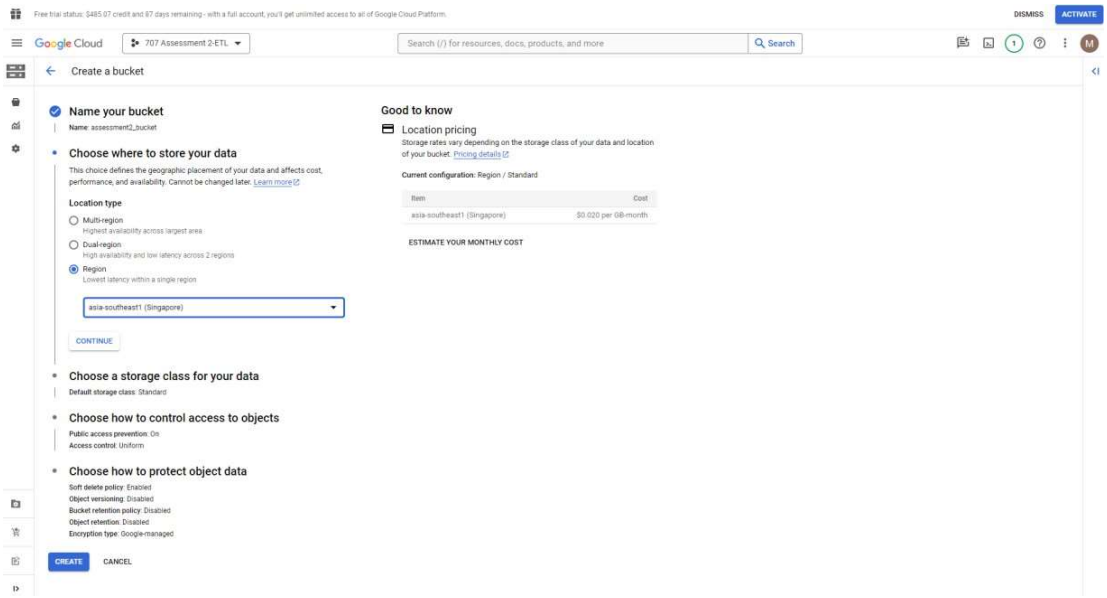
- Create a Bucket in GCP – in the navigation menu, go to Cloud Storage and click Buckets.



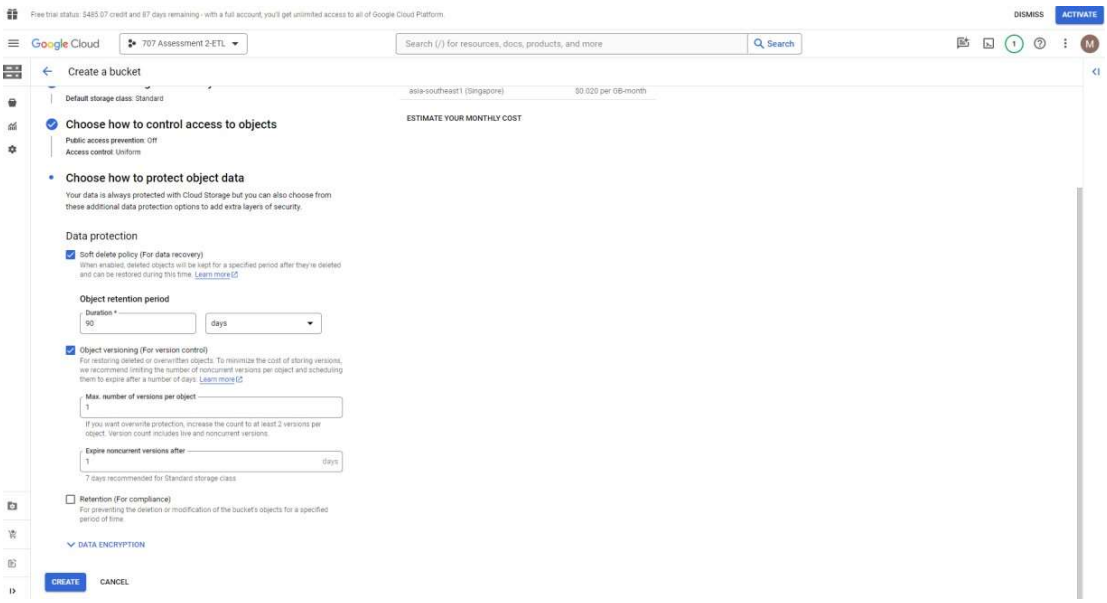
- On the upper left side, click +CREATE.



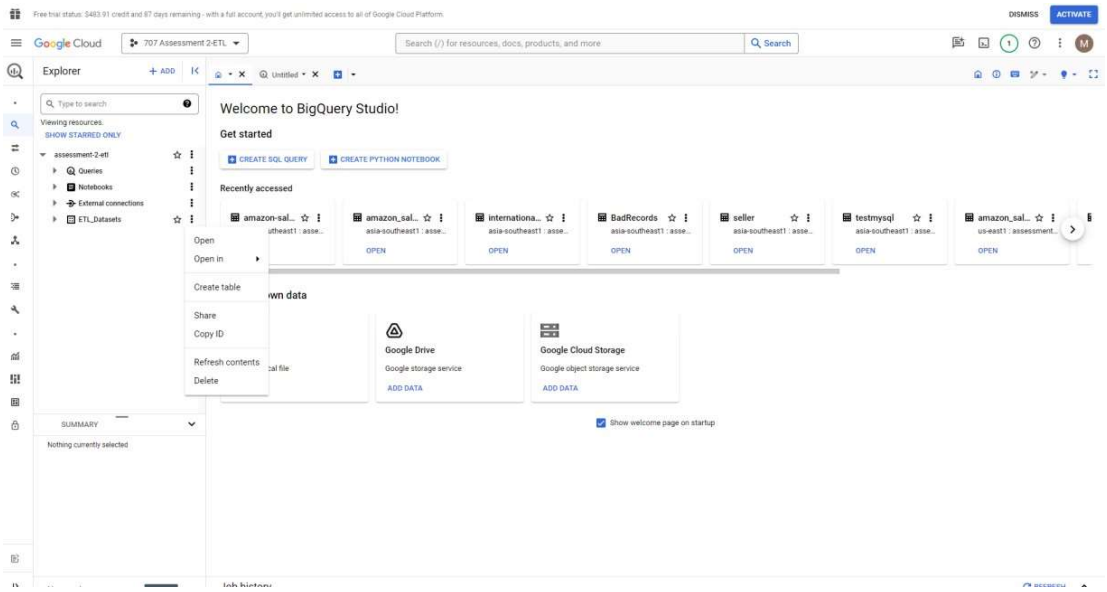
* Provide the needed details.



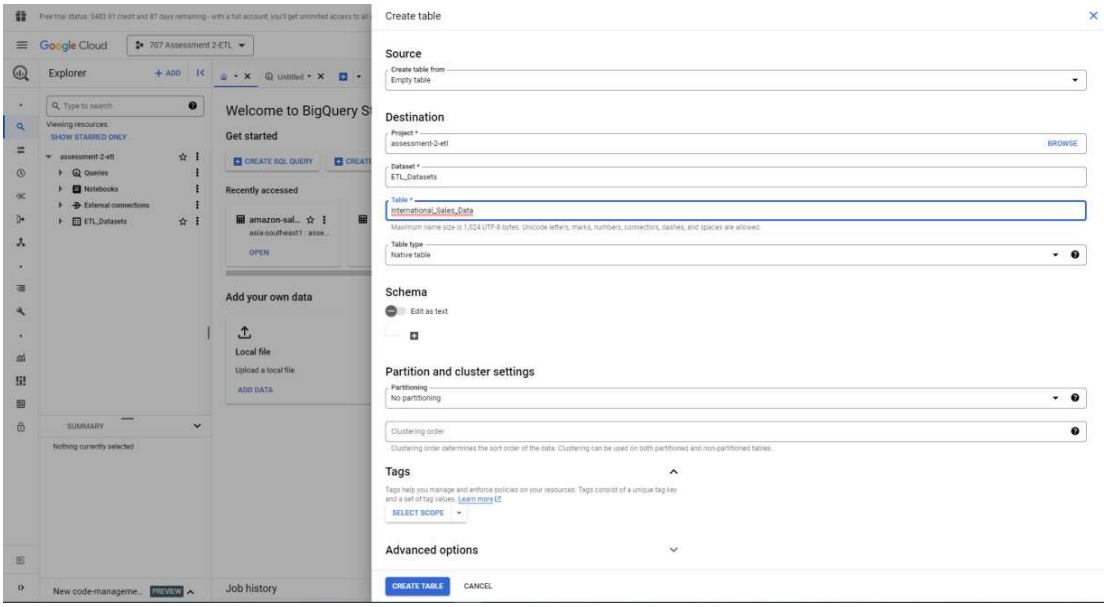
Click CONTINUE and CREATE



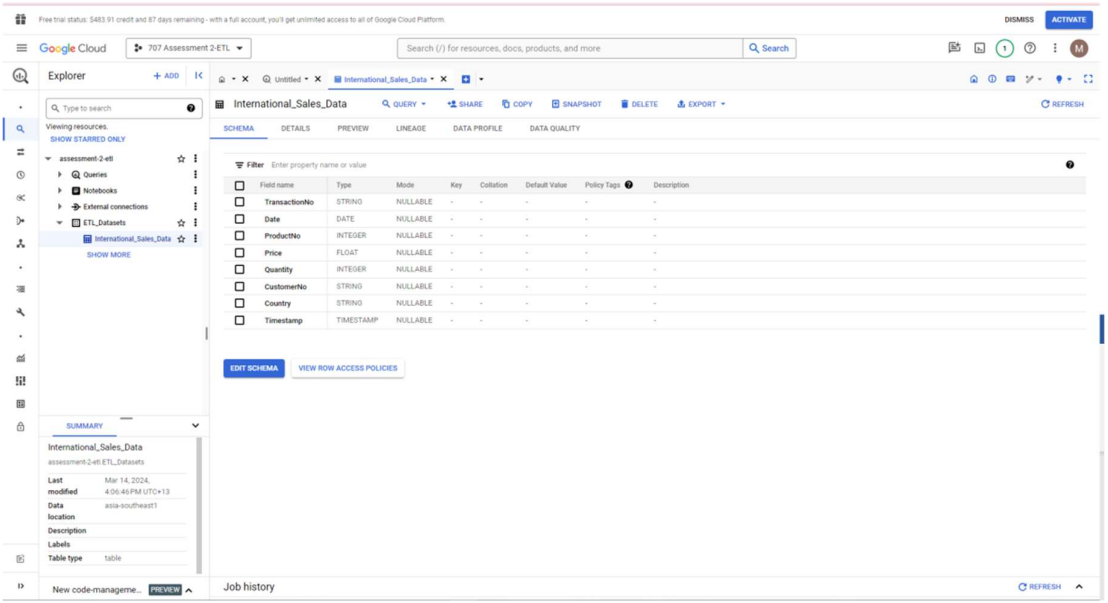
- After creating a bucket, create a table.



Naming the table

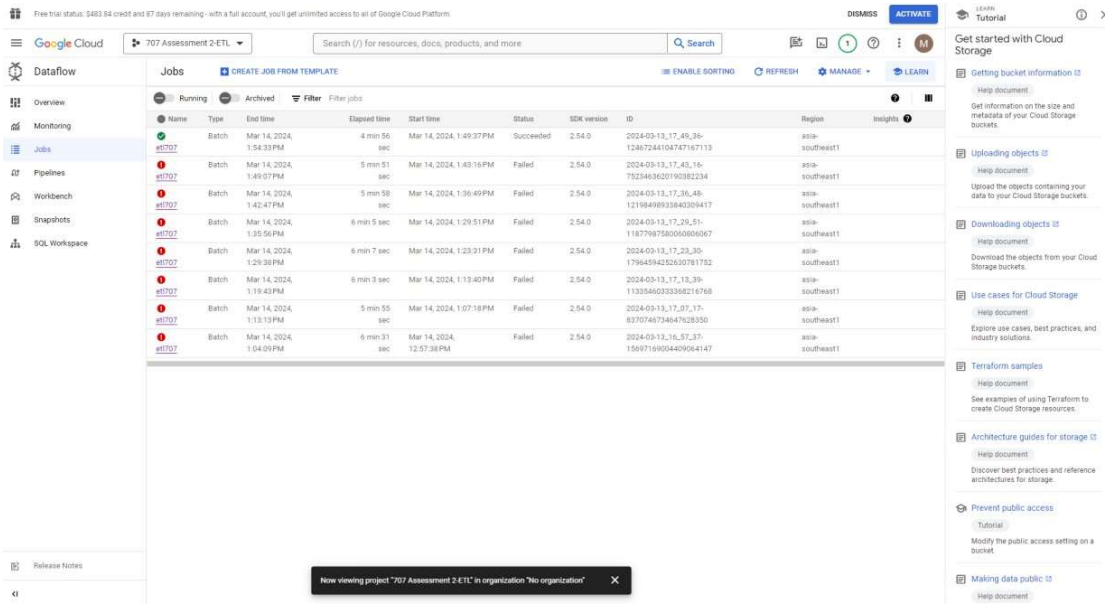


- Go to the table and EDIT SCHEMA

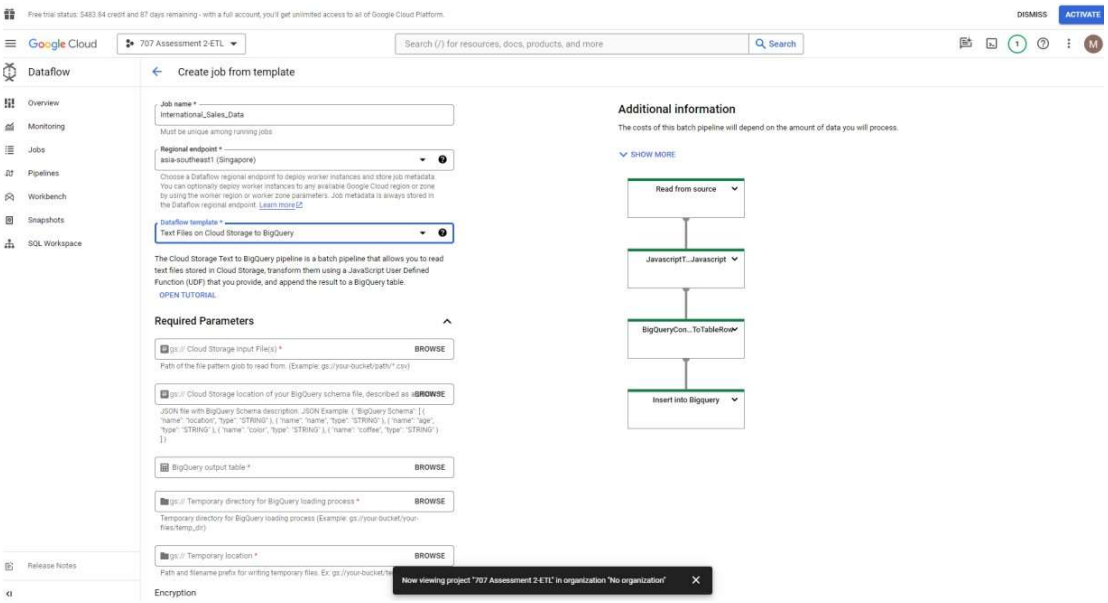


After creating the table in the BigQuery, we will now create a Job using the Dataflow template text file to BigQuery.

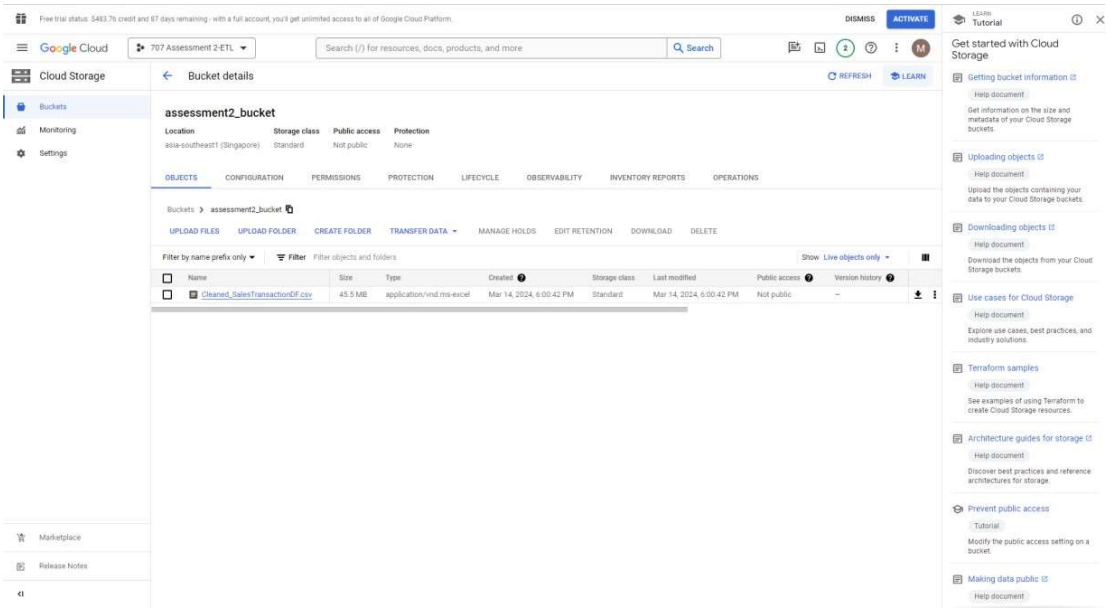
Go to Dataflow, CREATE JOB FROM TEMPLATE



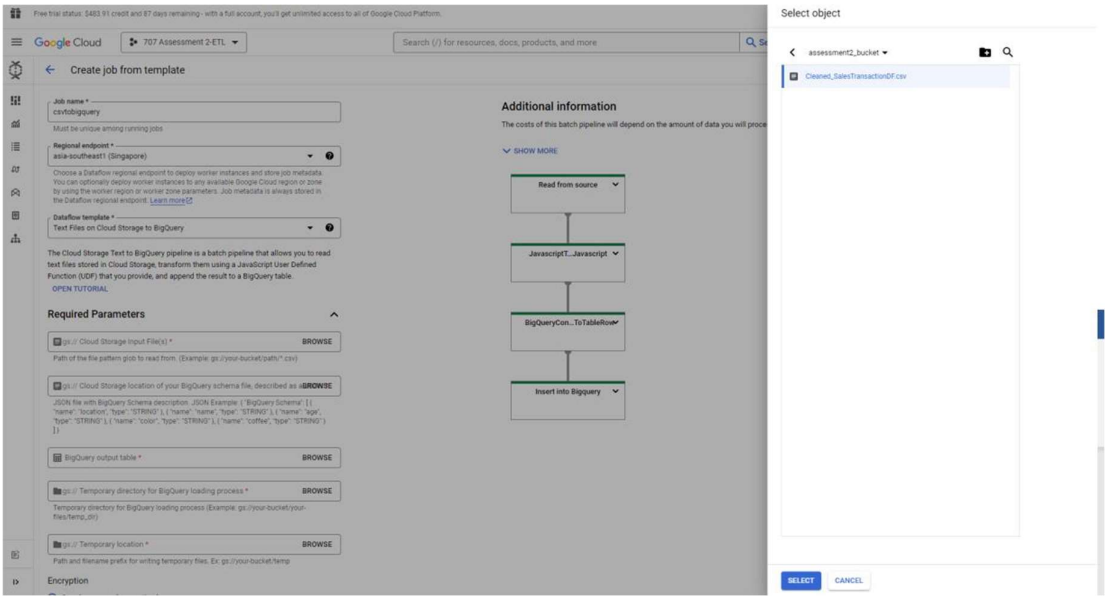
- Creating a job from a template.



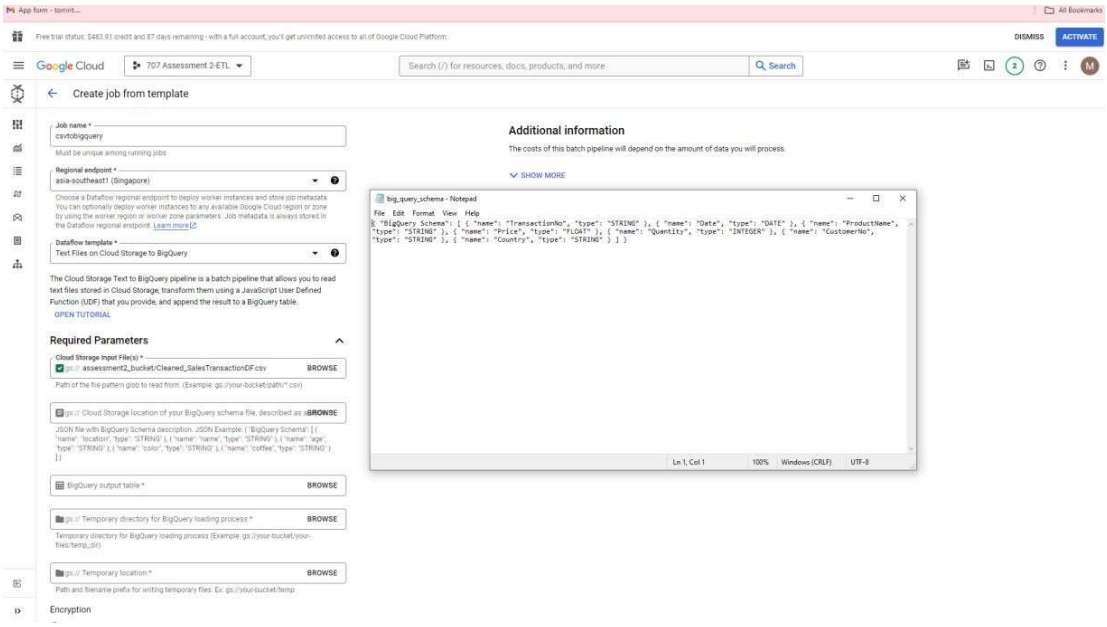
Uploaded the cleaned dataset



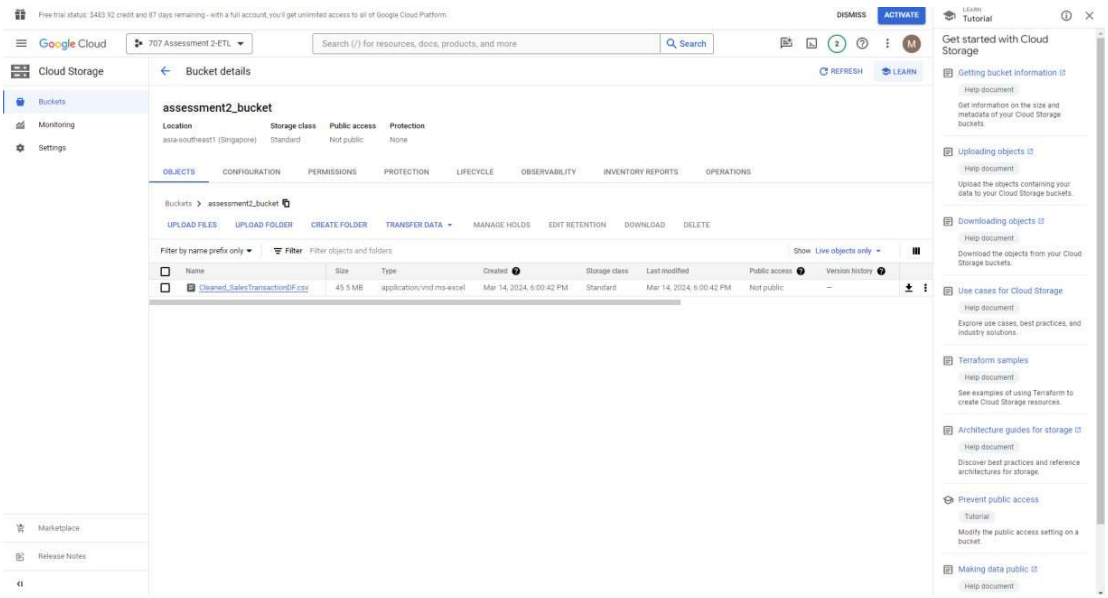
- Creating a Job template for source file (first parameter)

