Design Considerations / Assumptions:

1. For one time analysis and to fulfill current requirements, the simplest design for dimension tables used is SCD Type-1.
2. As the source data structure doesn’t provide historical data (using DATE/TIME) ; time dimension is not considered in the fact tables Order/Inventory/Order-Item. So the design of the fact tables is simply Append-Only.
3. No CDC ( Change Data Capture ) or BAU scheduling is done.
4. As the technology was not specified, I have used Google Cloud as :
   1. Google Cloud Storage : To put source csv files
   2. Google Big Query : To load Staging data
   3. Google Big Query : To load DWH data
   4. ETL Pipeline is written in python
   5. Cloud function is used to execute the pipeline

Answers :

1. Create a data model for the datawarehouse.
   1. Created Staging area to hold source data as is.
   2. Created DWH Fact and Dimension tables

DDLs are available in the Code-Artifact mentioned.

1. Create a data pipeline to load the data into the datawarehouse.
   1. Python is used to load CSV from GCS into BQ Staging area.
   2. Once the data has landed on BQ, its better to use BQ SQL to load it into BQ-DWH.

So, Staging-DWH is written in Stored Procedure and then called from python scipt.

1. Create a SQL query to calculate the revenue per customer.

SELECT customer\_name, ROUND( SUM(order\_total\_price) ,2) as revenue\_per\_customer

FROM vf-grp-gbissdbx-dev-1.customer\_service.Fact\_Order o

LEFT JOIN vf-grp-gbissdbx-dev-1.customer\_service.Dim\_Customer c ON o.order\_customer\_id = c.customer\_id

GROUP BY 1

ORDER BY 2 DESC ;

1. Create a SQL query to calculate the value of goods stored in the different inventory locations.

SELECT i.inventory\_location, ROUND(SUM(product\_customer\_price),2) AS value\_of\_goods

FROM vf-grp-gbissdbx-dev-1.customer\_service.Fact\_Inventory i

LEFT JOIN vf-grp-gbissdbx-dev-1.customer\_service.Dim\_Product p ON i.inventory\_product\_id = p.product\_id

GROUP BY 1

ORDER BY 2 DESC ;

1. Create a SQL query to calculate the average order value per customer.

--This is considered as total(order value) of a customer / number of orders of a customer

SELECT order\_customer\_id, SUM(order\_total\_price) / COUNT(order\_id) as avg\_order\_value\_per\_customer

FROM myfirstproject-269913.customer\_service.fact\_order o

GROUP BY 1

ORDER BY 2 DESC;

1. Run the data pipeline and execute the SQL queries on the datawarehouse.

Done. Screenshots attached.