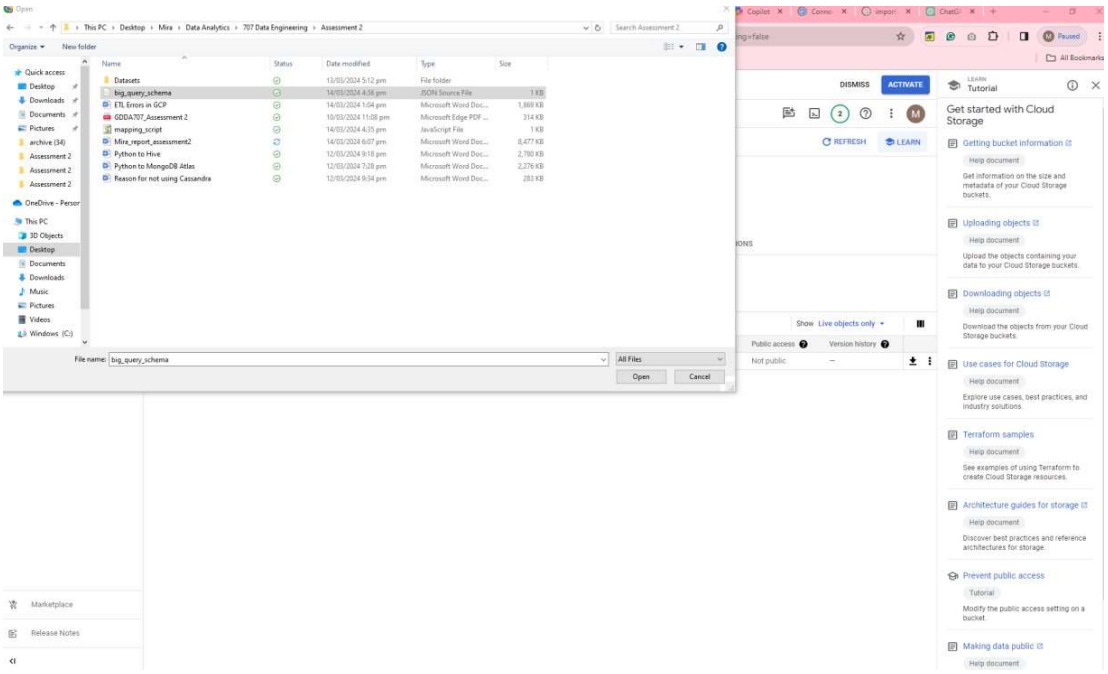


- Mapping between CSV and BigQuery

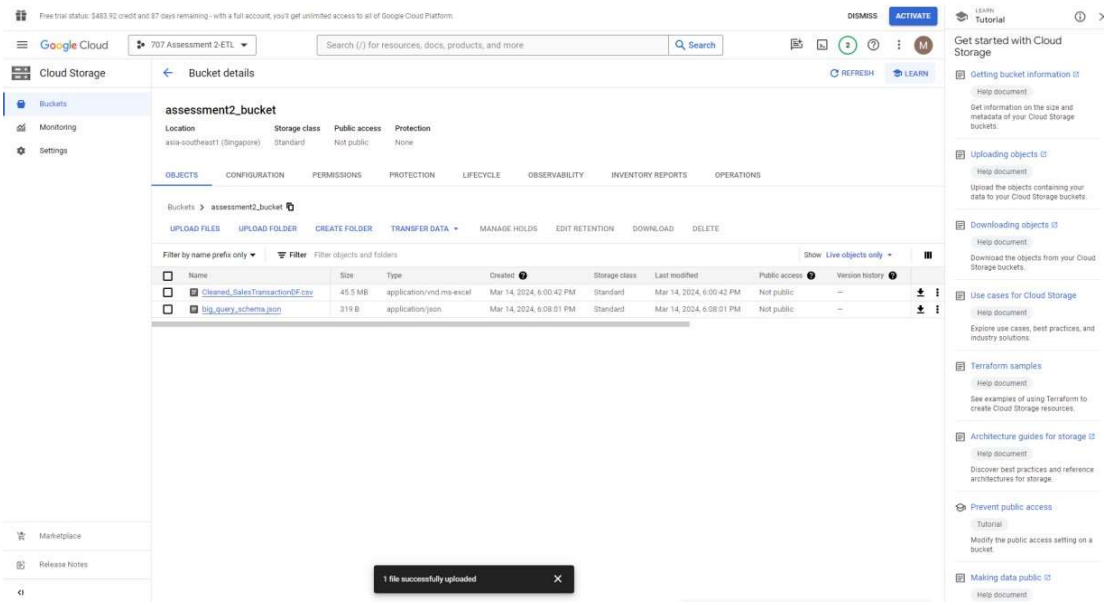


- Upload the JSON file to the bucket

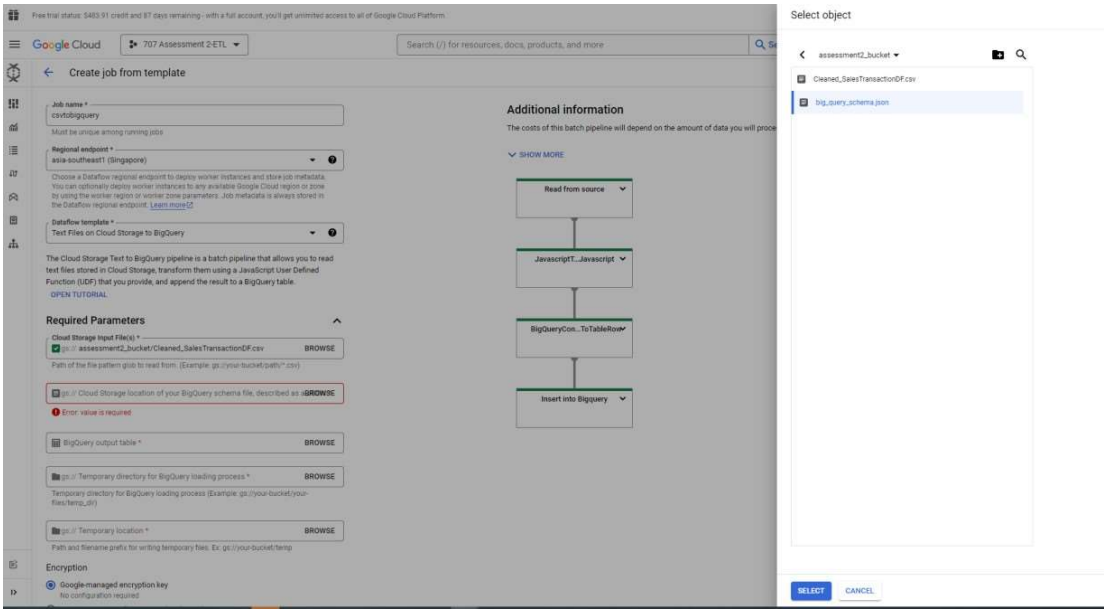




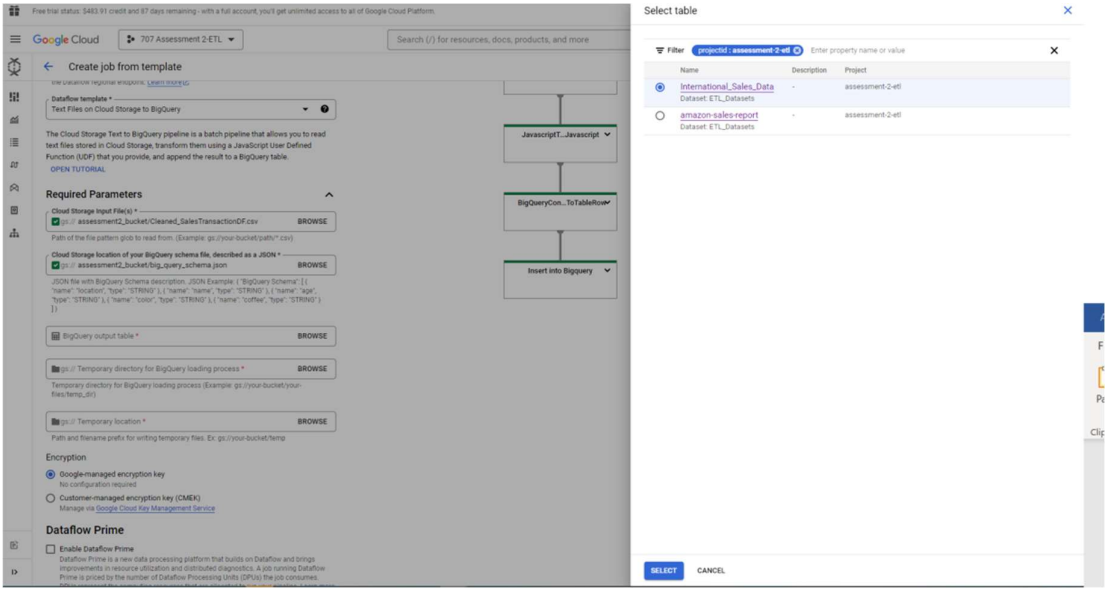
- Checking if it was successfully uploaded



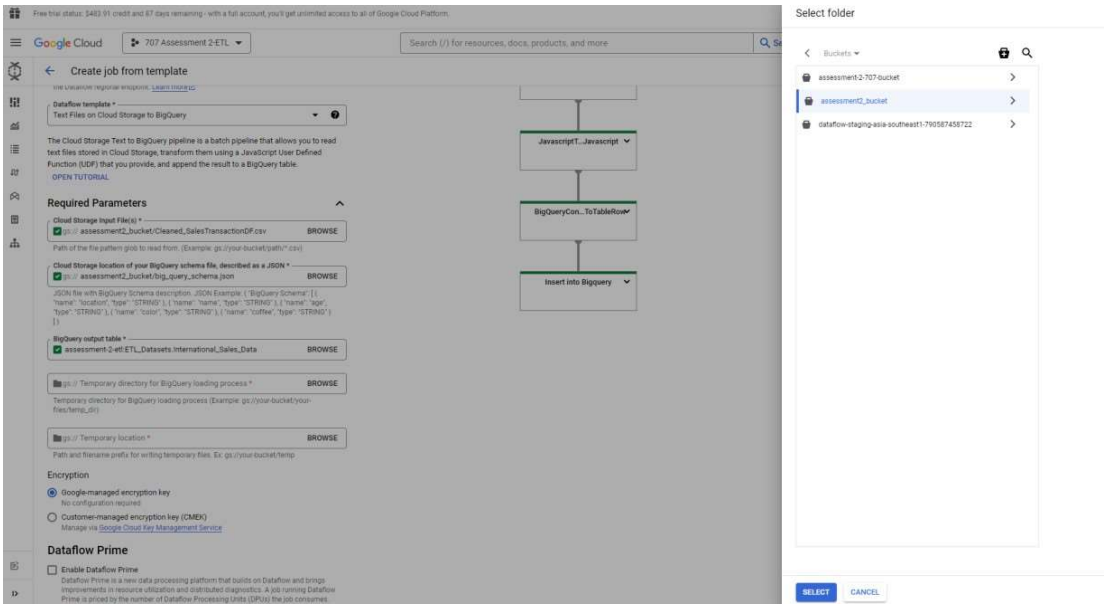
- Go back to Dataflow create job. Select the big_query_schema (second parameter)



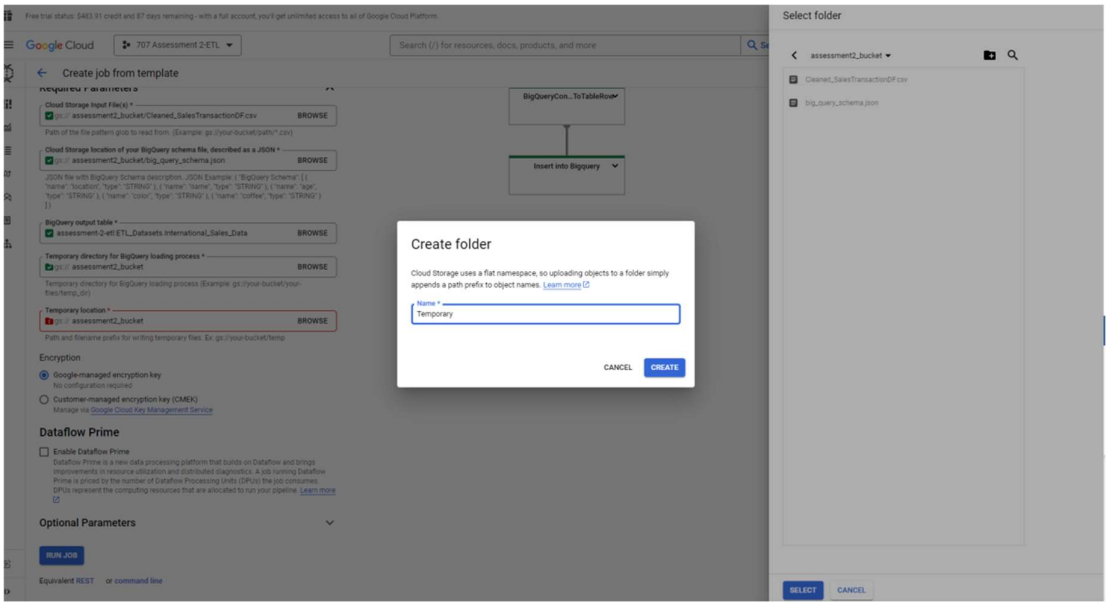
- Loading the transformed data from CSV to BigQuery. Select the dataset. (third parameter)



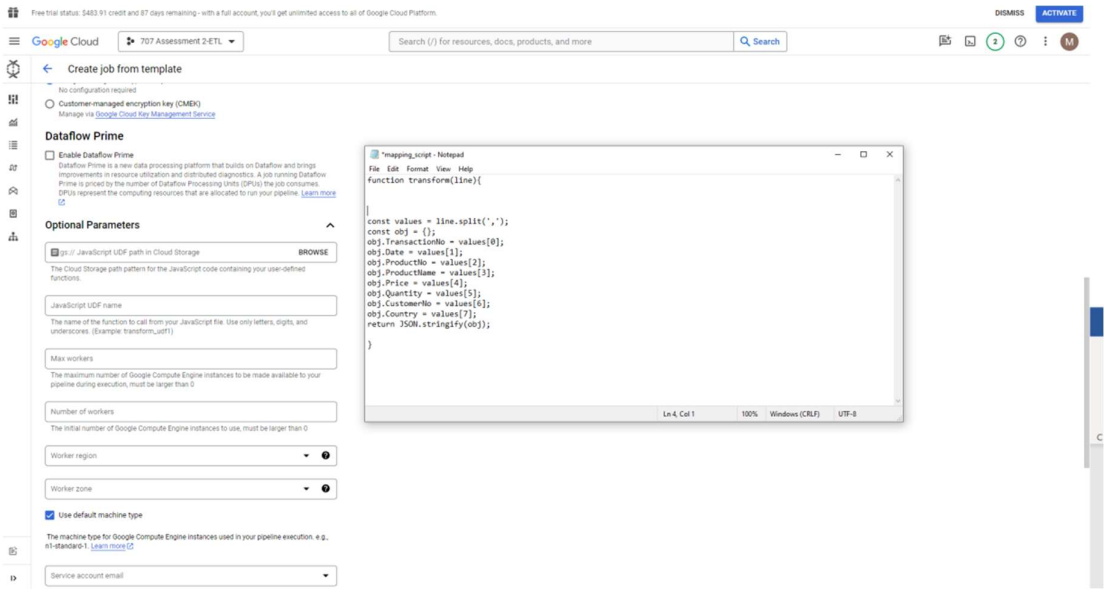
- Select the Bucket to be used as temporary location for loading process



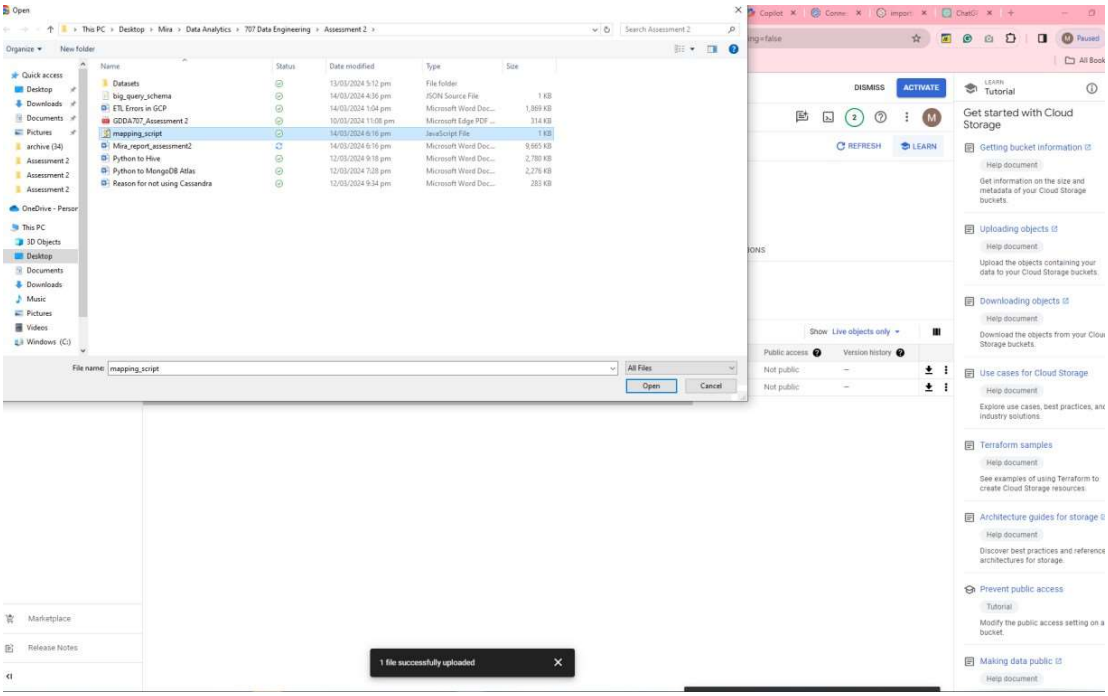
- Create temporary location for writing temporary files needed in the loading process



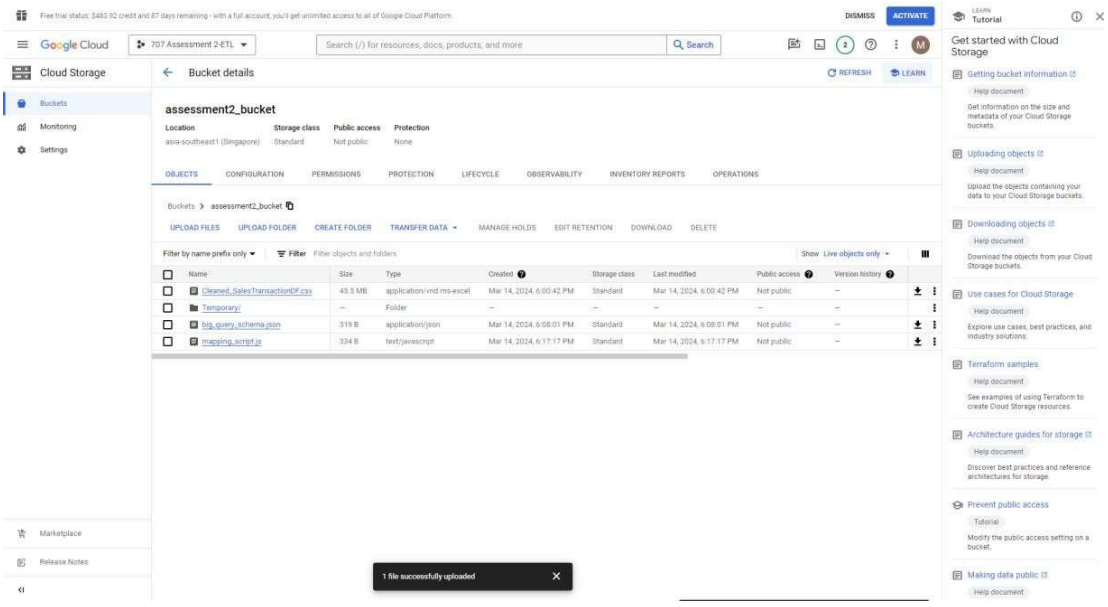
- Click Optional Parameters and go to JavaScript UDF name



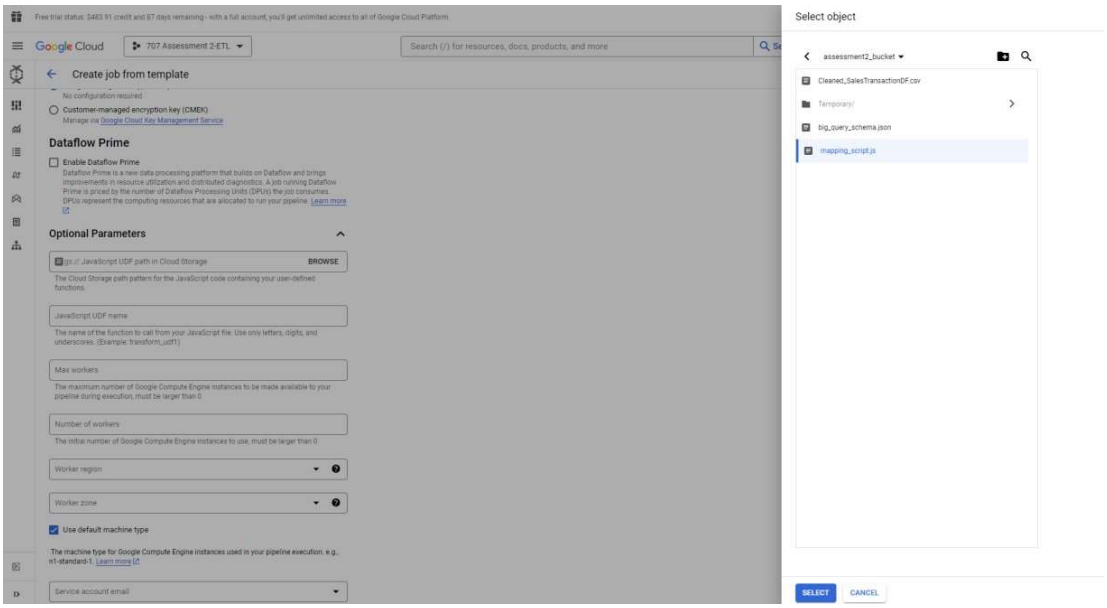
- Go to the Bucket and upload the mapping script file



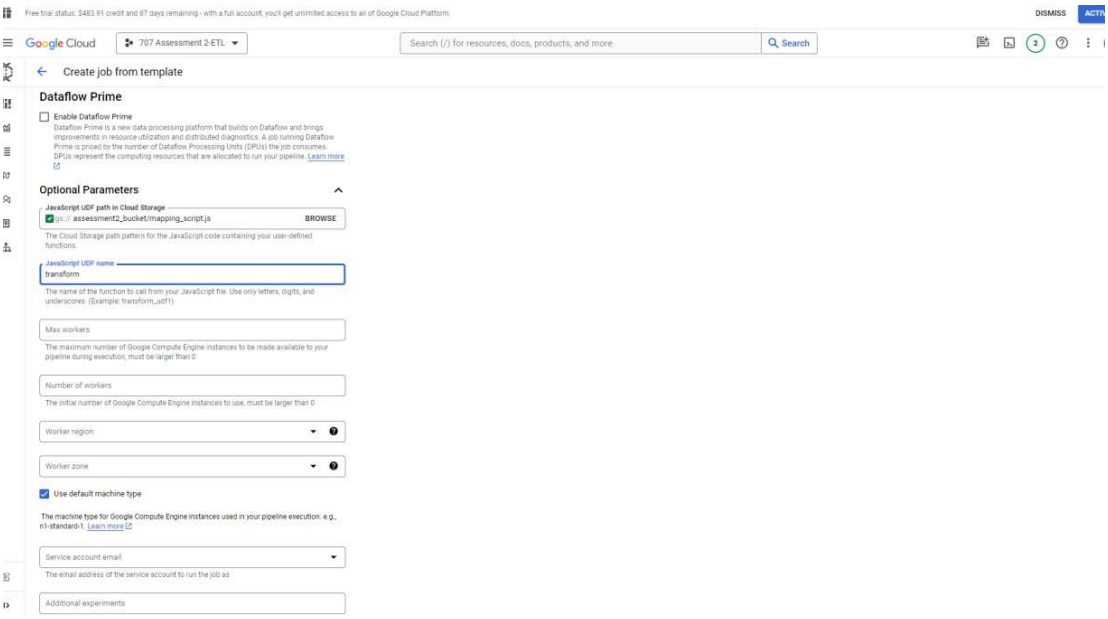
- Check if successfully uploaded



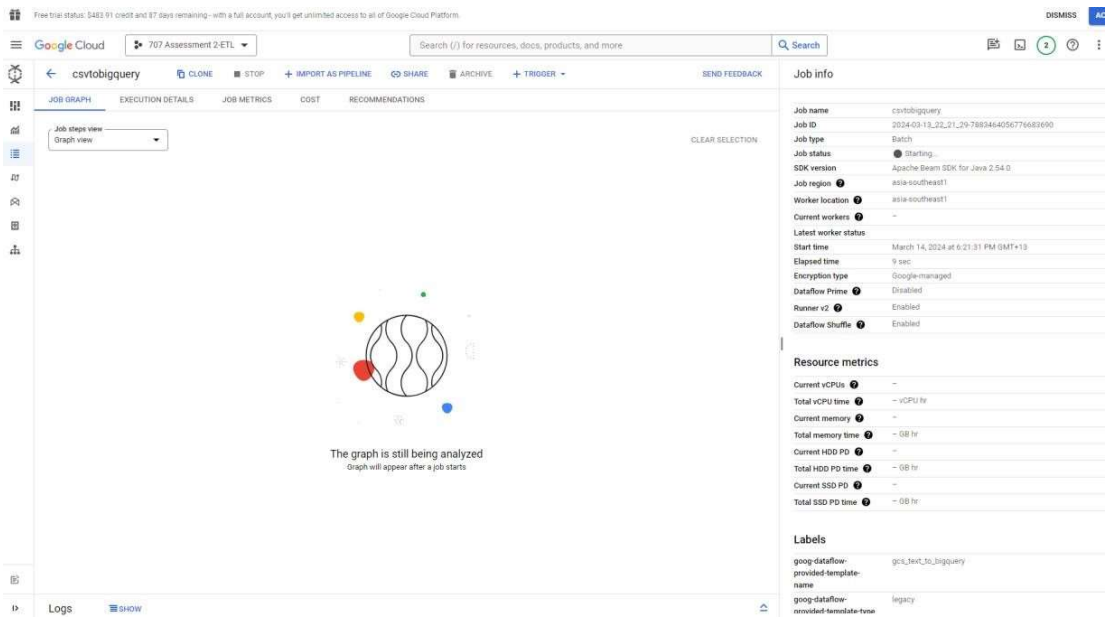
- Go back to the job creation page under Optional Parameters, browse and choose the mapping_script.js



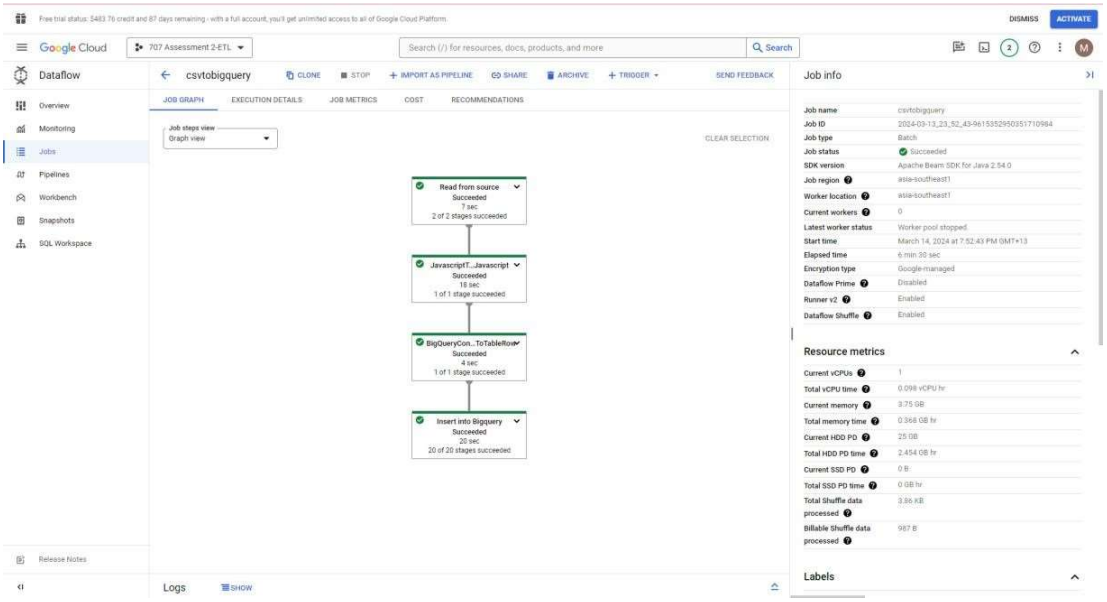
- Type ‘transform’ in the JavaScript UDF name and click RUN JOB



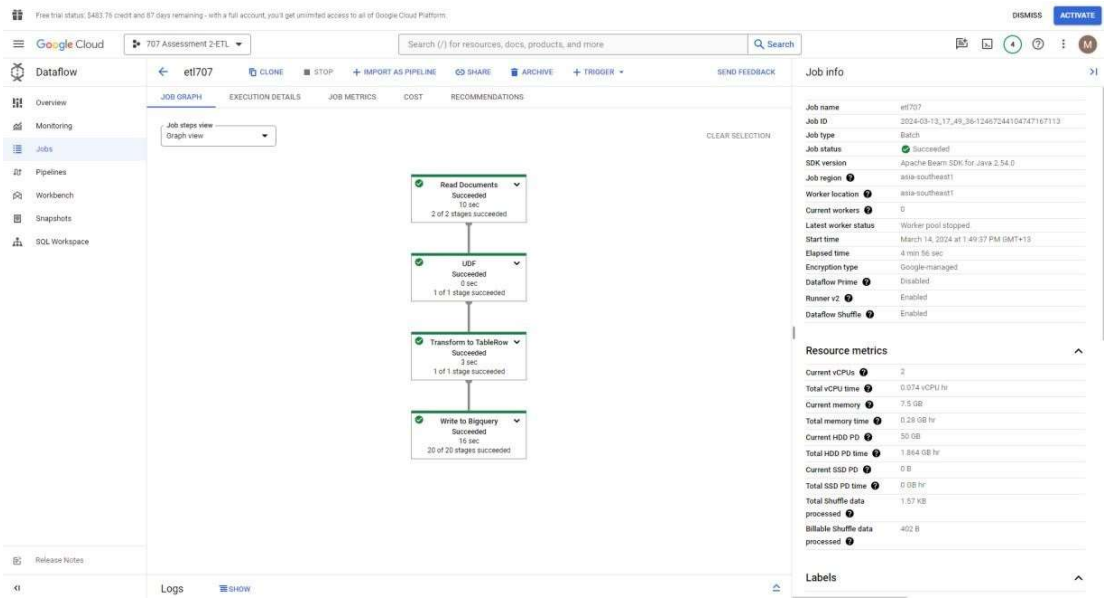
- Running the job



- Successfully done ETL

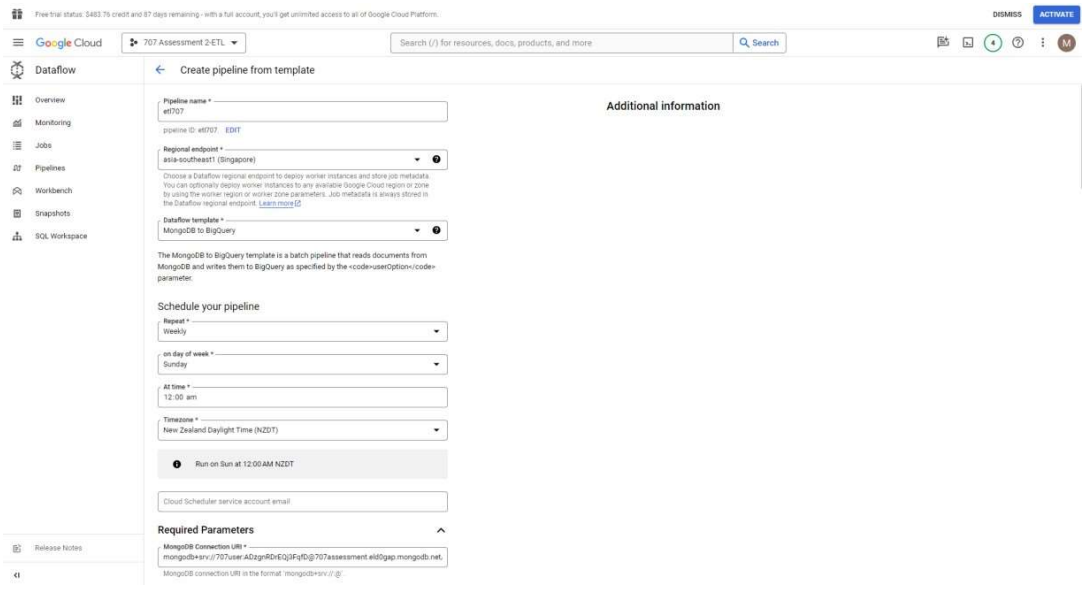


- Successfully created ETL

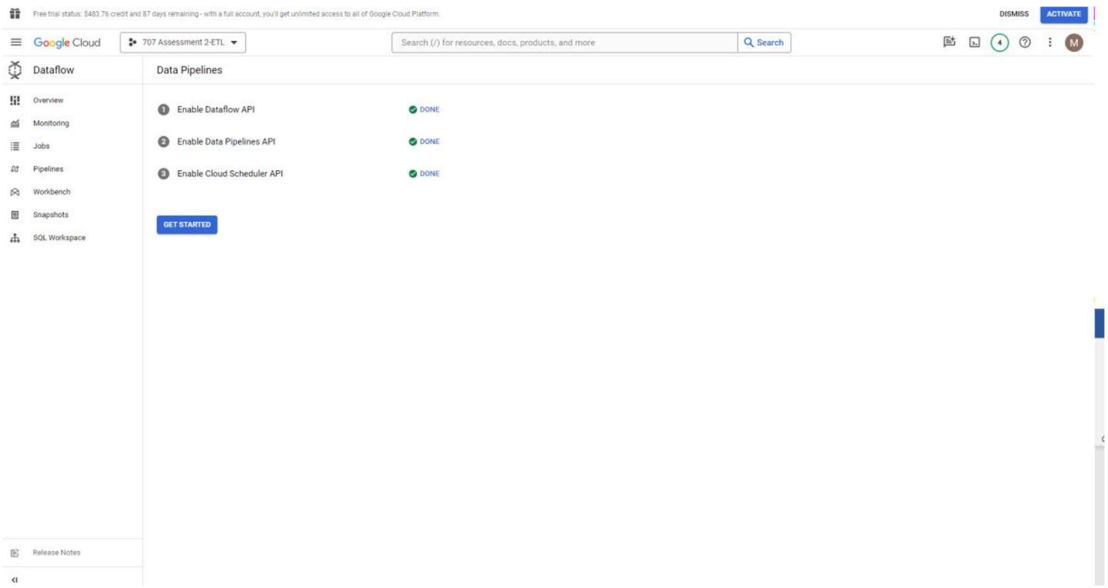


Creating a Pipeline to run a scheduled job

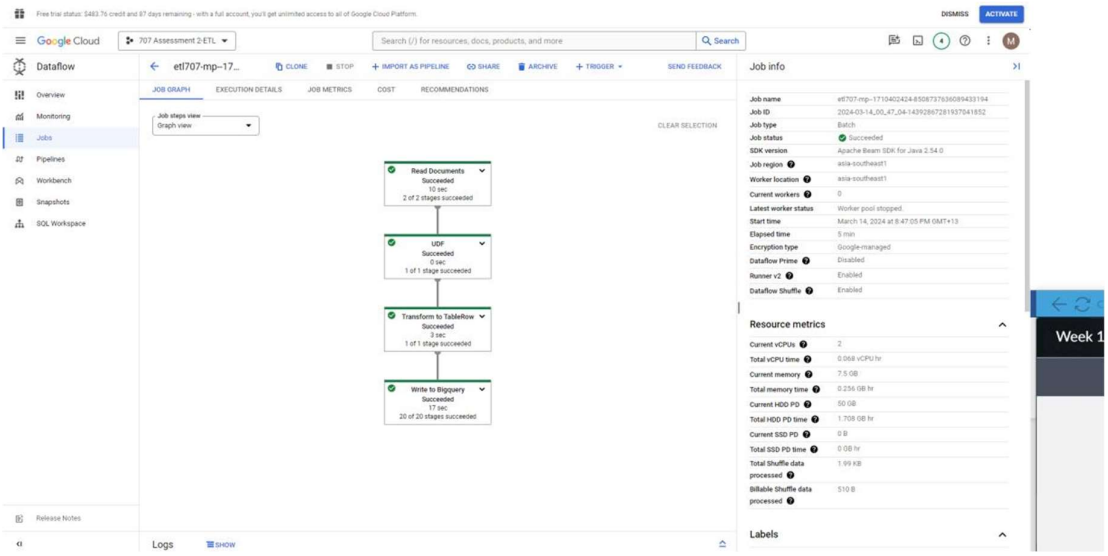
- On the same previous page, click +IMPORT AS PIPELINE, then click CREATE PIPELINE



- Running the Pipeline



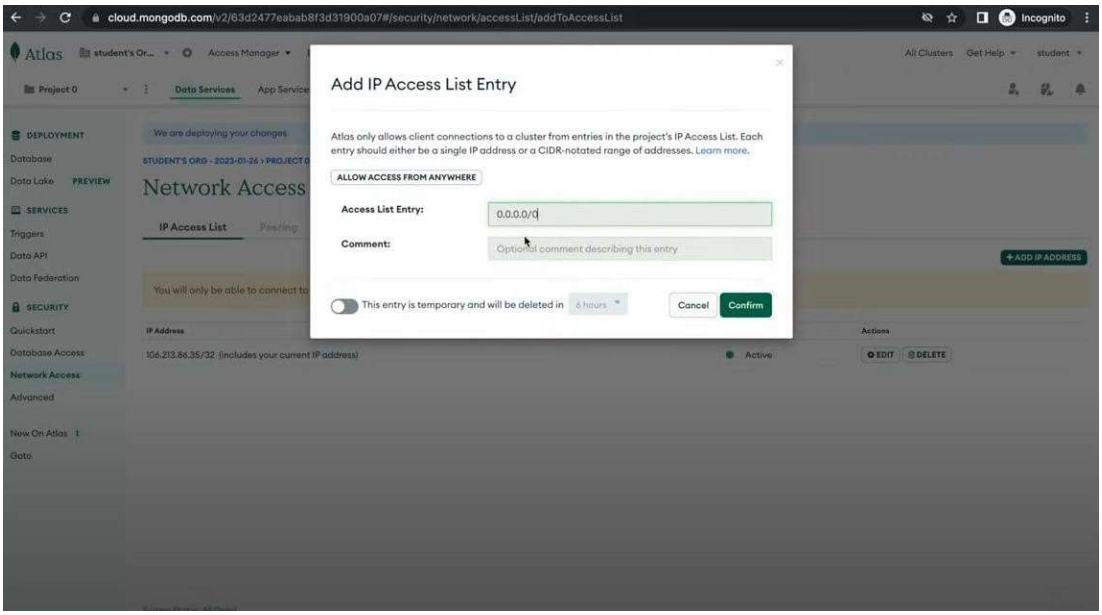
- Successfully created Pipeline

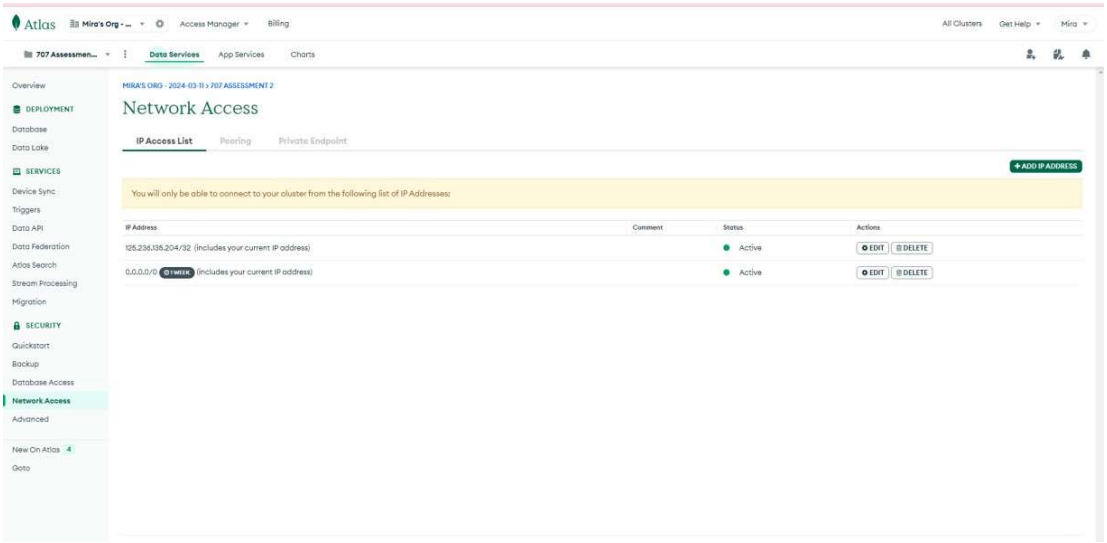


Challenges during ETL and Integration

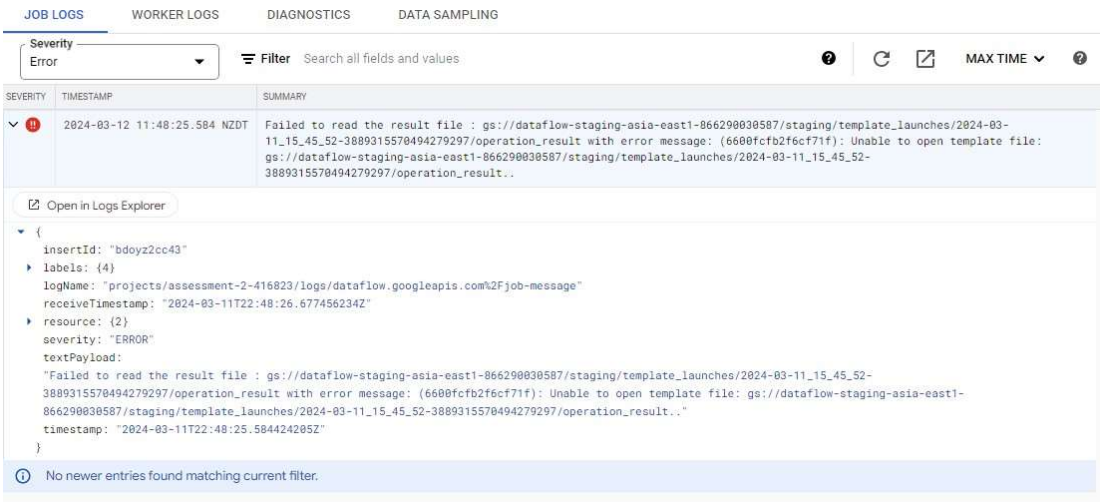
Some of the challenges I encountered are the following:

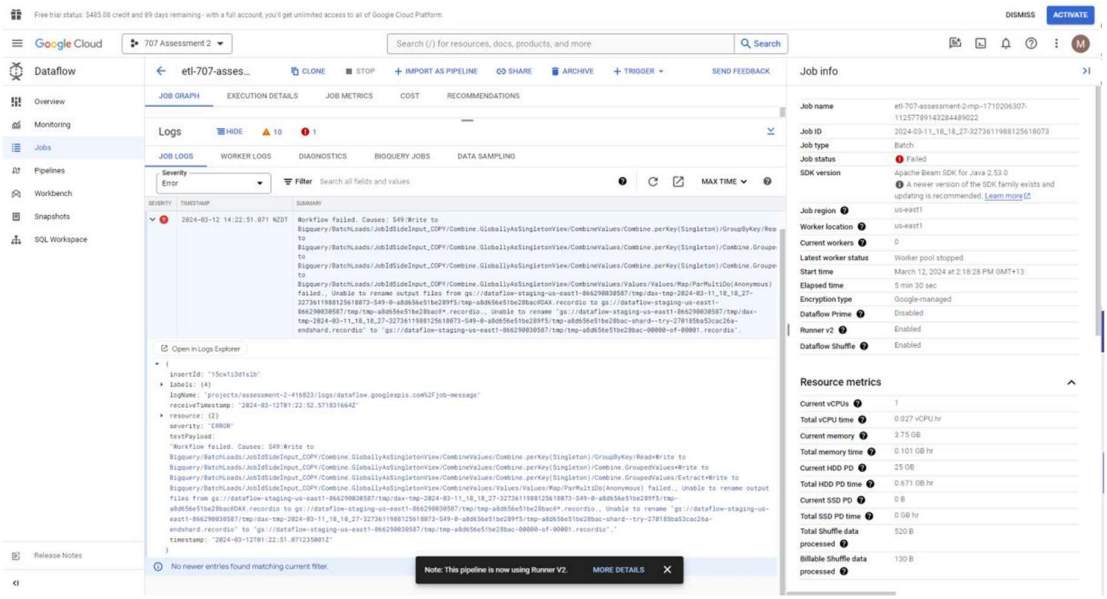
1. MongoDB connection to GCP – I added the IP address 0.0.0.0/0 in the MongoDB Atlas to allow access from anywhere since I could not determine the GCP IP address.

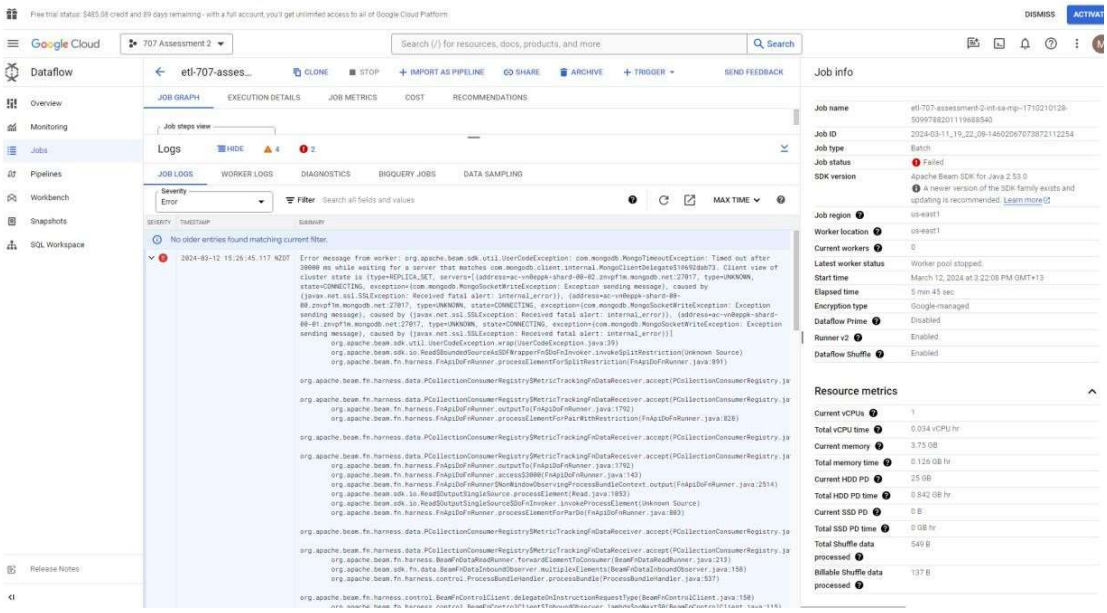


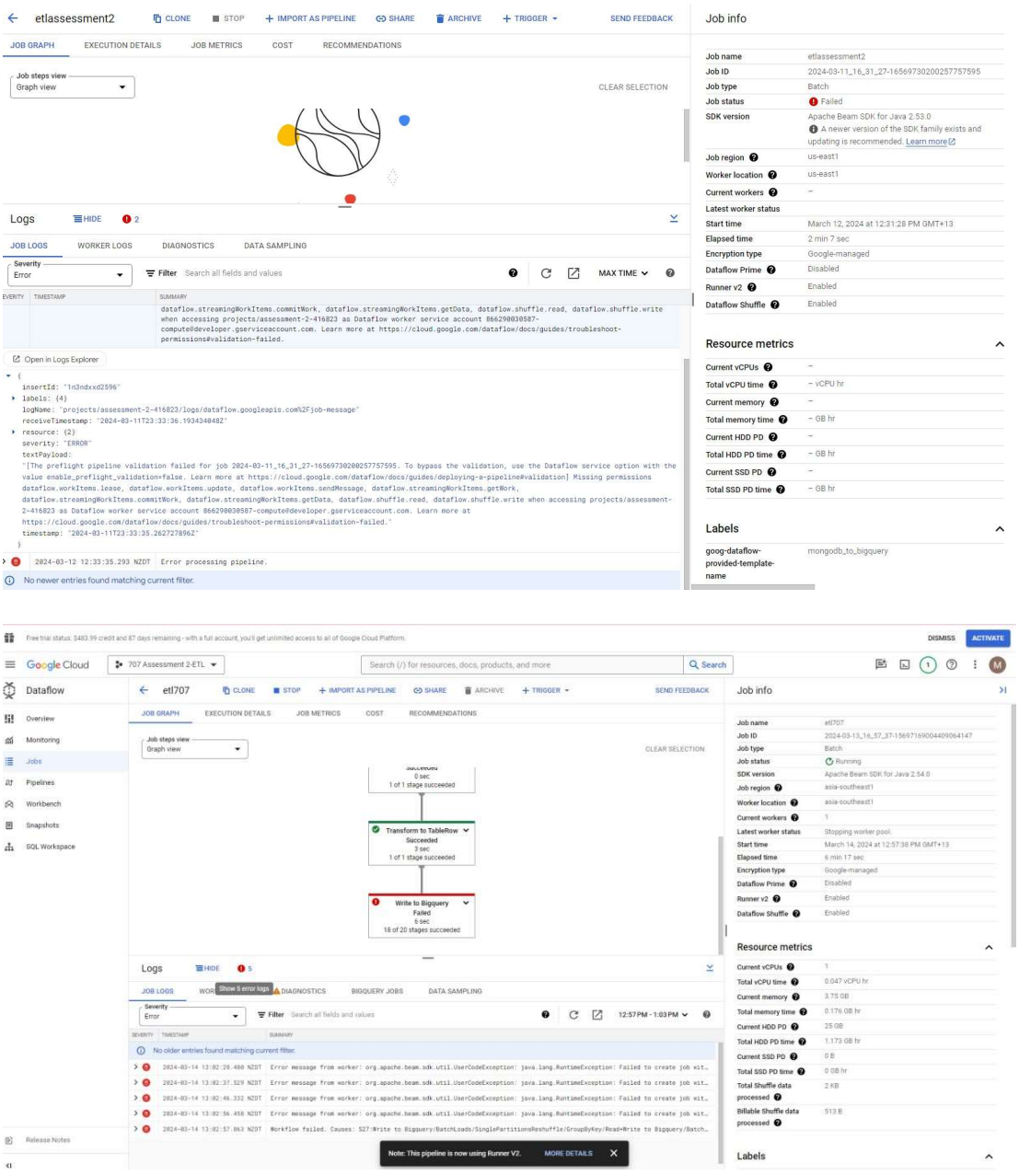


2. There are many errors in the Dataflow job creation in GCP due to schema, connections, and job settings. To solve these problems, I double-checked the field names in MongoDB and BigQuery and made sure they had the same schema. On the connections, I made sure that I provided the correct role, credentials (username and password), and the correct server address (connection strings).On the job settings, I made sure to choose the correct Dataflow templates and correct parameters.