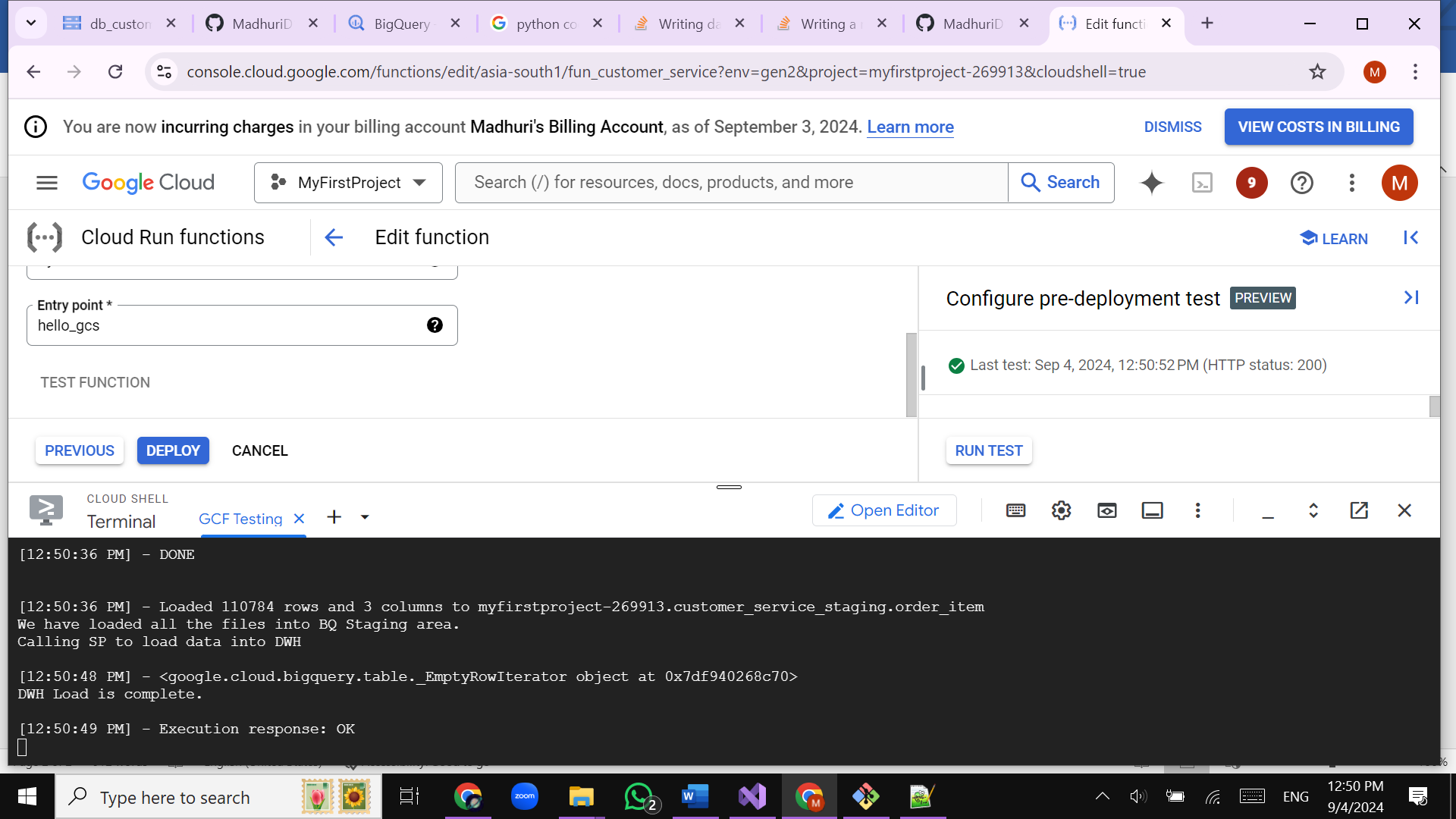
1. Have some thought how to improve the usage of the datawarehouse.
2. To enable historical analysis, source needs to provide DATE for Inventory data so that month-on-month / year-on-year reports of inventory can be created.
3. For historical reporting ( as-was + as-is), its recommended to create SCD Type-2 tables for both the dimensions – Customer & Product.
4. Order Fact table is generally accumulating snapshot table; in which the same order record gets updated as the order progresses; resulting in the change of order status and shipping/delivery/payment dates. Updates work very well on traditional DBs like Oracle, TD however updates are very slow on Cloud DB like Bigquery. So we can model this table as “periodic snapshot table” and append the data for specified period (daily/monthly). This will result in multiple records for one order & while reporting, we can always consider the latest record of the Order.
5. Once DATES are available in Fact tables, we can partition them based on the time grain requirement by the users ( Monthly/Daily/Weekly)