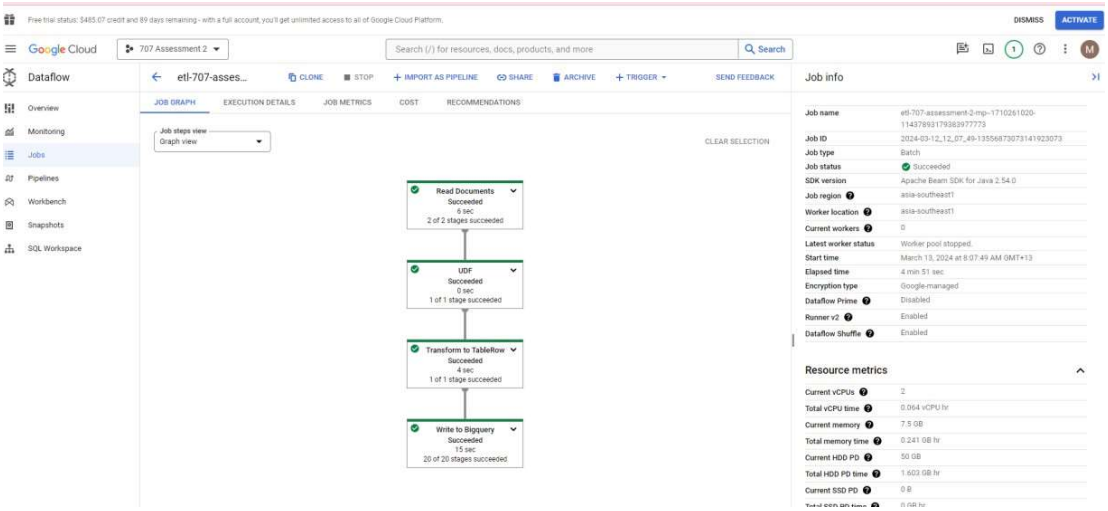

JOB LOGS
WORKER LOGS
DIAGNOSTICS
DATA SAMPLING
Severity
Error
Filter
Search all fields and values
MAX TIME
SEVERITY
TIMESTAMP
SUMMARY
2024-03-12 11:48:25.584 NZDT
Failed to read the result file : gs://dataflow-staging-asia-east1-866290030587/staging/template_launches/2024-03-11_15_45_52-3889315570494279297/operation_result with error message: (6600fcfb2f6cf71f): Unable to open template file: gs://dataflow-staging-asia-east1-866290030587/staging/template_launches/2024-03-11_15_45_52-3889315570494279297/operation_result..
Open in Logs Explorer
{
insertId: "bdoyz2cc43"
labels: {4}
logName: "projects/assessment-2-416823/logs/dataflow.googleapis.com%2Fjob-message"
receiveTimestamp: "2024-03-11T22:48:26.677456234Z"
resource: {2}
severity: "ERROR"
textPayload:
"Failed to read the result file : gs://dataflow-staging-asia-east1-866290030587/staging/template_launches/2024-03-11_15_45_52-3889315570494279297/operation_result with error message: (6600fcfb2f6cf71f): Unable to open template file: gs://dataflow-staging-asia-east1-866290030587/staging/template_launches/2024-03-11_15_45_52-3889315570494279297/operation_result.."
timestamp: "2024-03-11T22:48:25.584424205Z"
}
No newer entries found matching current filter.

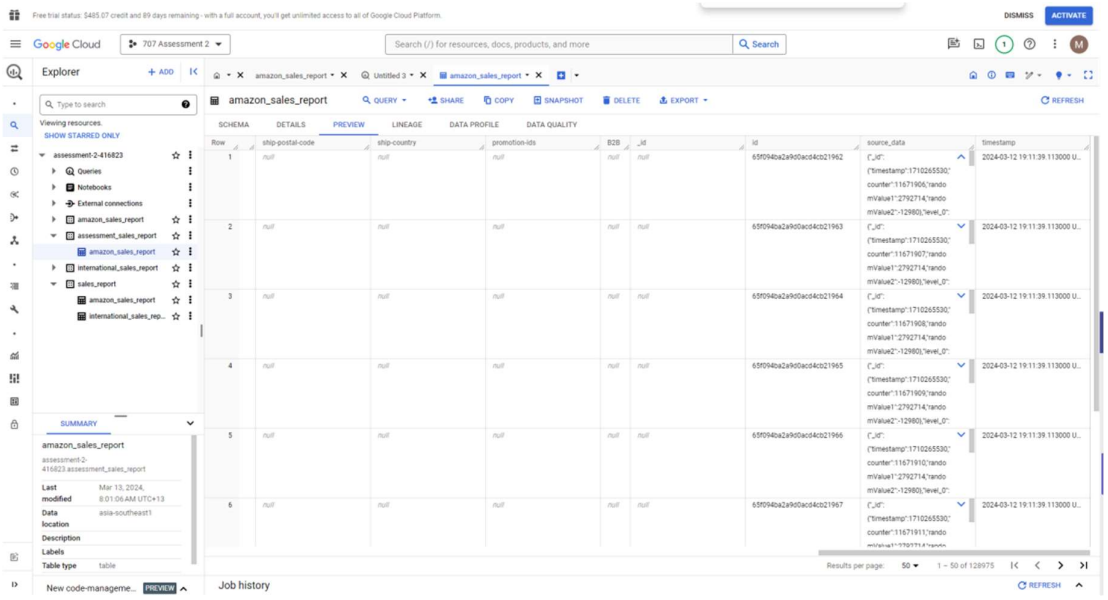

Google Cloud
Dataflow
etl-707-asses...
Job info
Resource metrics
Note: This pipeline is now using Runner V2.
MORE DETAILS


Google Cloud
Dataflow
etl-707-asses...
Job info
Resource metrics



3. ETL was successful but displayed null values in the PREVIEW table. – To solve this problem, I checked the job settings and changed the user option from NONE to Flatten.

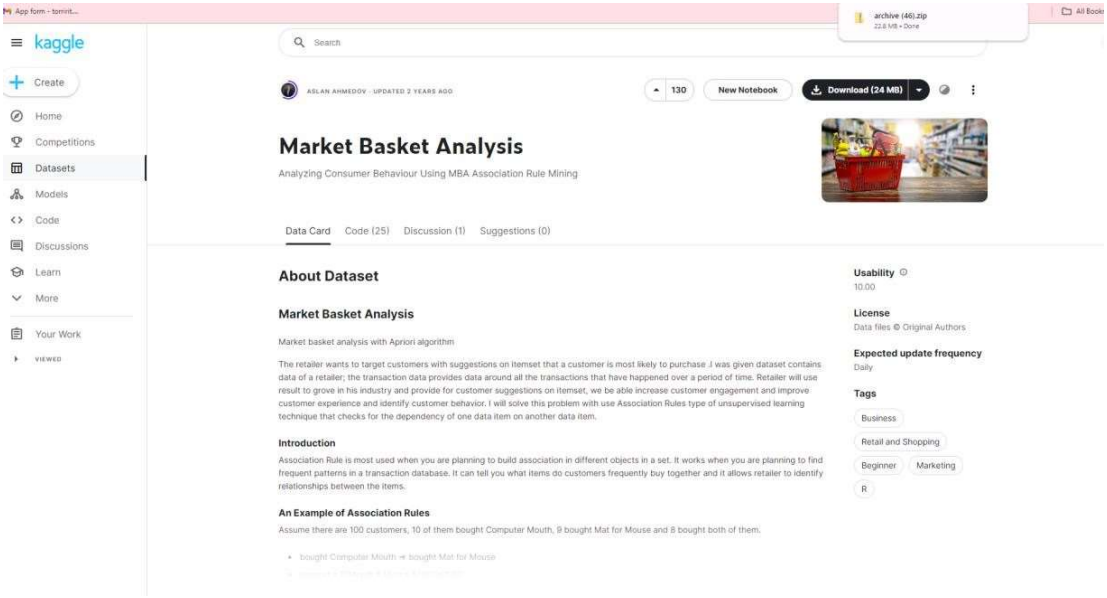

Dataflow
etl-707-asses...
CLONE
STOP
IMPORT AS PIPELINE
SHARE
ARCHIVE
TRIGGER
SEND FEEDBACK
JOB GRAPH
EXECUTION DETAILS
JOB METRICS
COST
RECOMMENDATIONS
Read Documents
Succeeded
6 sec
2 of 2 stages succeeded
UDF
Succeeded
0 sec
1 of 1 stage succeeded
Transform to TableRow
Succeeded
4 sec
1 of 1 stage succeeded
Write to Bigquery
Succeeded
15 sec
20 of 20 stages succeeded
Job info
Resource metrics


Explorer
amazon_sales_report
SCHEMA
DETAILS
PREVIEW
LINEAGE
DATA PROFILE
DATA QUALITY
SUMMARY
Job history

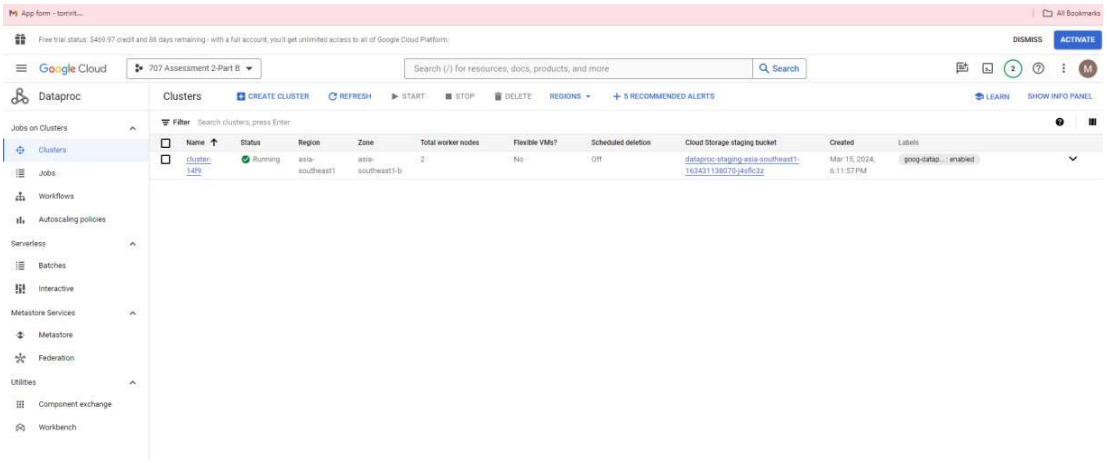
Part B: Big Data Analysis and Application of Engineering Techniques

Task 1: Data Ingestion Pipeline

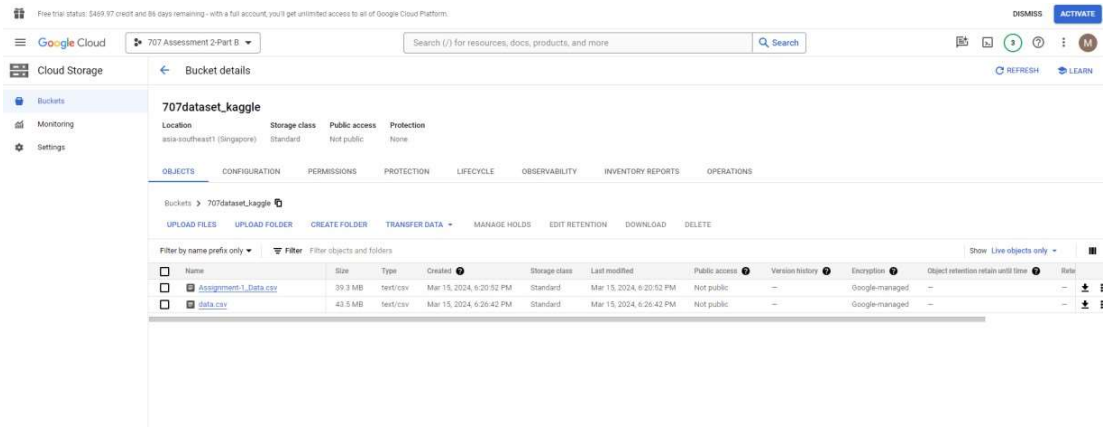
1. Download File from Kaggle.



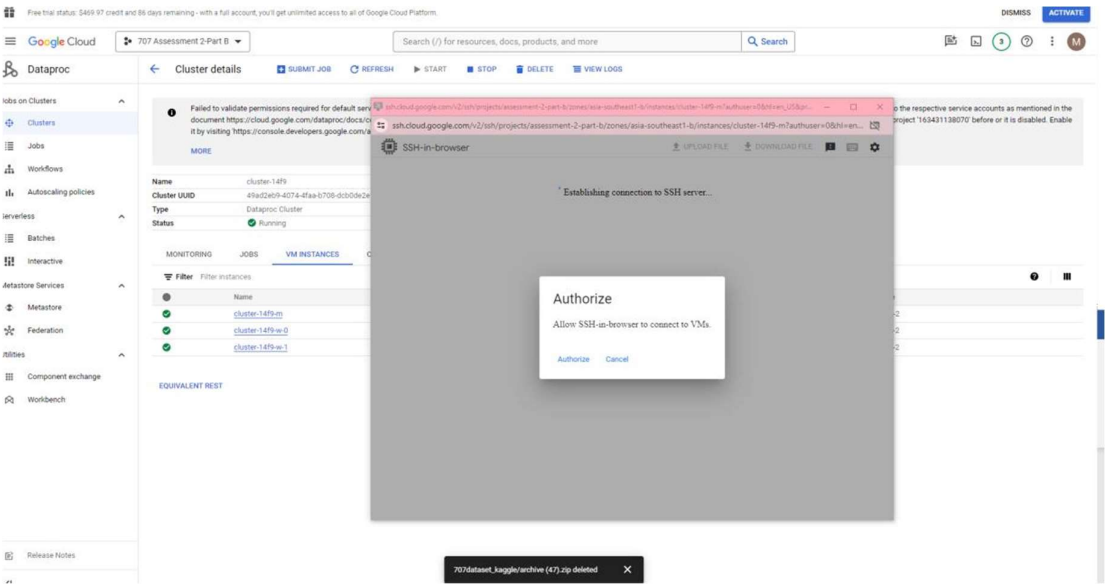
2. I created a new project in GCP, 707 Assessment 2—Part B. I also enabled Cloud Dataproc API and created a cluster on Compute Engine.

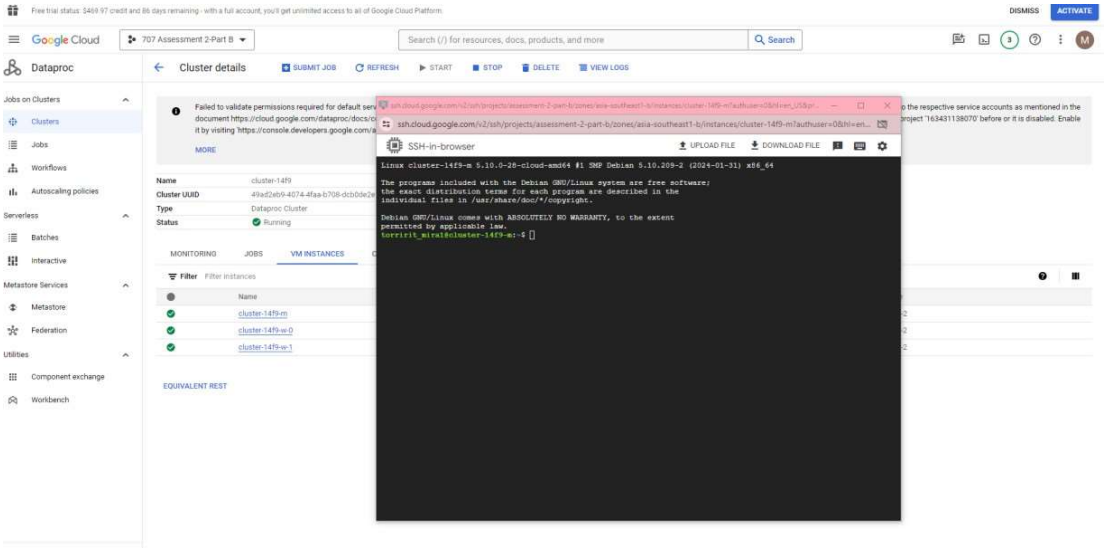


- Created a new Bucket named 707dataset_kaggle and uploaded the two datasets from Kaggle (Source GCP Storage Bucket)

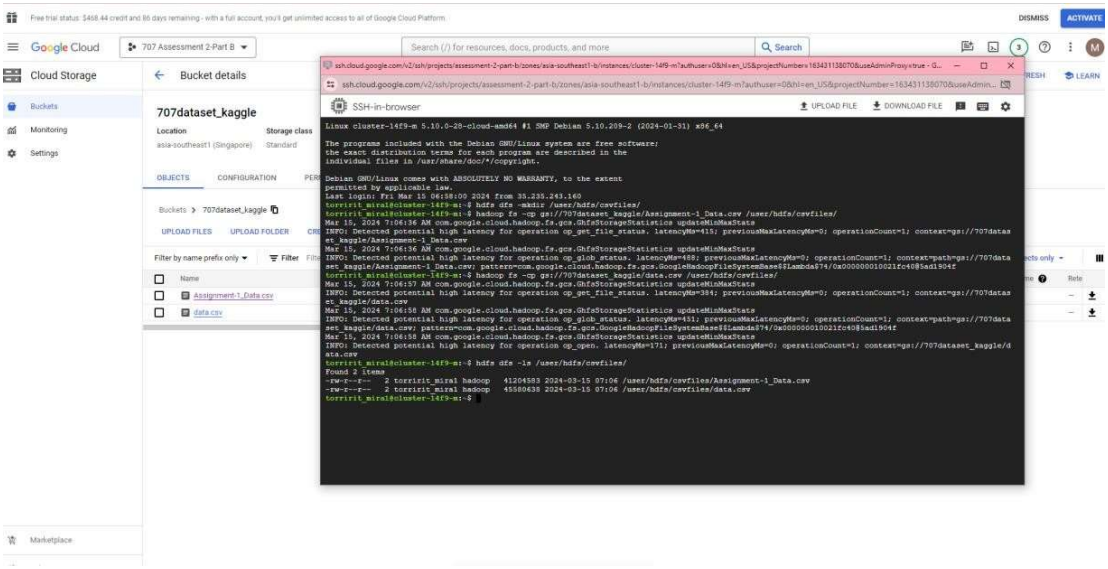


- Google Cloud console's built-in SSH interface





- Creating a destination folder (HDFS)



Task 2: Implementing Data Storage

MongoDB Atlas was chosen because it is user-friendly. The database can adjust easily to different formats as it is unstructured. The data was loaded in Python using the Pandas library and loaded into MongoDB.

```
import pandas as pd
import numpy as np
```

```
data_df = pd.read_csv('data.csv')
```

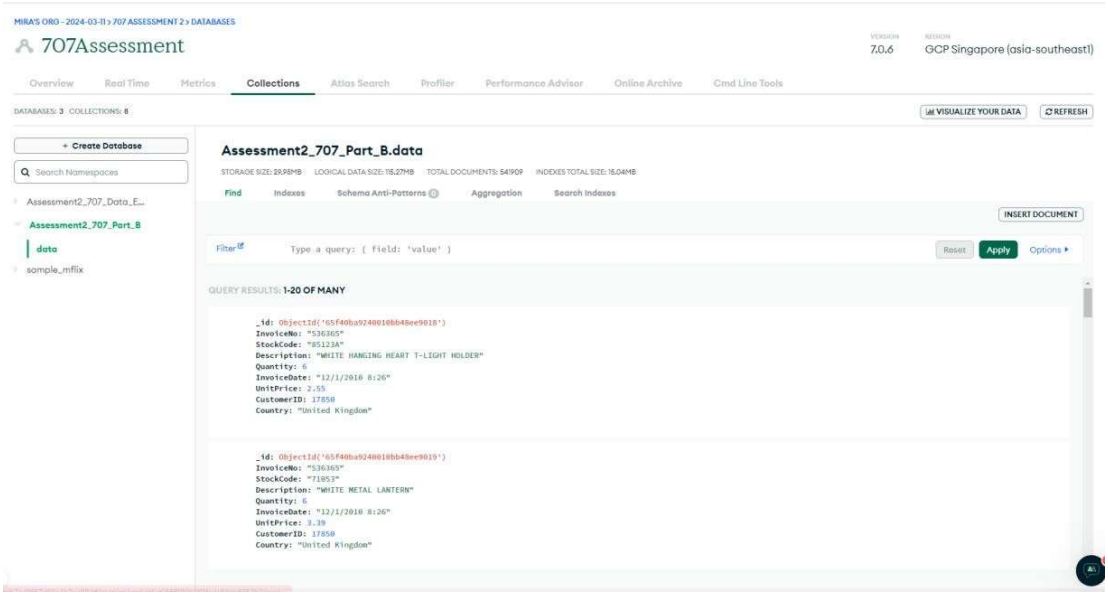
```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

def get_mongodbclient():

    uri = 'mongodb+srv://707user:ADzgnRDrEQj3FqfD@707assessment.eld0gap.mongodb.net/?retryWrites=true&w=majority&appName=707Asses

    # Create a new client and connect to the server
    client = MongoClient(uri, server_api=ServerApi('1'))

    # Send a ping to confirm a successful connection
    try:
        client.admin.command('ping')
        print("Pinged your deployment. You successfully connected to MongoDB!")
        return client
    except Exception as e:
        print(e)
        return None


# Connect to MongoDB Atlas
client = get_mongodbclient()
# Initiate an instance of the database
db = client['Assessment2_707_Part_B']
```

```
Pinged your deployment. You successfully connected to MongoDB!
```

Inserted 541,909 Documents

```
In [32]: # f) Import the dataset into a table or collection within the database.
         # In this case, import the data frame to the collection
         data_records = data_df.to_dict(orient='records')
         collection = db.data
         collection.insert_many(data_records)
```

```
Out[32]: InsertManyResult([ObjectId('65f40059d0a9120a5febfd4c'), ObjectId('65f40059d0a9120a5febfd4d'), ObjectId('65f40059d0a9120a5febfd4e'), ObjectId('65f40059d0a9120a5febfd4f'), ObjectId('65f40059d0a9120a5febfd50'), ObjectId('65f40059d0a9120a5febfd51'), ObjectId('65f40059d0a9120a5febfd52'), ObjectId('65f40059d0a9120a5febfd53'), ObjectId('65f40059d0a9120a5febfd54'), ObjectId('65f40059d0a9120a5febfd55'), ObjectId('65f40059d0a9120a5febfd56'), ObjectId('65f40059d0a9120a5febfd57'), ObjectId('65f40059d0a9120a5febfd58'), ObjectId('65f40059d0a9120a5febfd59'), ObjectId('65f40059d0a9120a5febfd5a'), ObjectId('65f40059d0a9120a5febfd5b'), ObjectId('65f40059d0a9120a5febfd5c'), ObjectId('65f40059d0a9120a5febfd5d'), ObjectId('65f40059d0a9120a5febfd5e'), ObjectId('65f40059d0a9120a5febfd5f'), ObjectId('65f40059d0a9120a5febfd60'), ObjectId('65f40059d0a9120a5febfd61'), ObjectId('65f40059d0a9120a5febfd62'), ObjectId('65f40059d0a9120a5febfd63'), ObjectId('65f40059d0a9120a5febfd64'), ObjectId('65f40059d0a9120a5febfd65'), ObjectId('65f40059d0a9120a5febfd66'), ObjectId('65f40059d0a9120a5febfd67'), ObjectId('65f40059d0a9120a5febfd68'), ObjectId('65f40059d0a9120a5febfd69'), ObjectId('65f40059d0a9120a5febfd6a'), ObjectId('65f40059d0a9120a5febfd6b'), ObjectId('65f40059d0a9120a5febfd6c'), ObjectId('65f40059d0a9120a5febfd6d'), ObjectId('65f40059d0a9120a5febfd6e'), ObjectId('65f40059d0a9120a5febfd6f'), ObjectId('65f40059d0a9120a5febfd70'), ObjectId('65f40059d0a9120a5febfd71'), ObjectId('65f40059d0a9120a5febfd72'), ObjectId('65f40059d0a9120a5febfd73'), ObjectId('65f40059d0a9120a5febfd74'), ObjectId('65f40059d0a9120a5febfd75'), ObjectId('65f40059d0a9120a5febfd76'), ObjectId('65f40059d0a9120a5febfd77'), ObjectId('65f40059d0a9120a5febfd78'), ObjectId('65f40059d0a9120a5febfd79'), ObjectId('65f40059d0a9120a5febfd7a'), ObjectId('65f40059d0a9120a5febfd7b'), ObjectId('65f40059d0a9120a5febfd7c'), ObjectId('65f40059d0a9120a5febfd7d'), ObjectId('65f40059d0a9120a5febfd7e'), ObjectId('65f40059d0a9120a5febfd7f'), ObjectId('65f40059d0a9120a5febfd80'), ObjectId('65f40059d0a9120a5febfd81'), ObjectId('65f40059d0a9120a5febfd82'), ObjectId('65f40059d0a9120a5febfd83'), ObjectId('65f40059d0a9120a5febfd84'), ObjectId('65f40059d0a9120a5febfd85'), ObjectId('65f40059d0a9120a5febfd86'), ObjectId('65f40059d0a9120a5febfd87'), ObjectId('65f40059d0a9120a5febfd88'), ObjectId('65f40059d0a9120a5febfd89'), O
```



Performed Query to get the 541, 909 Documents.

```
In [33]: # g) Retrieve and display records or documents from the table or collection.
         # using collection.find to get all the records from adidas_collection
         collection = db.data
         data = collection.find()

         new_data_df = pd.DataFrame(list(data))
         new_data_df
```

Out[33]:

	_id	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	65f40059d0a9120a5febfd4c	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	65f40059d0a9120a5febfd4d	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	65f40059d0a9120a5febfd4e	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	65f40059d0a9120a5febfd4f	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	65f40059d0a9120a5febfd50	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...	...
541904	65f4005bd0a9120a5ff4421c	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	France
541905	65f4005bd0a9120a5ff4421d	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	France
541906	65f4005bd0a9120a5ff4421e	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	France
541907	65f4005bd0a9120a5ff4421f	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	France
541908	65f4005bd0a9120a5ff44220	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	France

541909 rows × 9 columns

Performed sorting for the 541,909 Documents

```
In [34]: # h) Sort the records or documents based on a given condition.
         # Sort by Invoice Date
         sorted_new_data_df = new_data_df.sort_values(by='InvoiceDate', ascending=False)
         sorted_new_data_df
```

Out[34]:

	_id	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
332571	65f4005ad0a9120a5ff11067	566079	22923	FRIDGE MAGNETS LES ENFANTS ASSORTED	24	9/9/2011 9:52	0.85	17593.0	United Kingdom
332544	65f4005ad0a9120a5ff1104c	566079	23403	LETTER HOLDER HOME SWEET HOME	4	9/9/2011 9:52	3.75	17593.0	United Kingdom
332551	65f4005ad0a9120a5ff11053	566079	22396	MAGNETS PACK OF 4 RETRO PHOTO	24	9/9/2011 9:52	0.39	17593.0	United Kingdom
332550	65f4005ad0a9120a5ff11052	566079	22400	MAGNETS PACK OF 4 HOME SWEET HOME	24	9/9/2011 9:52	0.39	17593.0	United Kingdom
332549	65f4005ad0a9120a5ff11051	566079	20838	FRENCH LATTICE CUSHION COVER	12	9/9/2011 9:52	0.85	17593.0	United Kingdom
...	...	...	...	...	...	...	...	...	...
50831	65f40059d0a9120a5fecc3db	540561	22743	MAKE YOUR OWN FLOWERPOWER CARD KIT	6	1/10/2011 10:32	2.95	13004.0	United Kingdom
50809	65f40059d0a9120a5fecc3c5	540561	22343	PARTY PIZZA DISH RED RETROSPOT	24	1/10/2011 10:32	0.21	13004.0	United Kingdom
50808	65f40059d0a9120a5fecc3c4	540560	21589	NaN	-14	1/10/2011 10:08	0.00	NaN	United Kingdom
50807	65f40059d0a9120a5fecc3c3	C540559	21888	BINGO SET	-4	1/10/2011 10:07	3.75	NaN	EIRE
50806	65f40059d0a9120a5fecc3c2	540558	21258	?	-29	1/10/2011 10:04	0.00	NaN	United Kingdom

541909 rows × 9 columns

Counted the number of documents using the Python Application

```
In [35]: # i) Count the number of records or documents present in the table or collection.(
         collection = db.data
         document_count = collection.count_documents({})
         print(f'Total Documents: {document_count}')
```

```
Total Documents in adidas_collection: 541909
```

Performed Grouping

```
In [36]: # j) Perform grouping operations on records or documents within the table or collection.
collection = db.data
grouping = [{
    "$group": {
        "_id" : "$Country",
        "count" : {"$sum" : 1}
    }
}]
result = list(collection.aggregate(grouping))
print(result)
```

```
[{'_id': 'Australia', 'count': 1259}, {'_id': 'EIRE', 'count': 8196}, {'_id': 'Greece', 'count': 146}, {'_id': 'USA', 'count':
291}, {'_id': 'Iceland', 'count': 182}, {'_id': 'Canada', 'count': 151}, {'_id': 'Israel', 'count': 297}, {'_id': 'Unspecifie
d', 'count': 446}, {'_id': 'RSA', 'count': 58}, {'_id': 'Malta', 'count': 127}, {'_id': 'France', 'count': 8557}, {'_id': 'Swed
en', 'count': 462}, {'_id': 'Portugal', 'count': 1519}, {'_id': 'Spain', 'count': 2533}, {'_id': 'Lithuania', 'count': 35}, {'_
id': 'Netherlands', 'count': 2371}, {'_id': 'Denmark', 'count': 389}, {'_id': 'Belgium', 'count': 2069}, {'_id': 'Bahrain', 'co
unt': 19}, {'_id': 'Saudi Arabia', 'count': 10}, {'_id': 'Norway', 'count': 1086}, {'_id': 'Poland', 'count': 341}, {'_id': 'Ho
ng Kong', 'count': 288}, {'_id': 'United Kingdom', 'count': 495478}, {'_id': 'Channel Islands', 'count': 758}, {'_id': 'Italy',
'count': 803}, {'_id': 'Czech Republic', 'count': 30}, {'_id': 'Switzerland', 'count': 2002}, {'_id': 'European Community', 'co
unt': 61}, {'_id': 'Singapore', 'count': 229}, {'_id': 'Cyprus', 'count': 622}, {'_id': 'Finland', 'count': 695}, {'_id': 'Japa
n', 'count': 358}, {'_id': 'Lebanon', 'count': 45}, {'_id': 'Germany', 'count': 9495}, {'_id': 'United Arab Emirates', 'count':
68}, {'_id': 'Brazil', 'count': 32}, {'_id': 'Austria', 'count': 401}]
```

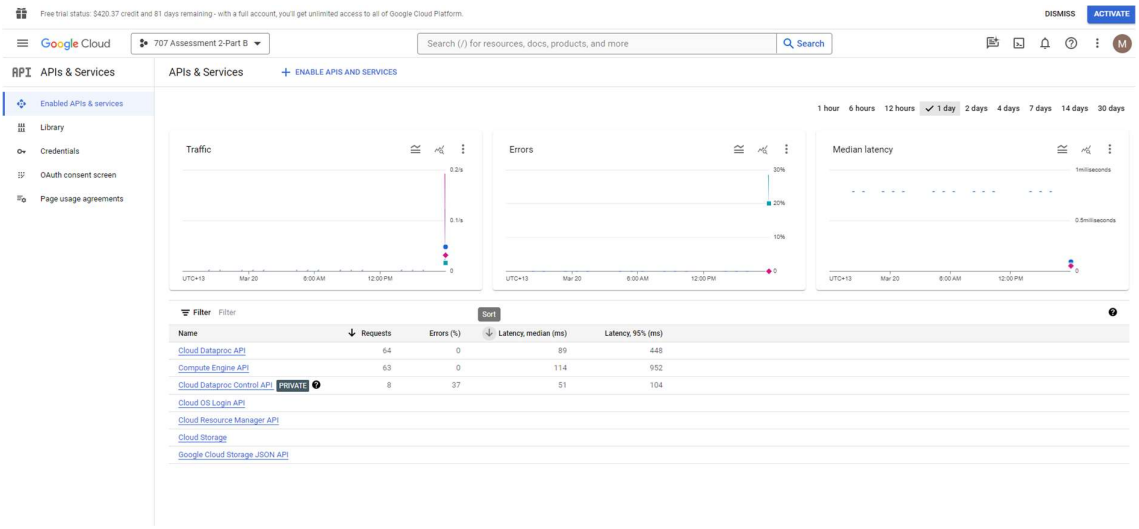
Task 3: Building a real-time data clustering system using Apache Spark

In this activity, I still used the GCP to utilize my free credits. Also, the following are the reasons for considering Data Proc on GCP in managing real-time data, from Facebook Graph API.

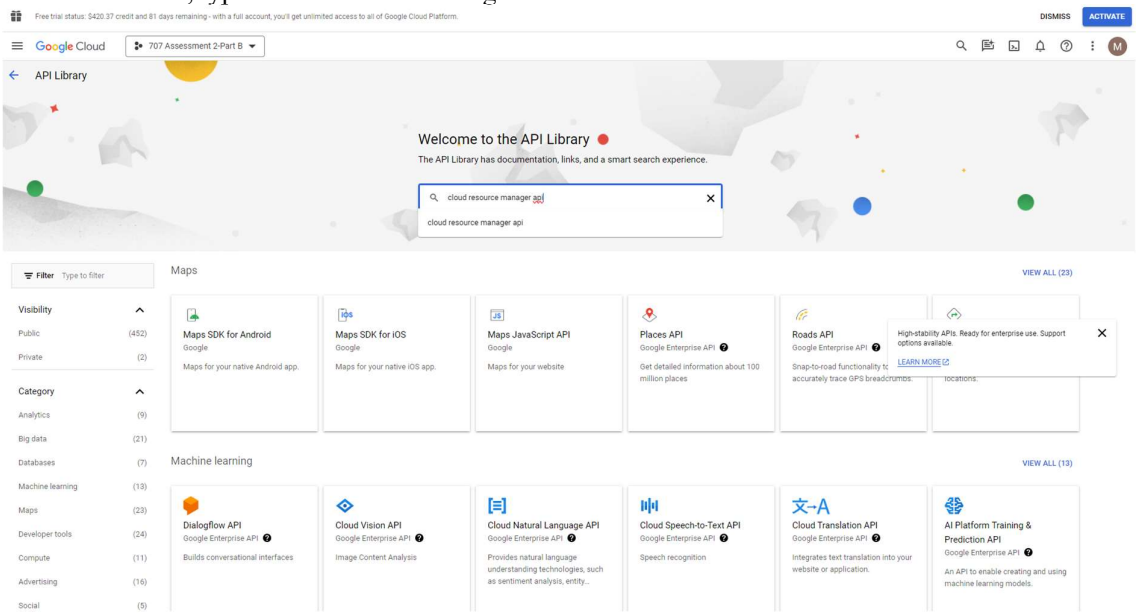
1. Scalability: Can smoothly adjust its scale up or down based on the amount of data being processed. Dealing with fluctuating volumes of data from services, like the Facebook Graph API requires processing resources to efficiently handle the workload.
2. Managed Service: This is fully managed by Google Cloud taking care of tasks like infrastructure management such as provisioning, configuration, and monitoring of computer instances. This allows you to concentrate on developing your streaming application without being burdened by responsibilities.
3. Integration with GCP Services: Seamlessly integrates with Google Cloud services like Big Query for data warehousing Pub/Sub for messaging and Dataflow for stream processing. This integration facilitates the creation of end-to-end data pipelines—from ingestion to analysis—in a manner.
4. Support for Spark Streaming: When you use Apache Spark for processing streaming data DataProc offers support for Spark Streaming that's ideal, for managing real-time data streams. With familiar APIs used in batch processing, Spark Streaming simplifies the development of streaming applications.
5. Resource Optimization: Gives you the ability to enhance your resource utilization and manage expenses by defining the type and quantity of machines (VMs) required for your processing activities. This adaptability empowers you to strike a balance between performance and cost efficiency according to your needs. In essence, leveraging DataProc on GCP for streaming data from the Facebook Graph API offers an expandable solution with operational burden enabling you to concentrate on extracting valuable insights, from your streaming data.

Steps on building the real-time data clustering (with an FB account)

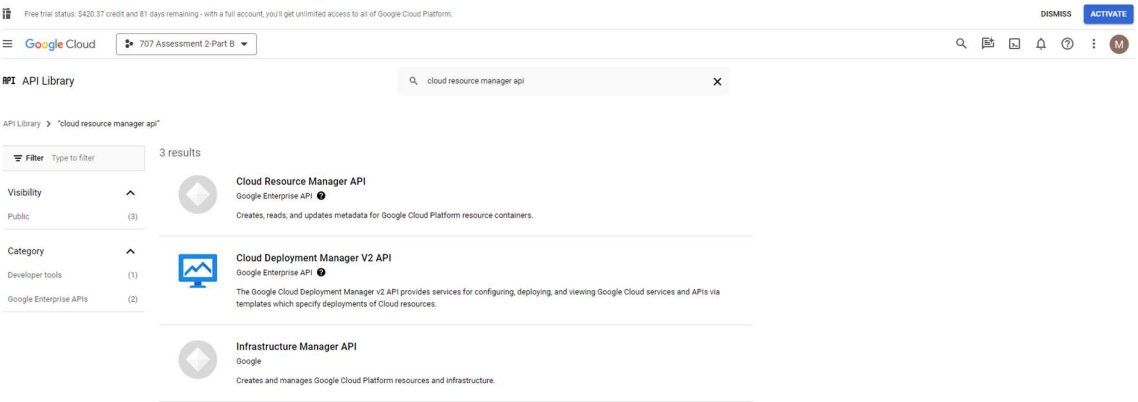
1. Go to Data proc clusters, select the cluster, or create a new cluster (essential details to add: Jupyter). To enable the Jupyter interface, go to the navigation menu, API & services, and click + enable APIS and Services.



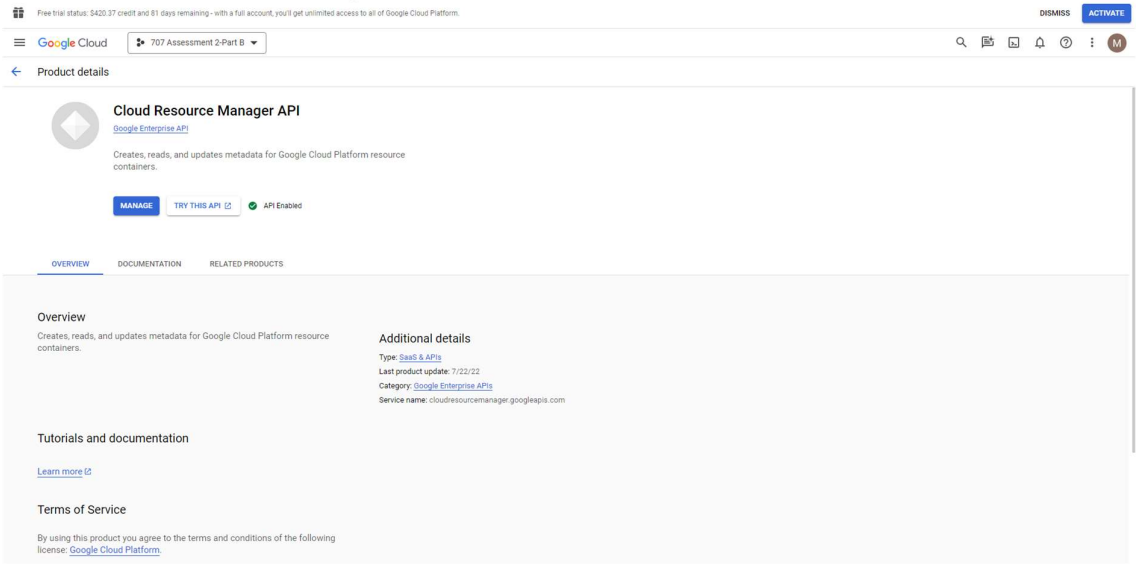
In the search bar, type cloud resource manager API



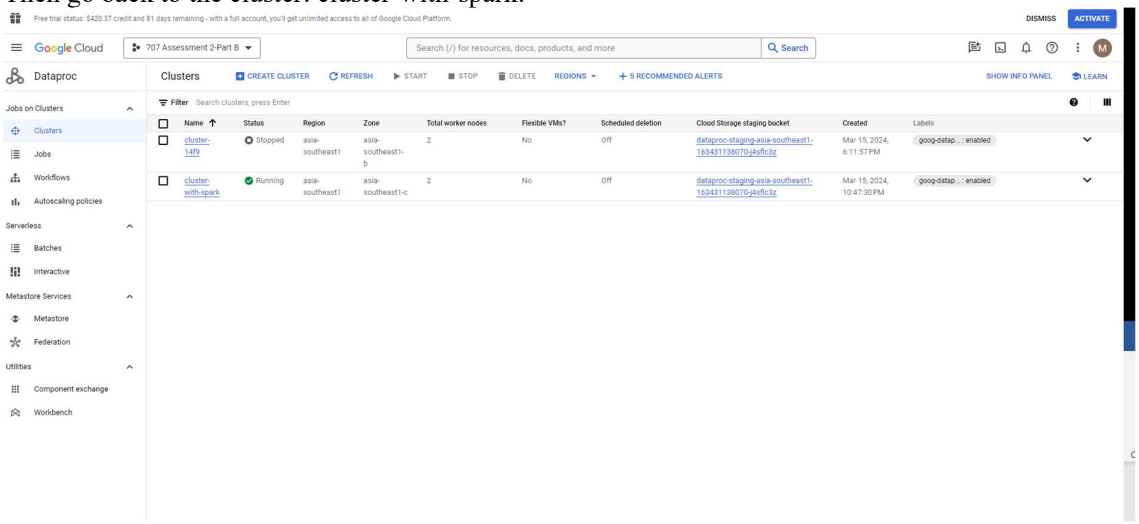
Click Cloud Resource Manager API to Enable



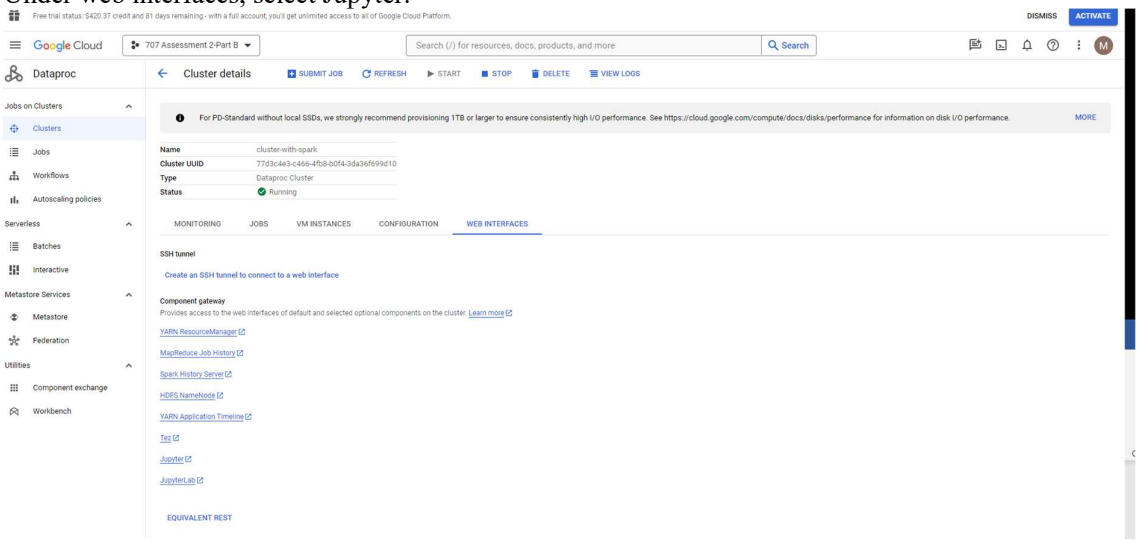
Enabled Cloud Resource Manager API



Then go back to the cluster: cluster-with-spark.

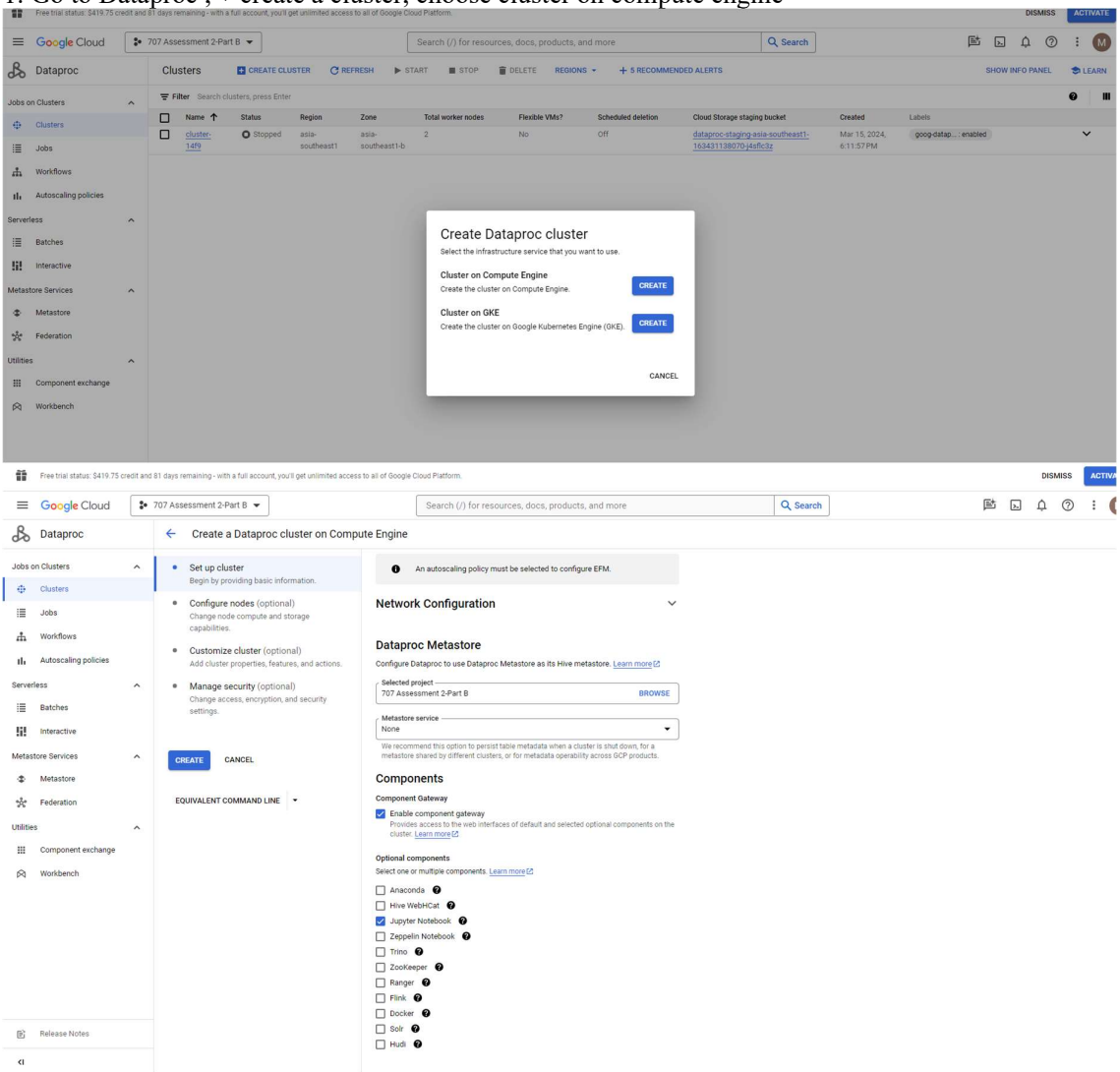


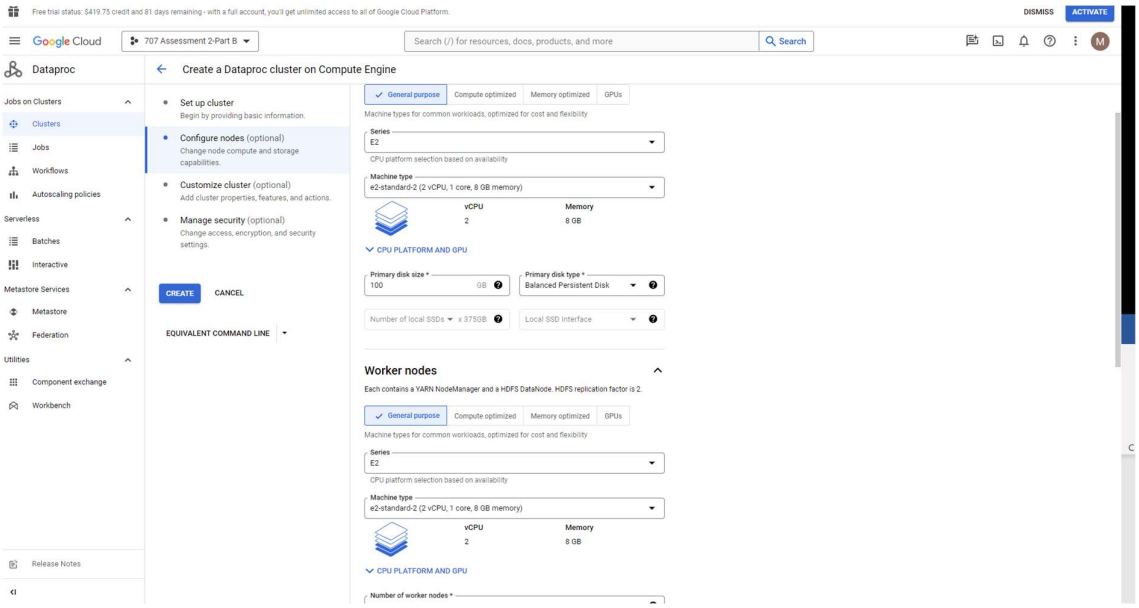
Under web interfaces, select Jupyter.