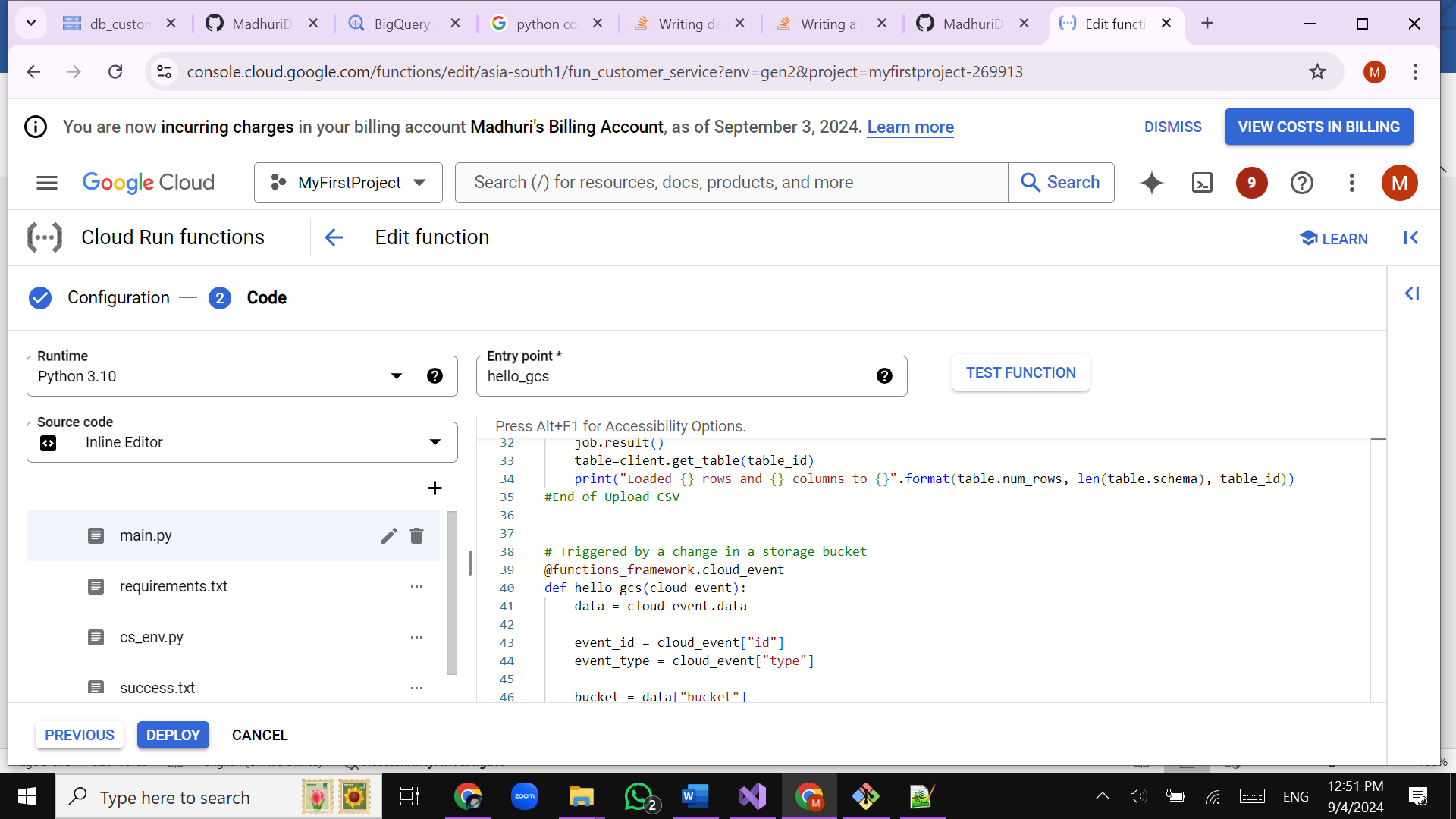
Code Artifacts:

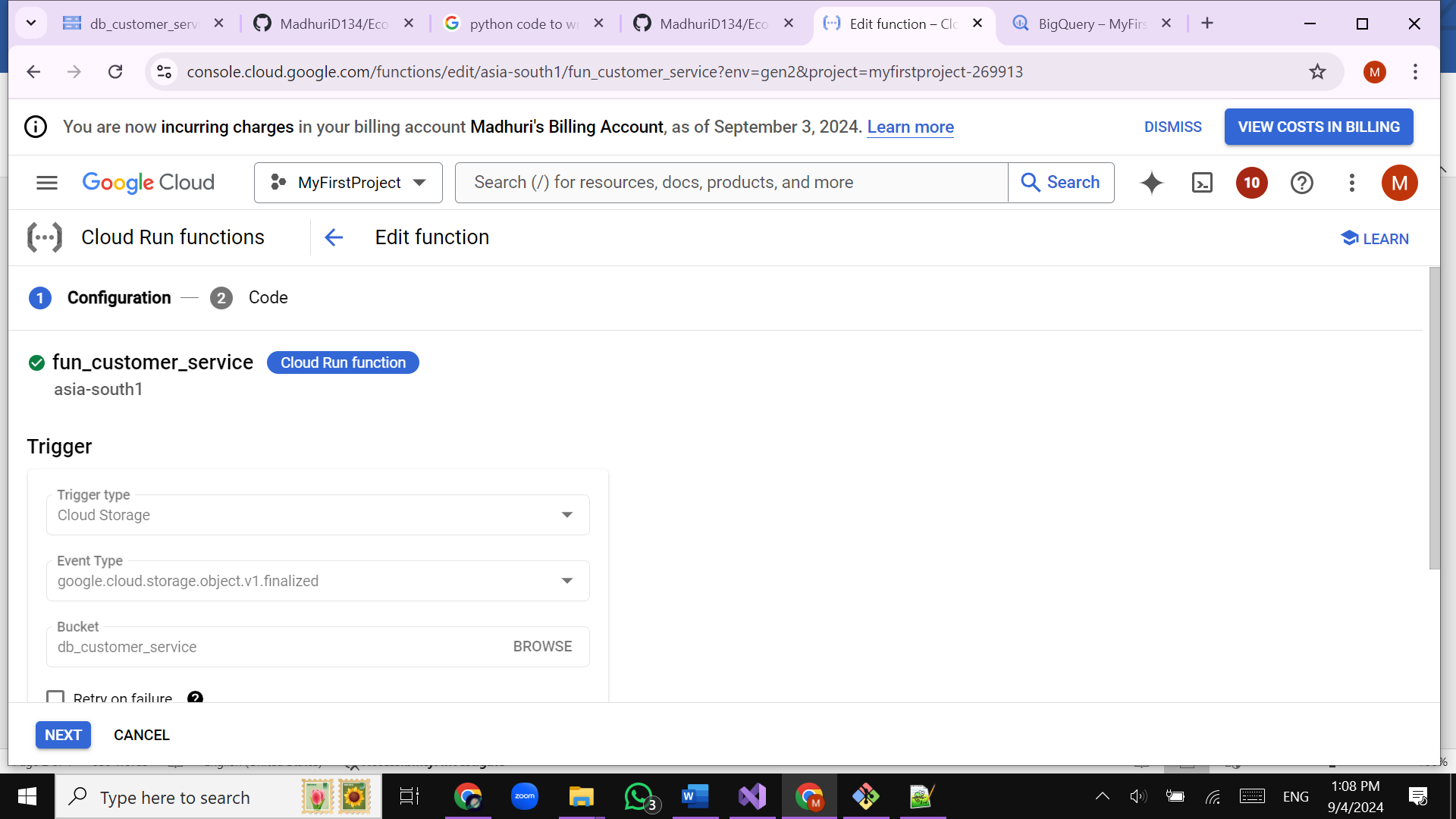
GIT Repository has following artifacts:

https://github.com/MadhuriD134/Ecommerce-DWH

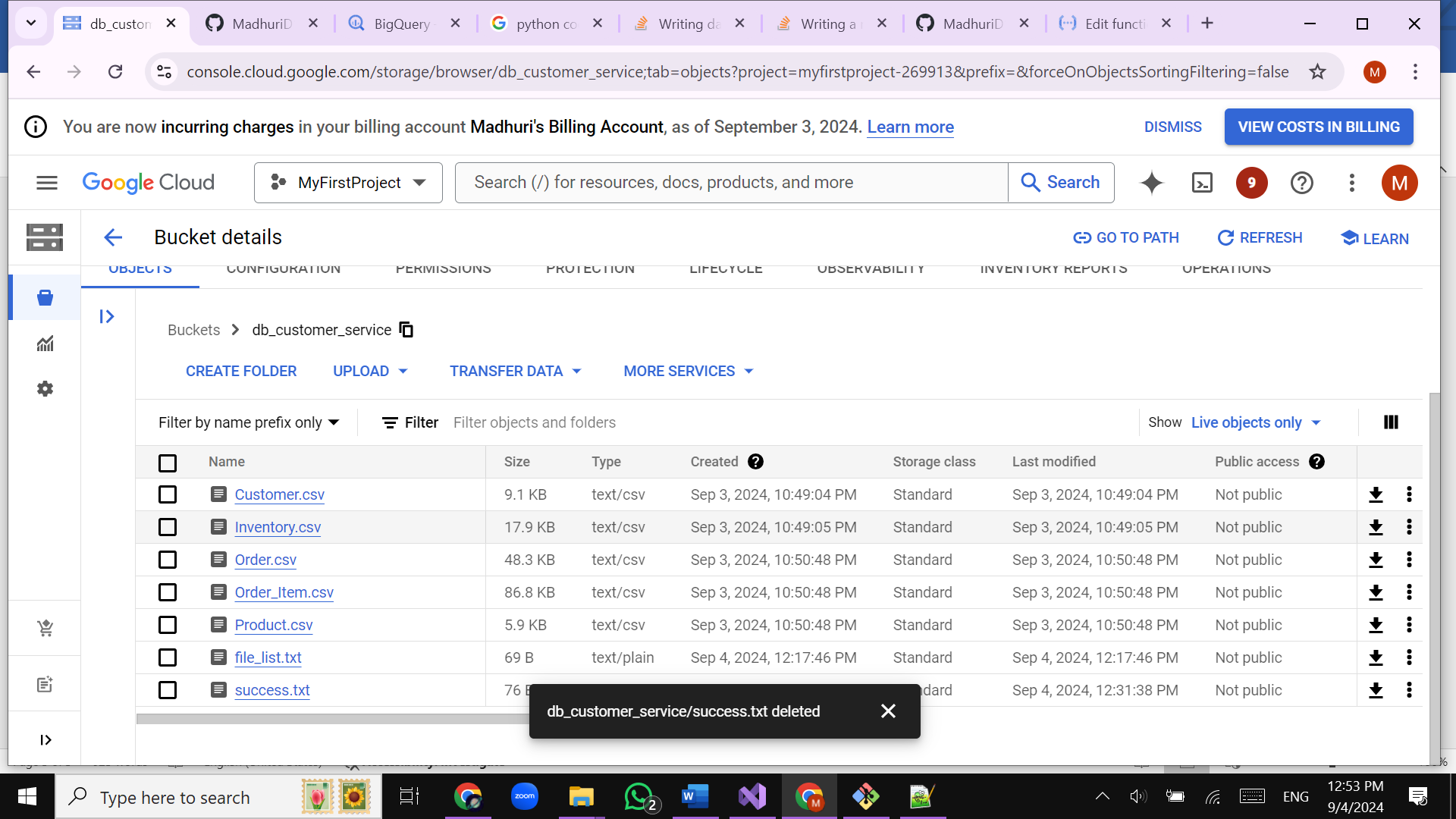
1. Staging-DDL.sql : DDL for Staging tables
2. DWH -DDL.sql : DDL for DWH tables
3. SP\_Load\_DWH : DDL of Stored procedure which loads from Staging into DWH
4. Python scripts to be used in Cloud Function
   1. CF-main.py : Main cloud function
   2. CF-cs\_env.py : Has env variables like Staging-Dataset, src-bucket etc.
   3. CF-requirements.txt : Python modules to install
5. File\_list.txt : It has the lists of all source files which need to be processed.
6. Screenshots of GCS Bucket & Cloud function created

Cloud Function Test run execution : 





GCS Bucket :



BQ Query :