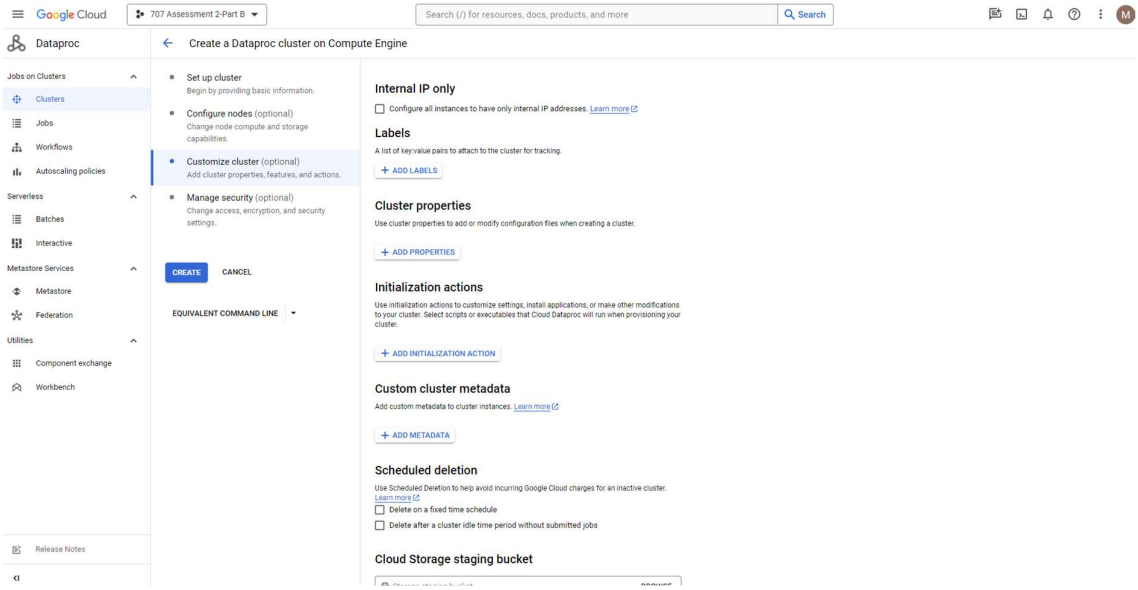


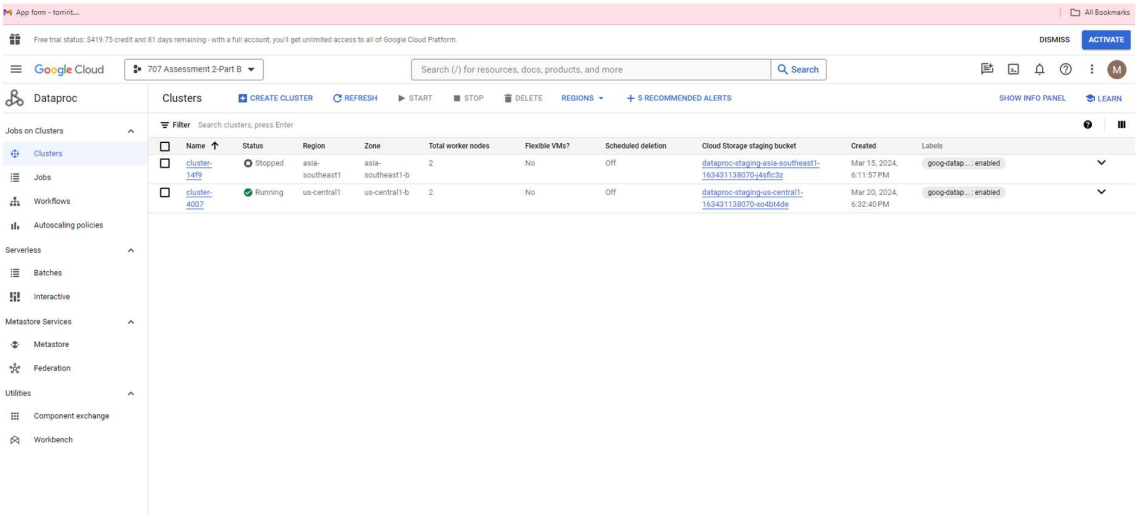
Creating a cluster

1. Go to Dataproc , + create a cluster, choose cluster on compute engine

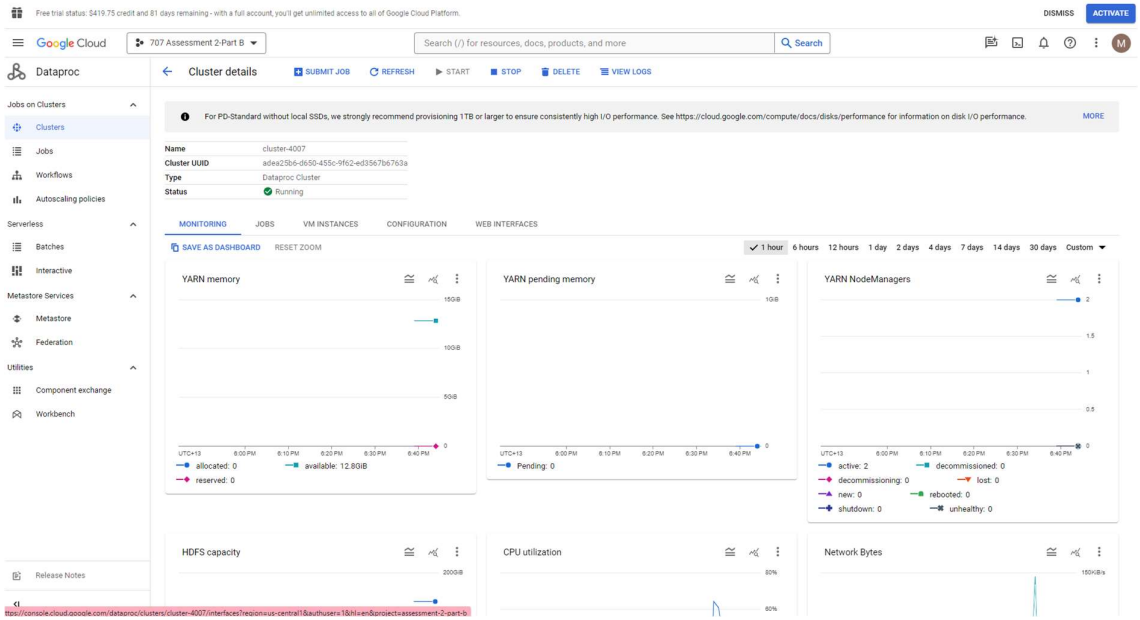



Free trial status: $419.75 credit and 81 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.
DISMISS
ACTIVATE
Google Cloud
707 Assessment 2-Part B
Search (/) for resources, docs, products, and more
Search
Dataproc
Create a Dataproc cluster on Compute Engine
Jobs on Clusters
Clusters
Jobs
Workflows
Autoscaling policies
Serverless
Batches
Interactive
Metastore Services
Metastore
Federation
Utilities
Component exchange
Workbench
Release Notes
Set up cluster
Begin by providing basic information.
Configure nodes (optional)
Change node compute and storage capabilities.
Customize cluster (optional)
Add cluster properties, features, and actions.
Manage security (optional)
Change access, encryption, and security settings.
CREATE
CANCEL
EQUIVALENT COMMAND LINE
General purpose
Compute optimized
Memory optimized
GPUs
Machine types for common workloads, optimized for cost and flexibility
Series
E2
CPU platform selection based on availability
Machine type
e2-standard-2 (2 vCPU, 1 core, 8 GB memory)
vCPU
2
Memory
8 GB
CPU PLATFORM AND GPU
Primary disk size *
100
GB
Primary disk type *
Balanced Persistent Disk
Number of local SSDs
x 375GB
Local SSD interface
Worker nodes
Each contains a YARN NodeManager and a HDFS DataNode. HDFS replication factor is 2.
Number of worker nodes *

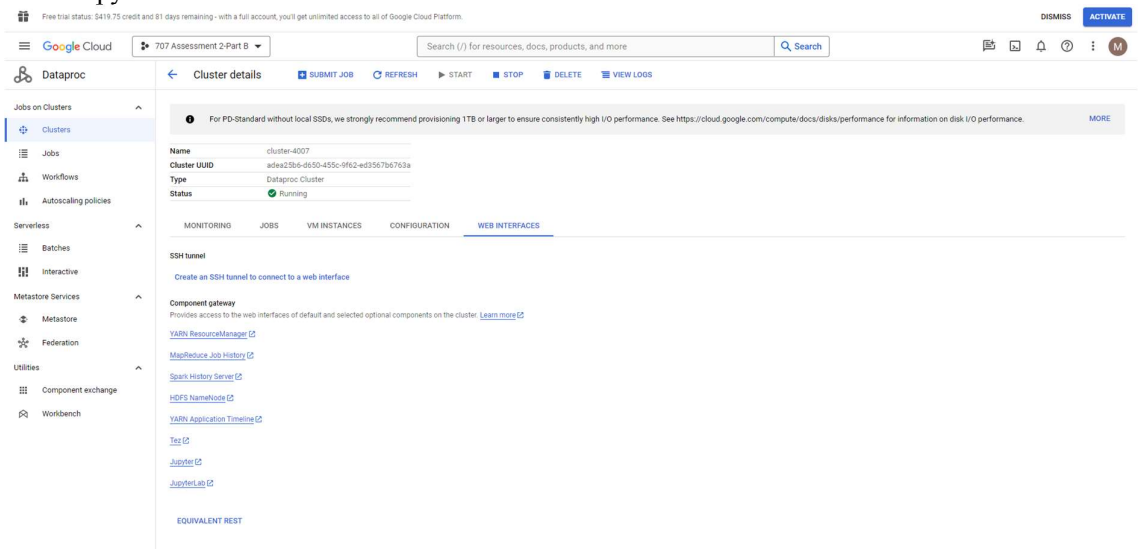

Google Cloud
707 Assessment 2-Part B
Search (/) for resources, docs, products, and more
Search
Dataproc
Create a Dataproc cluster on Compute Engine
Set up cluster
Begin by providing basic information.
Configure nodes (optional)
Change node compute and storage capabilities.
Customize cluster (optional)
Add cluster properties, features, and actions.
Manage security (optional)
Change access, encryption, and security settings.
CREATE
CANCEL
EQUIVALENT COMMAND LINE
Internal IP only
Configure all instances to have only internal IP addresses. Learn more
Labels
A list of key:value pairs to attach to the cluster for tracking.
+ ADD LABELS
Cluster properties
Use cluster properties to add or modify configuration files when creating a cluster.
+ ADD PROPERTIES
Initialization actions
Use initialization actions to customize settings, install applications, or make other modifications to your cluster. Select scripts or executables that Cloud Dataproc will run when provisioning your cluster.
+ ADD INITIALIZATION ACTION
Custom cluster metadata
Add custom metadata to cluster instances. Learn more
+ ADD METADATA
Scheduled deletion
Use Scheduled Deletion to help avoid incurring Google Cloud charges for an inactive cluster. Learn more
Delete on a fixed time schedule
Delete after a cluster idle time period without submitted jobs
Cloud Storage staging bucket

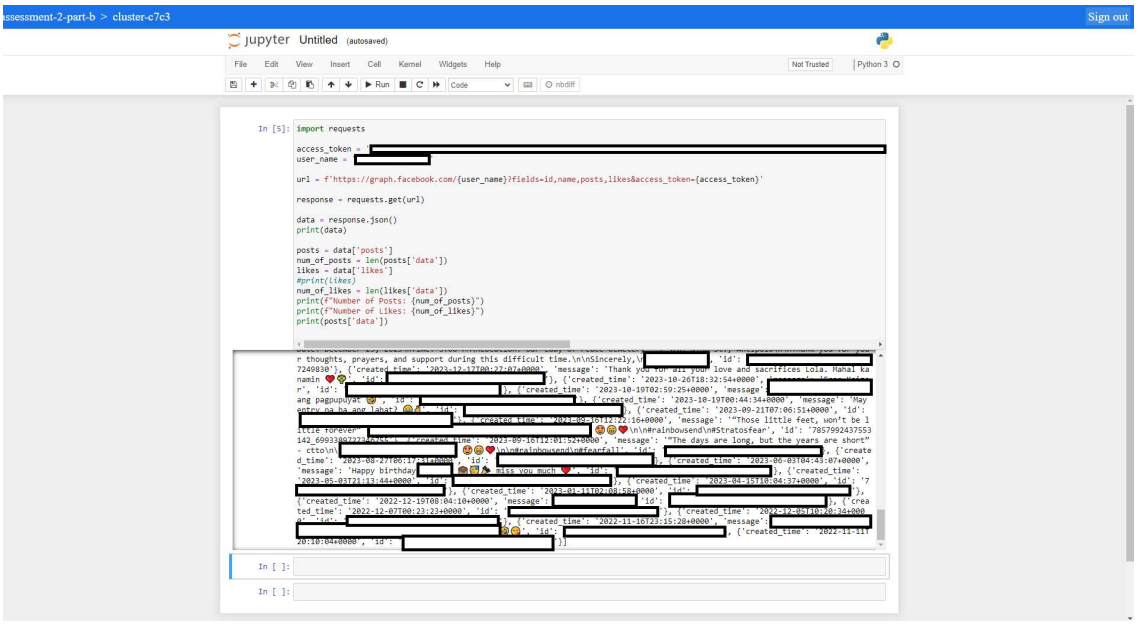

App form - torririt...
All Bookmarks
Free trial status: $419.75 credit and 81 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.
DISMISS
ACTIVATE
Google Cloud
707 Assessment 2-Part B
Search (/) for resources, docs, products, and more
Search
Dataproc
Clusters
CREATE CLUSTER
REFRESH
START
STOP
DELETE
REGIONS
5 RECOMMENDED ALERTS
SHOW INFO PANEL
LEARN
Filter
Search clusters, press Enter
Name
Status
Region
Zone
Total worker nodes
Flexible VMs?
Scheduled deletion
Cloud Storage staging bucket
Created
Labels
cluster-14f9
Stopped
asia-southeast1
asia-southeast1-b
2
No
Off
dataproc-staging-asia-southeast1-163431138070-j4efic3z
Mar 15, 2024, 6:11:57 PM
goog-datap... enabled
cluster-4007
Running
us-central1
us-central1-b
2
No
Off
dataproc-staging-us-central1-163431138070-xo4bt4de
Mar 20, 2024, 6:32:40 PM
goog-datap... enabled

Click the web interface.



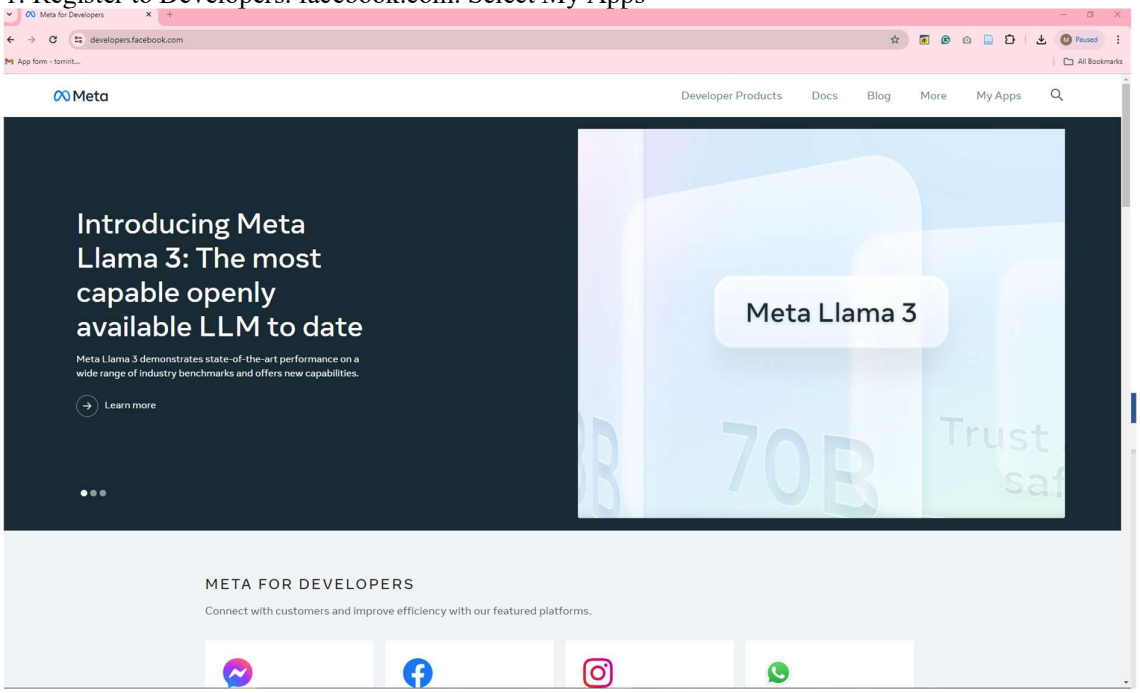
Click Jupyter



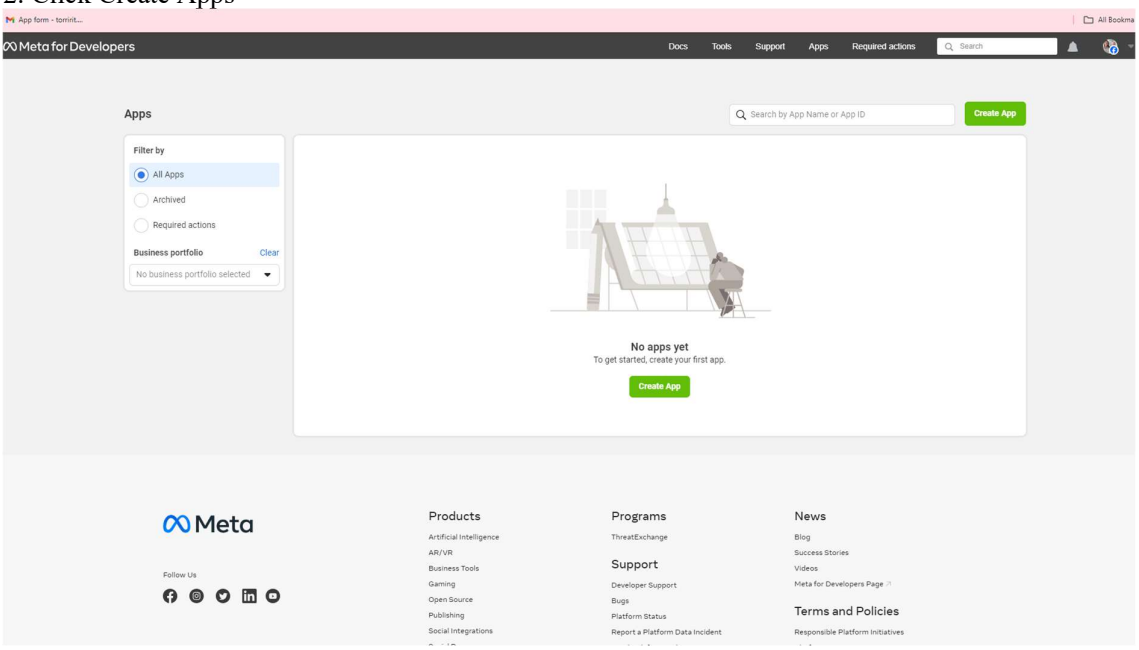


- Generating Access Token from Facebook – this allows you to connect to the Facebook account.

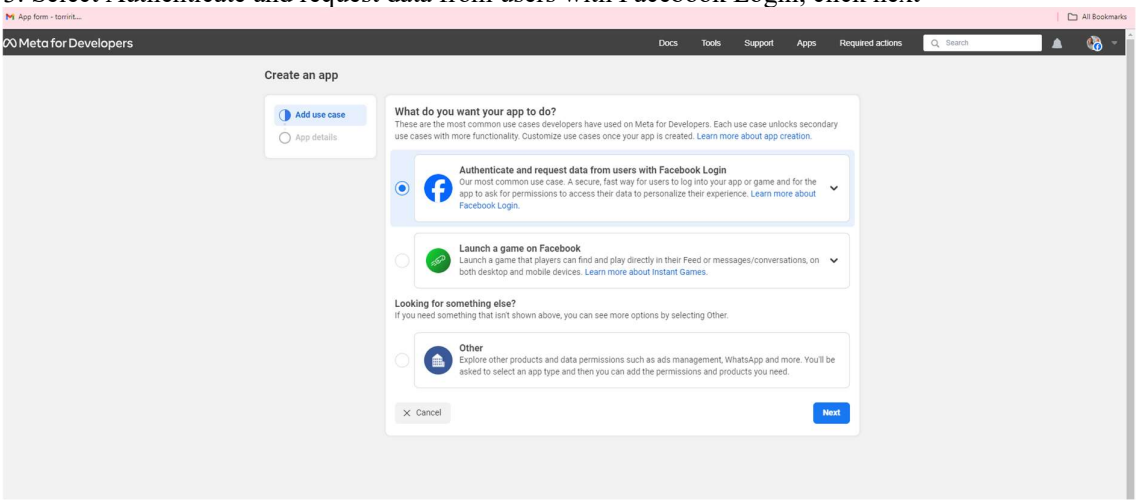
1. Register to Developers. facebook.com. Select My Apps



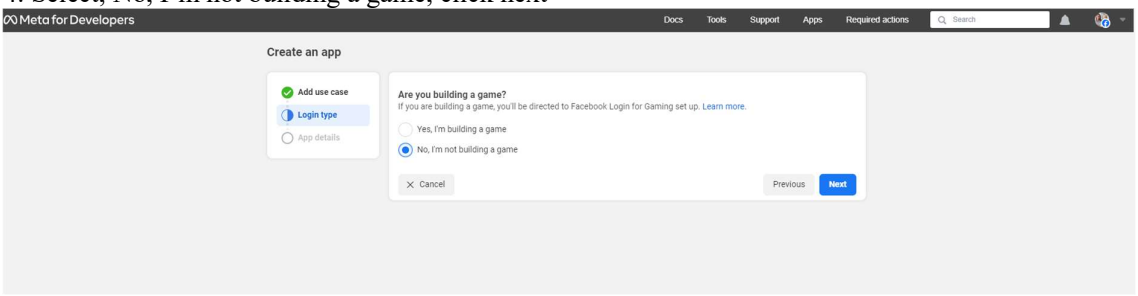
2. Click Create Apps



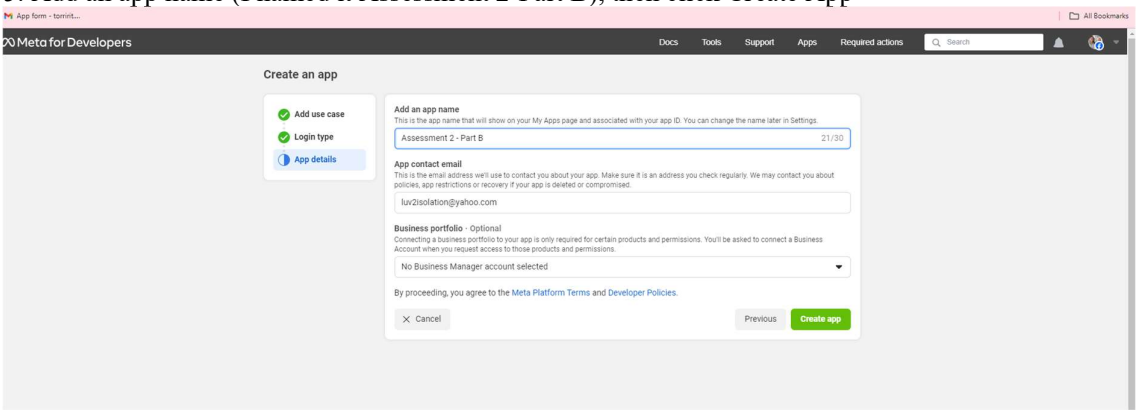
3. Select Authenticate and request data from users with Facebook Login, click next



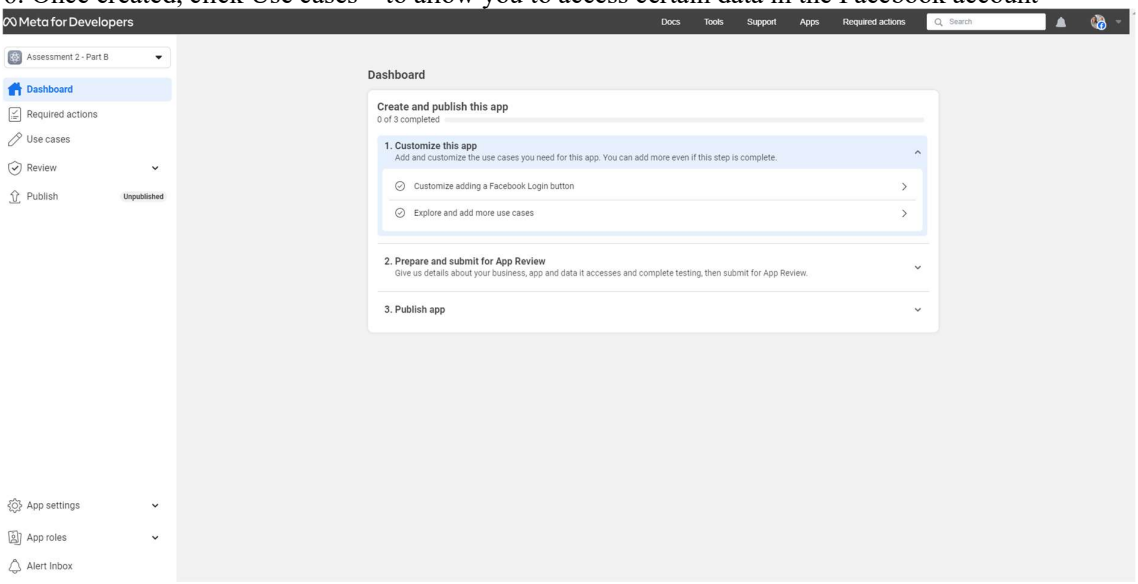
4. Select, No, I'm not building a game, click next



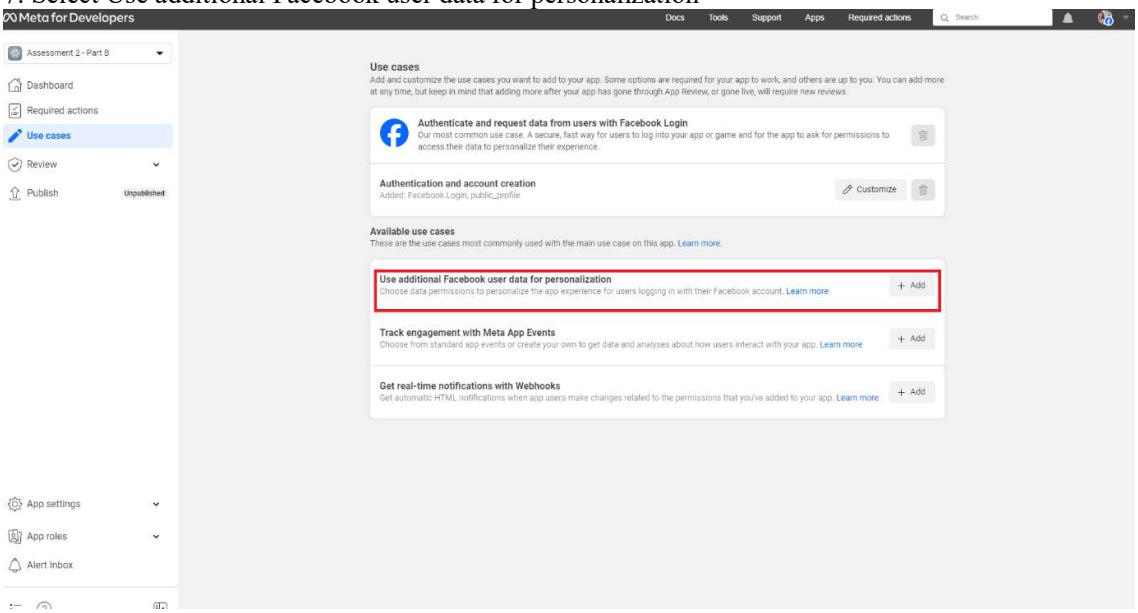
5. Add an app name (I named it Assessment 2-Part B), then click Create App



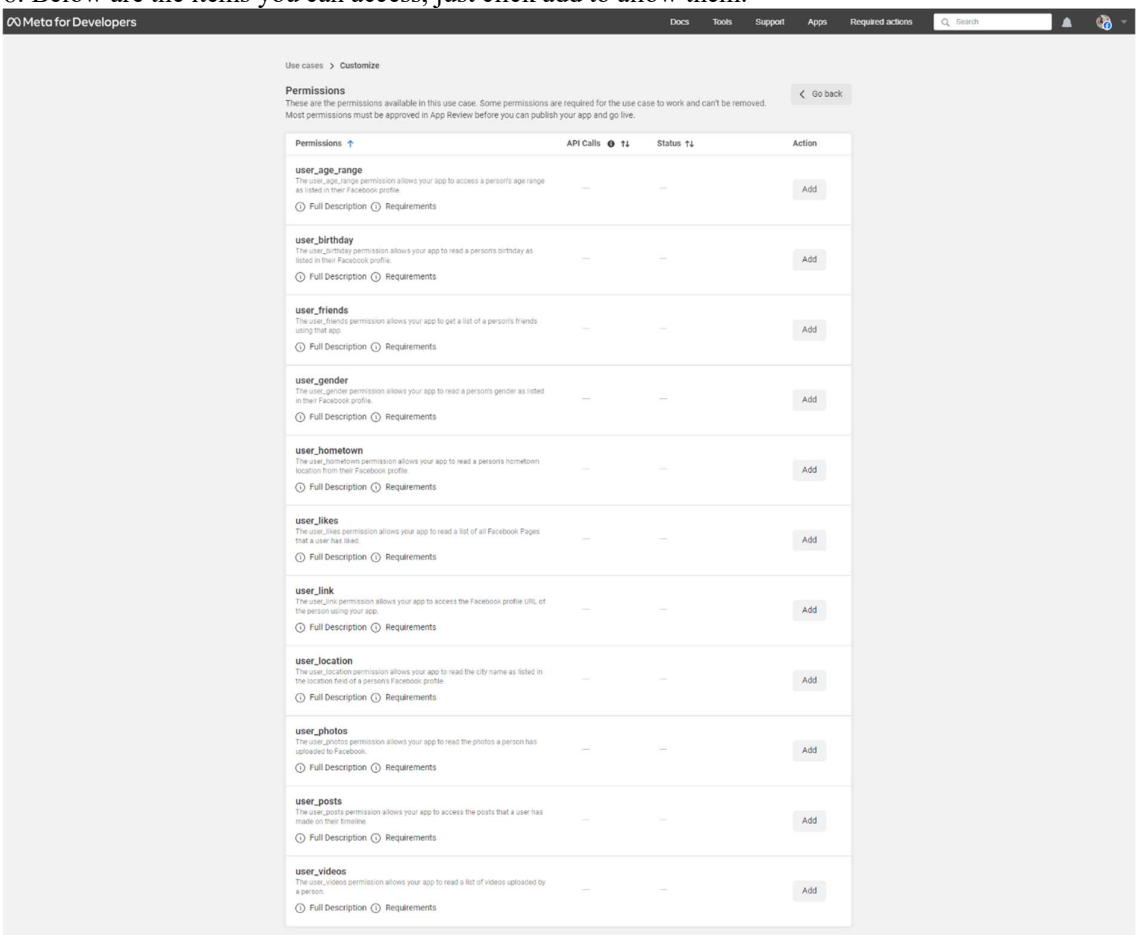
6. Once created, click Use cases – to allow you to access certain data in the Facebook account



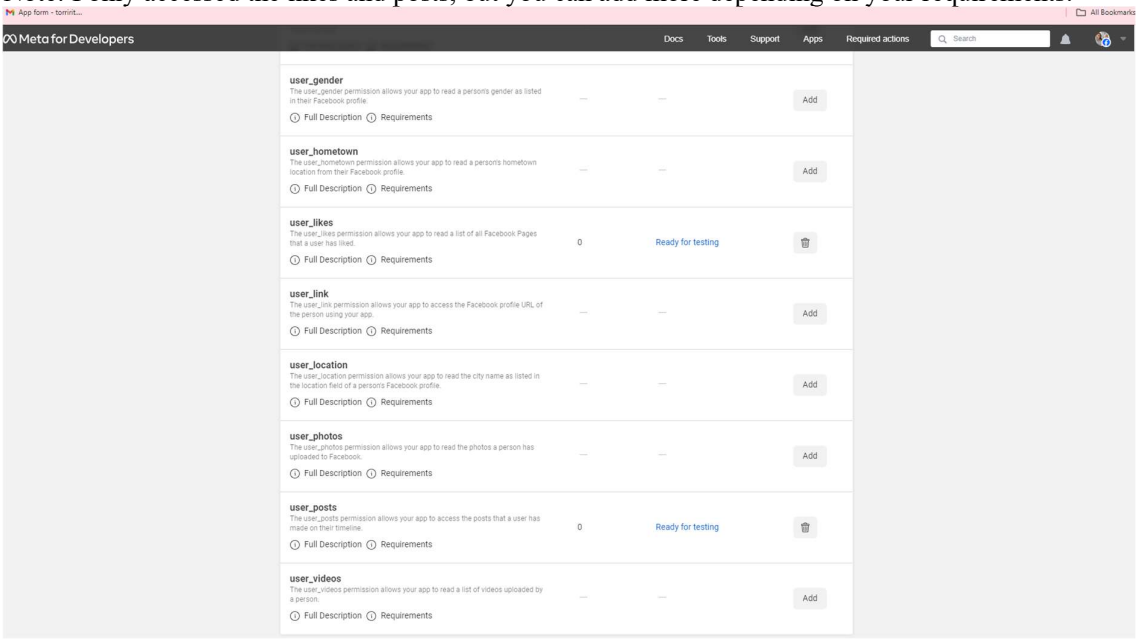
7. Select Use additional Facebook user data for personalization



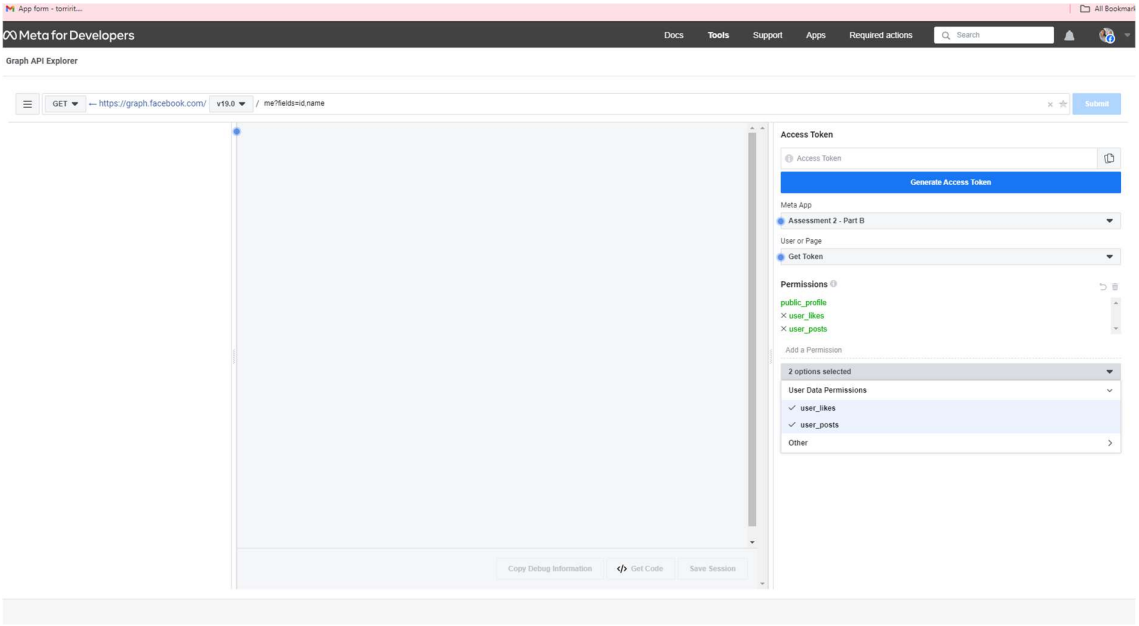
8. Below are the items you can access, just click add to allow them.



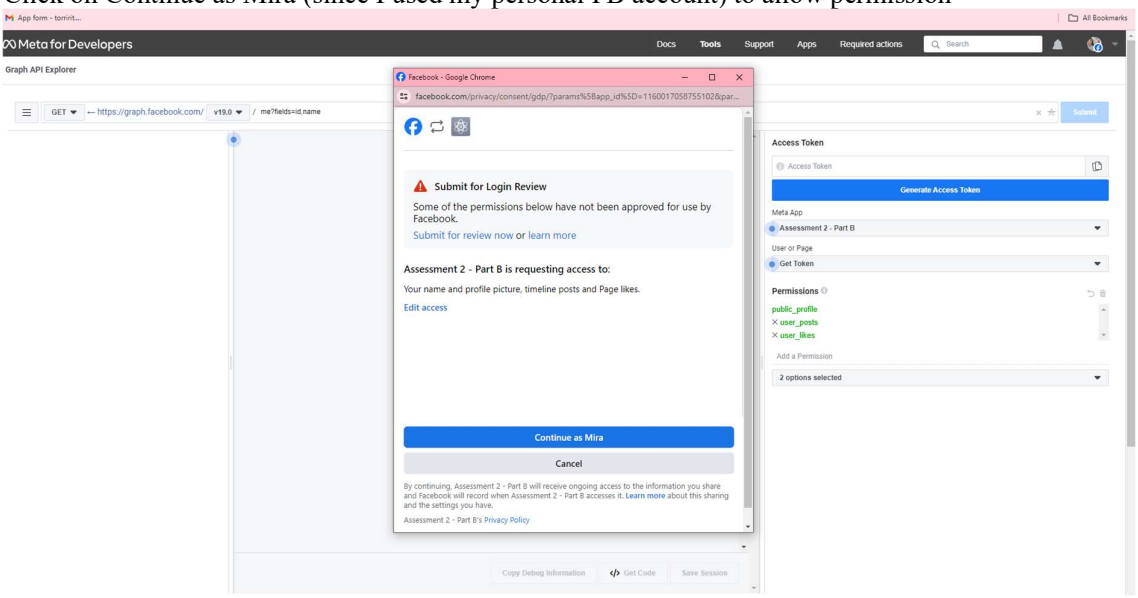
Note: I only accessed the likes and posts, but you can add more depending on your requirements.



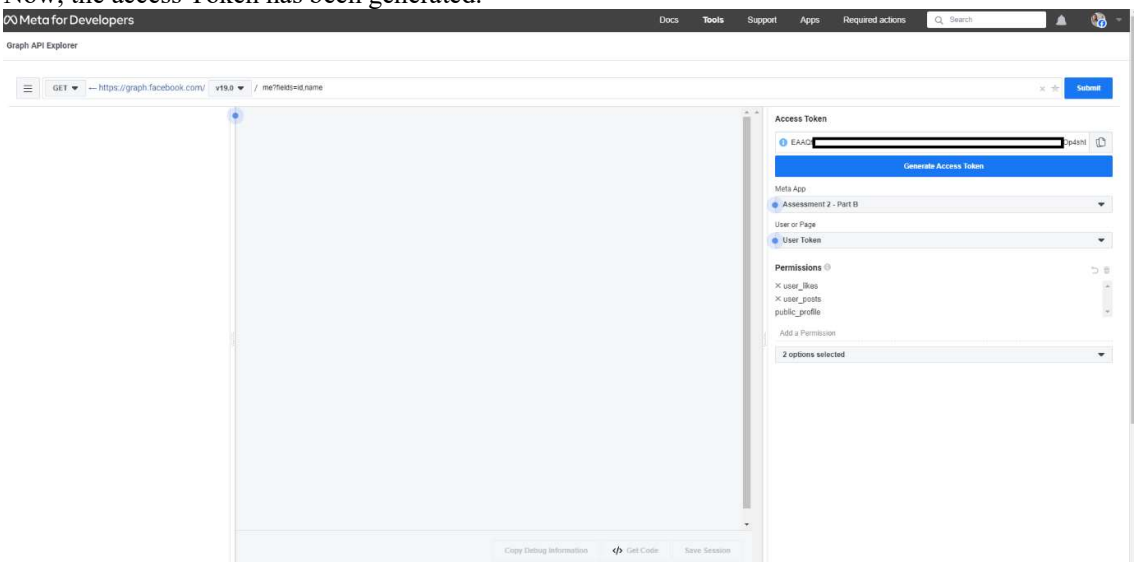
In the top menu bar, hover Tools and select Graph API Explorer. Choose the Meta App created, the action you are going to do (Get Token) and Add permissions, to access the two permissions you added from Use cases. Then click Generate Access Token.



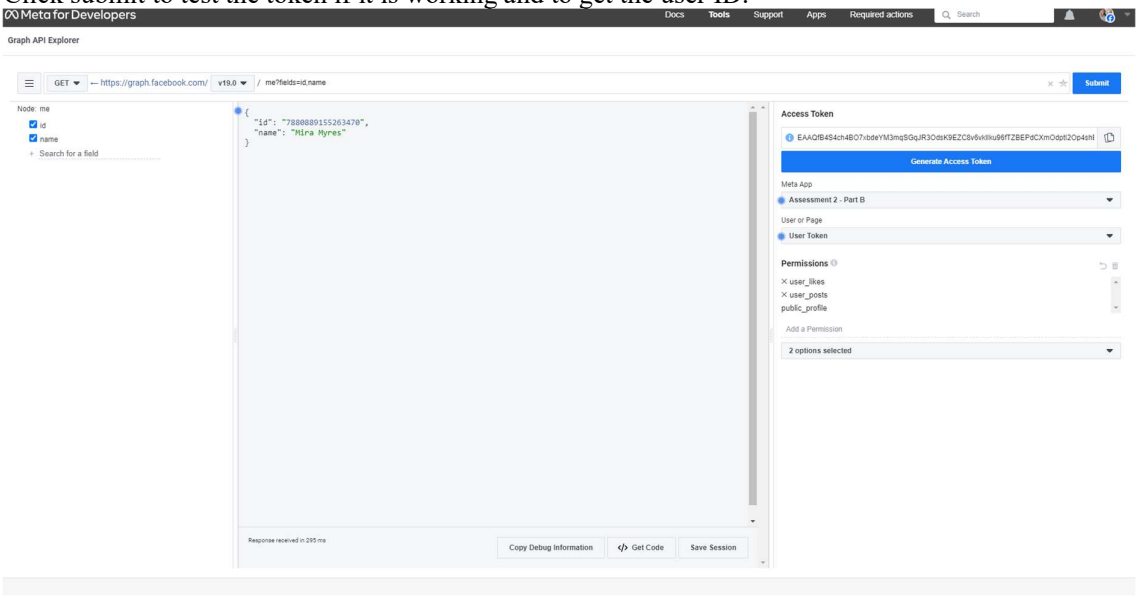
Click on Continue as Mira (since I used my personal FB account) to allow permission



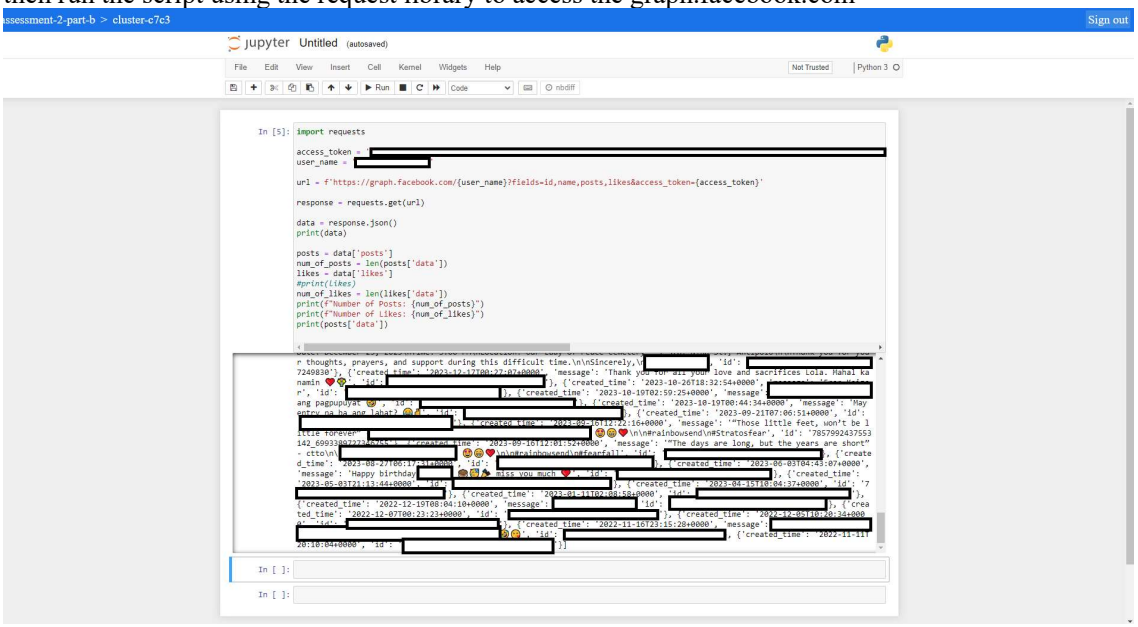
Now, the access Token has been generated.



Click submit to test the token if it is working and to get the user ID.



Then Go to GCP, Dataproc, Cluster, and Jupyter and copy the access token and ID to the Python script, then run the script using the request library to access the graph.facebook.com



You can also access the number of likes and posts.

```
response = requests.get(url)

data = response.json()
#print(data)

posts = data['posts']
num_of_posts = len(posts['data'])
likes = data['likes']
#print(likes)
num_of_likes = len(likes['data'])
print(f"Number of Posts: {num_of_posts}") #TODO: numbers are not yet in total as we need to collect per paging request
print(f"Number of Likes: {num_of_likes}") #TODO: numbers are not yet in total as we need to collect per paging request
#print(posts['data'])
```

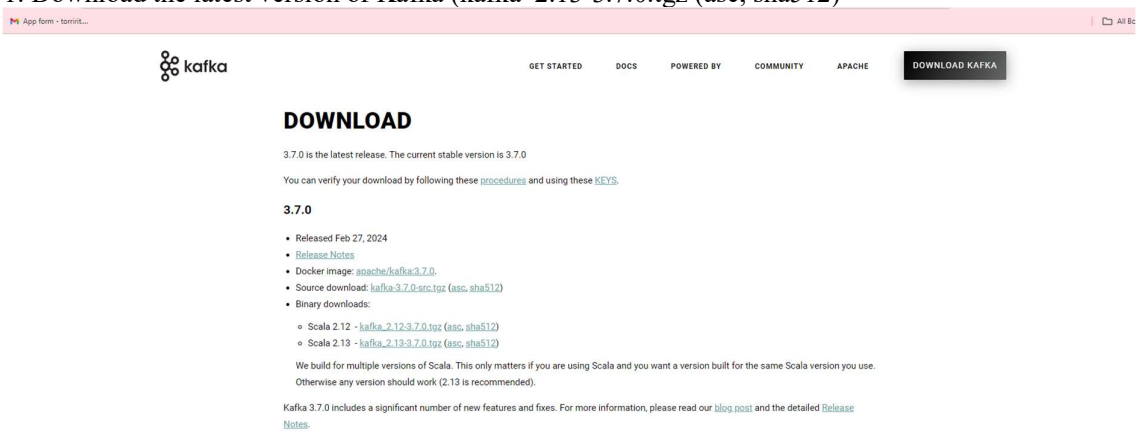
```
Number of Posts: 25
Number of Likes: 25
```

Task 4: Setting up a data streaming pipeline with Kafka to integrate data for data analysis

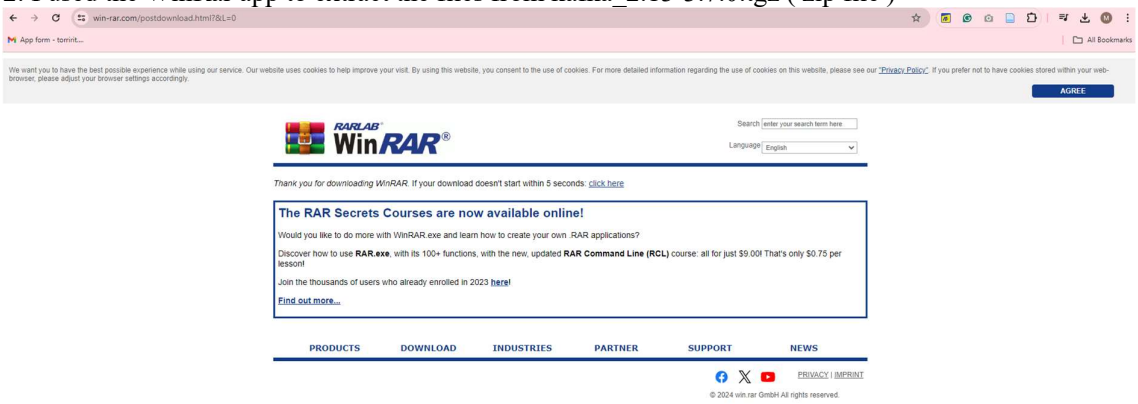
Kafka offers a base, for constructing data streaming systems that guarantee data accuracy and resilience crucial for tasks, like data analysis and real-time processing.

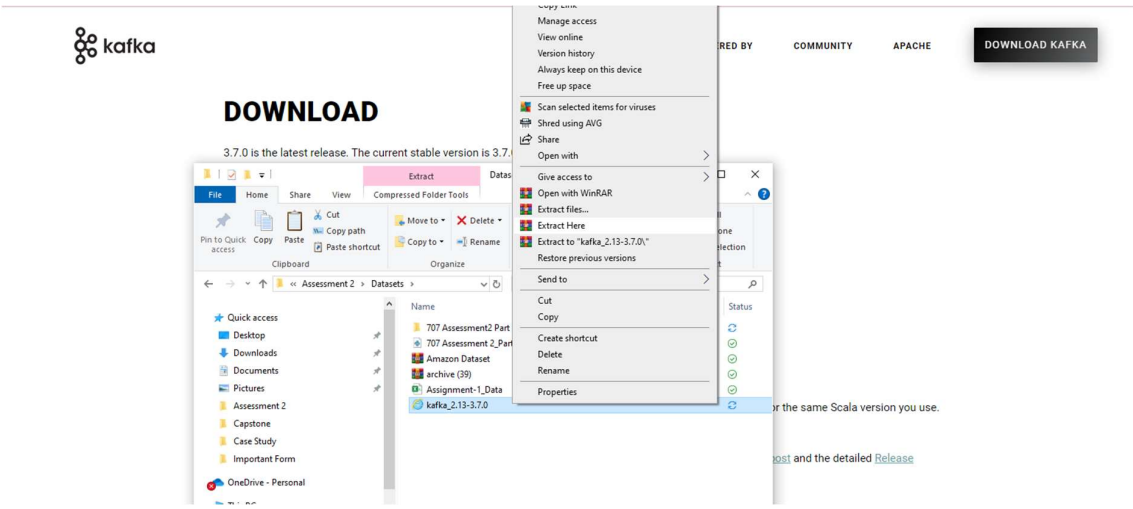
Steps on setting up a data streaming pipeline with kafka

1. Download the latest version of Kafka (kafka_2.13-3.7.0.tgz (asc, sha512)

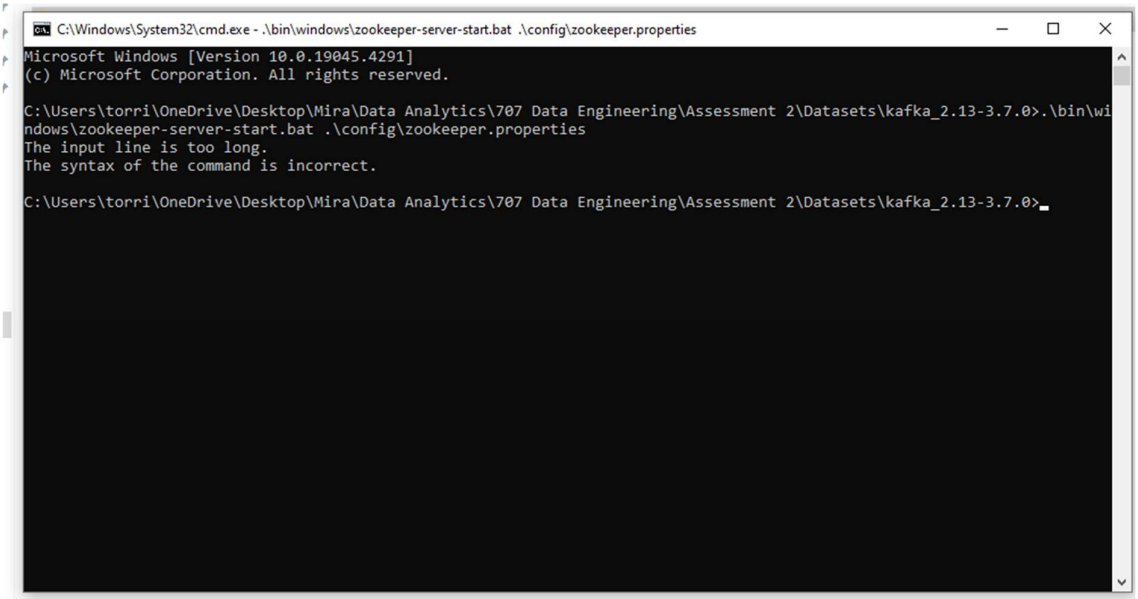


2. I used the WinRar app to extract the files from kafka_2.13-3.7.0.tgz (zip file)

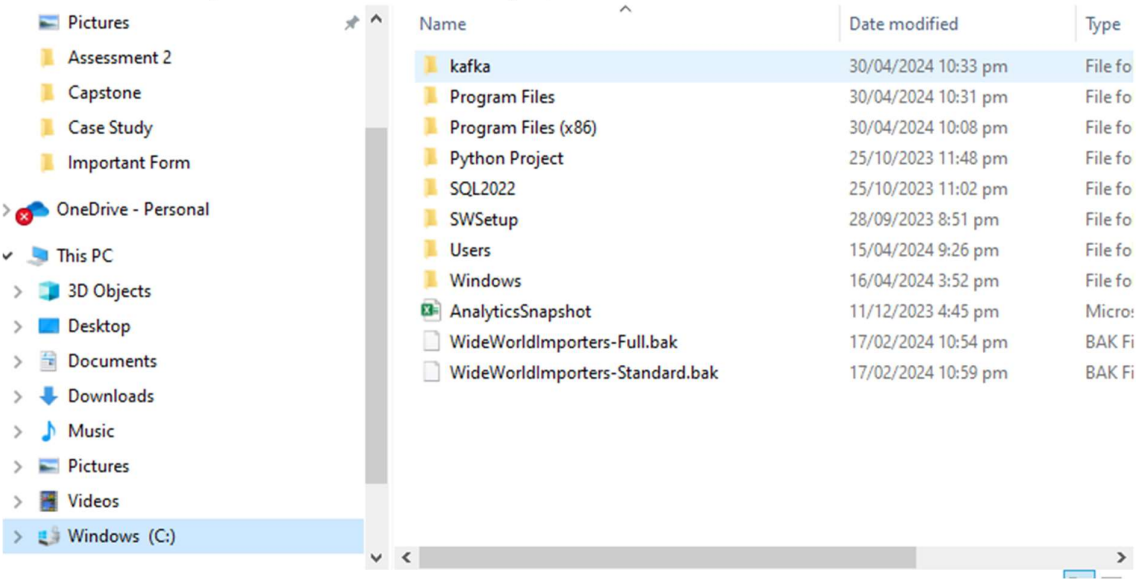




3. Move the extracted file to drive C to avoid command line errors (The input line is too long) shown below

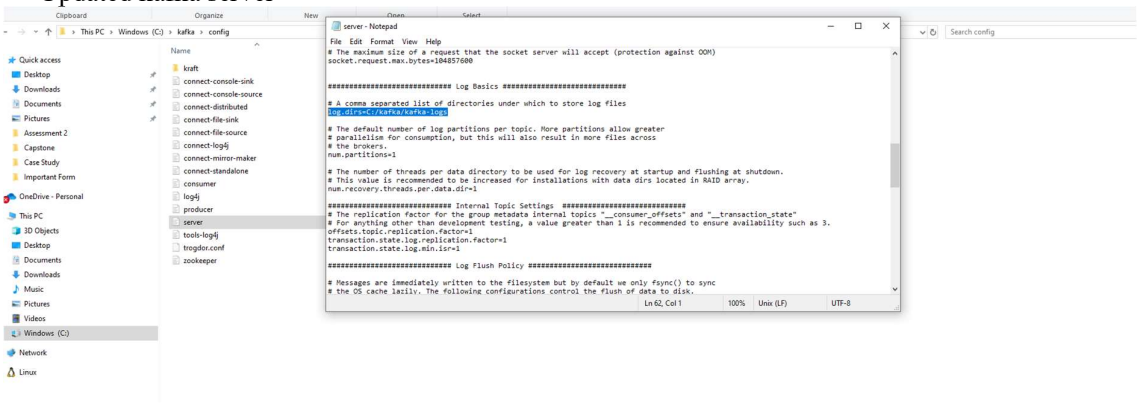


** Putting kafka in Drive C will simplify the command

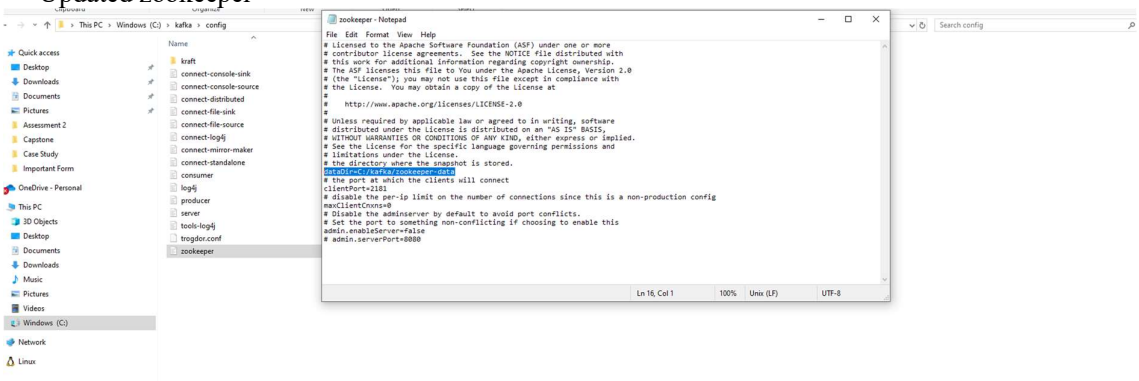


4. Update the kafka server and zookeeper in the config file

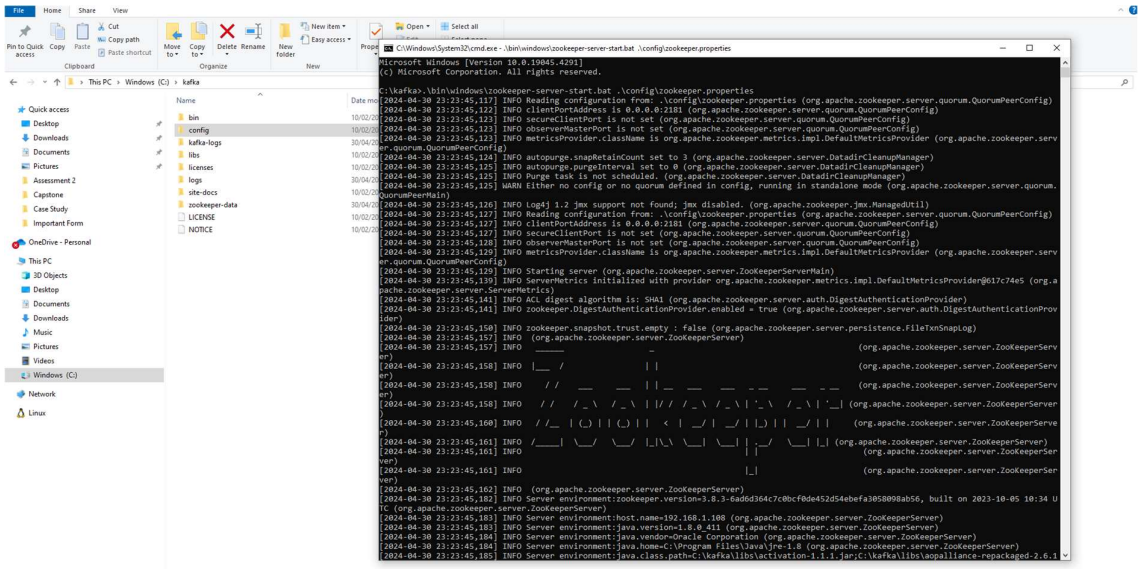
** Updated kafka server



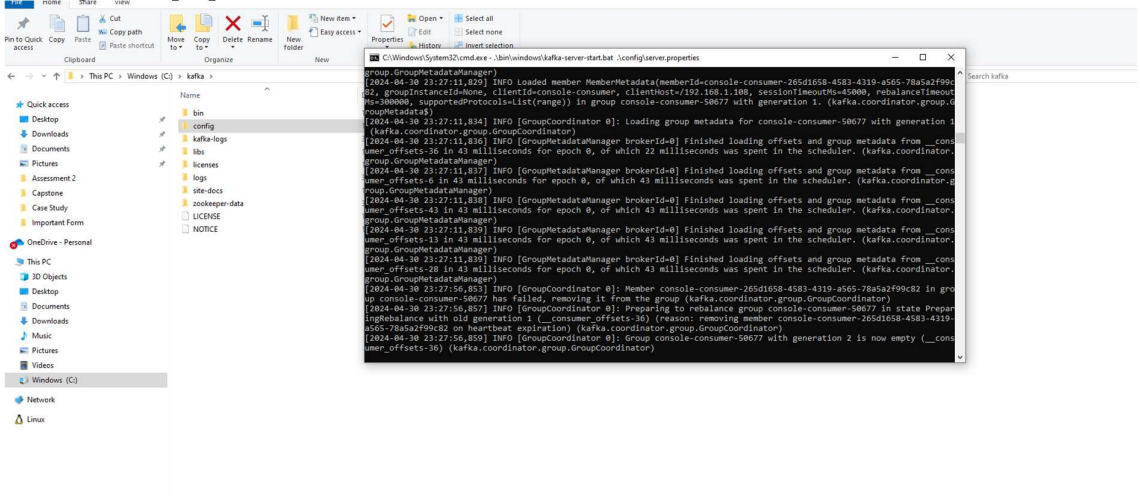
** Updated zookeeper



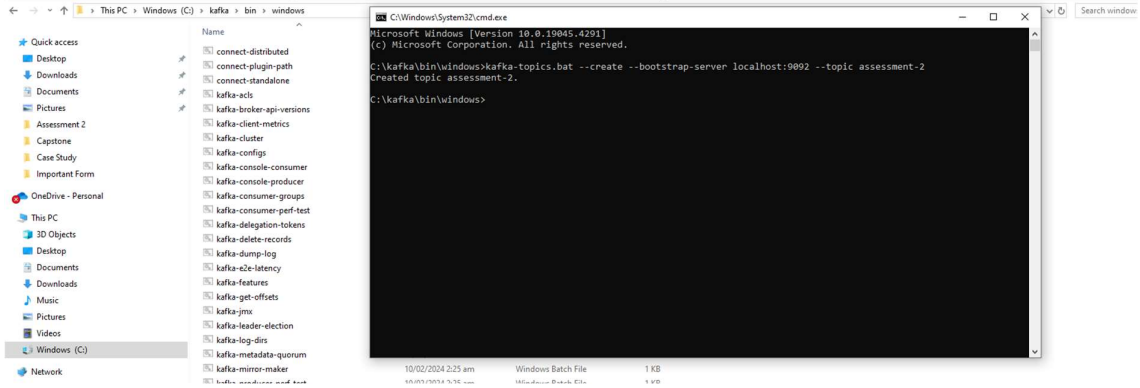
5. Run the zookeeper in the command line (cmd) using the code .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties



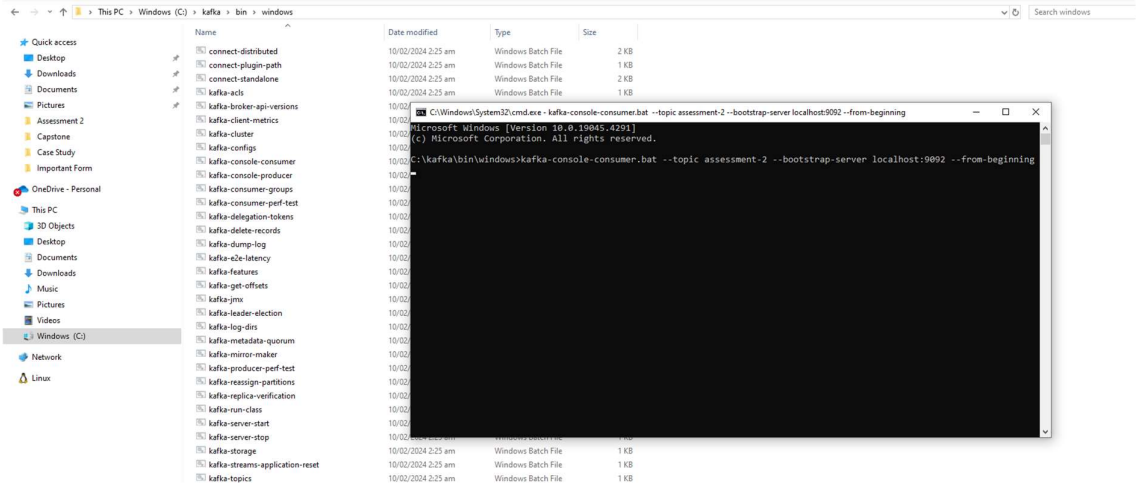
6. Run the kafka server in the command line(cmd) using the code .\bin\windows\kafka-server-start.bat .\config\server.properties



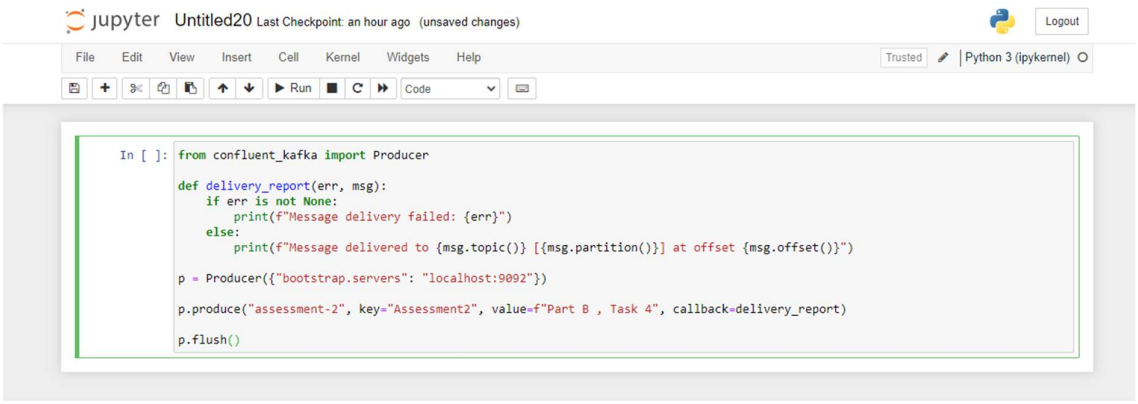
7. Create a kafka topic that serves as a table in kafka. Run the code kafka-topics.bat --create --bootstrap-server localhost:9092 --topic assessment-2 in bin/windows path.



8. Run the consumer command in kafka which will receive the data

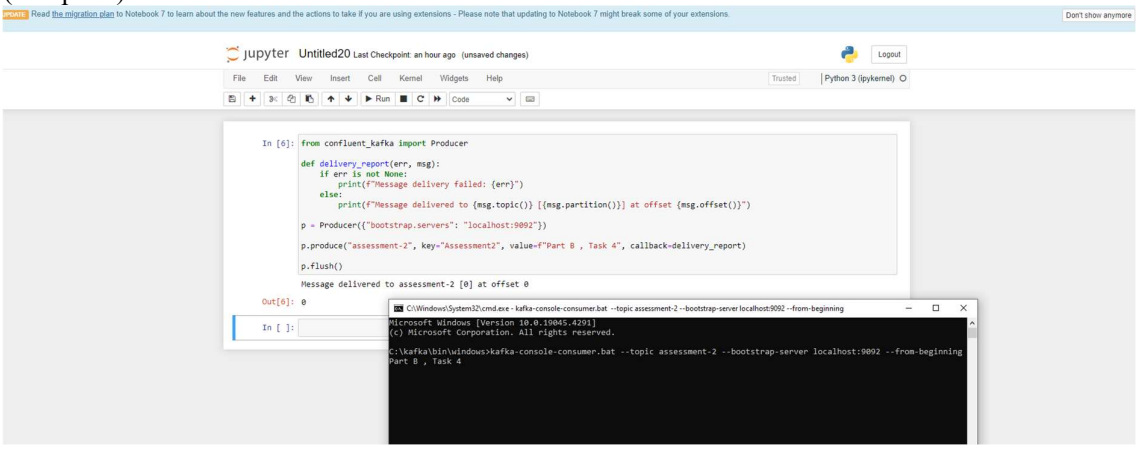


** Python code that will send data to kafka

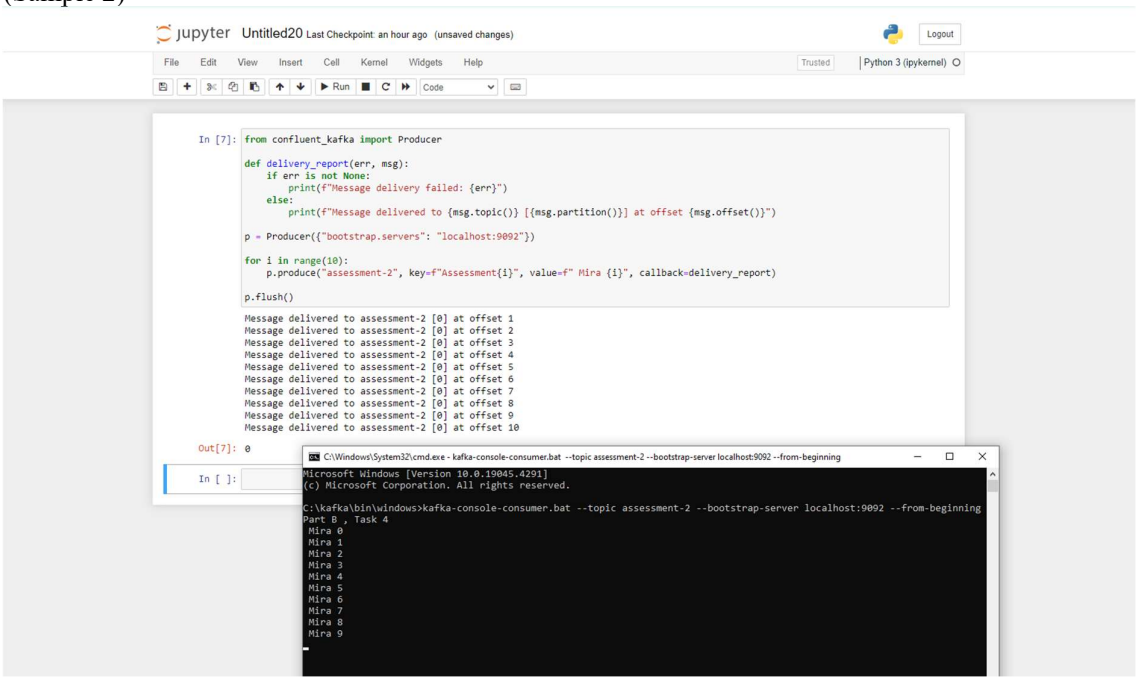


**** Sending data from Python to kafka**

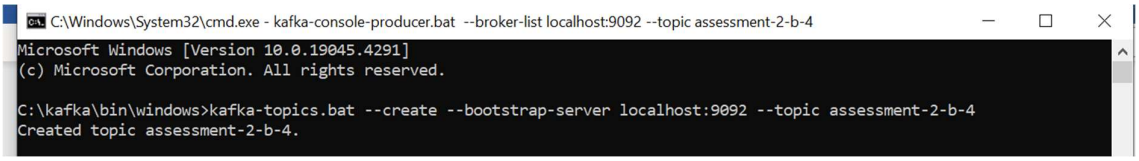
(Sample 1)



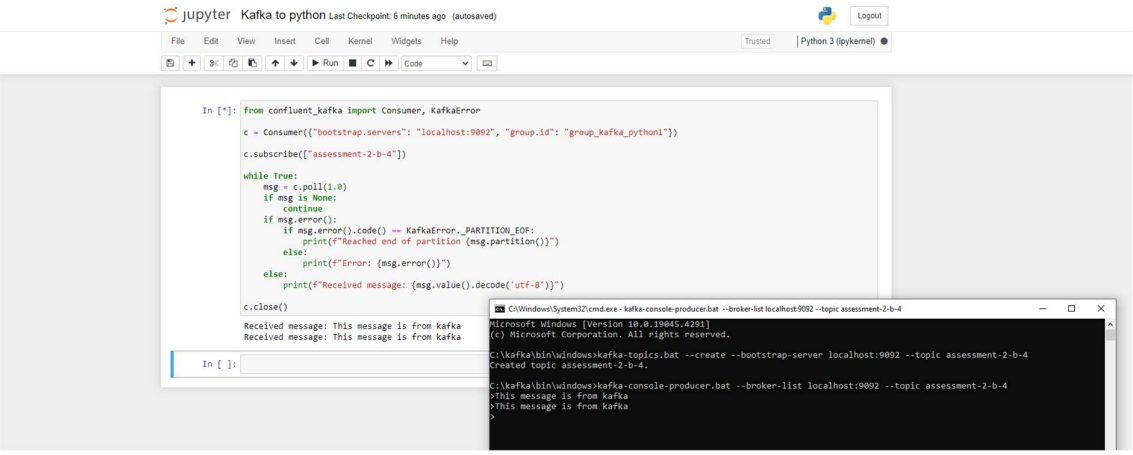
(Sample 2)



** **Sending data from kafka to Python** – run the code kafka-console-producer.bat --broker-list localhost:9092 --topic assessment-2-b-4 to allow sending data from kafka



** Python code to retrieve data from kafka



Conclusion:

Data integration is essential in business, especially with big data. However, it can be a very risky process if not done correctly. Most errors occur in connection and configuration, thus requiring attention to detail. Knowledge of the different integration processes will also help manipulate and treat the data. Data cleaning is always the first step to ensure the quality of the data. As per experience in the tasks, integration planning is crucial to facilitate the integration and avoid errors during the process. Familiarization with the data structure is also needed to choose the proper storage or database.

References

Ahmedov, A. (n.d.). Market Basket Analysis: Analyzing Consumer Behaviour Using MBA Association Rule Mining. Kaggle. https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis/data

Anil. (n.d.). data.world. https://data.world/anilsharma87

E-Commerce Data. (n.d.). Customer Segmentation & Recommendation. Kaggle. https://www.kaggle.com/code/farzadnekouei/customer-segmentation-recommendation-system/input

The Devastator.(n.d.). E-Commerce Sales Dataset.Kaggle. https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data

Ramos, G. (n.d.). E-commerce Business Transaction: Sales transaction of a UK-based e-commerce (online retail) for one year. Kaggle. https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business/data