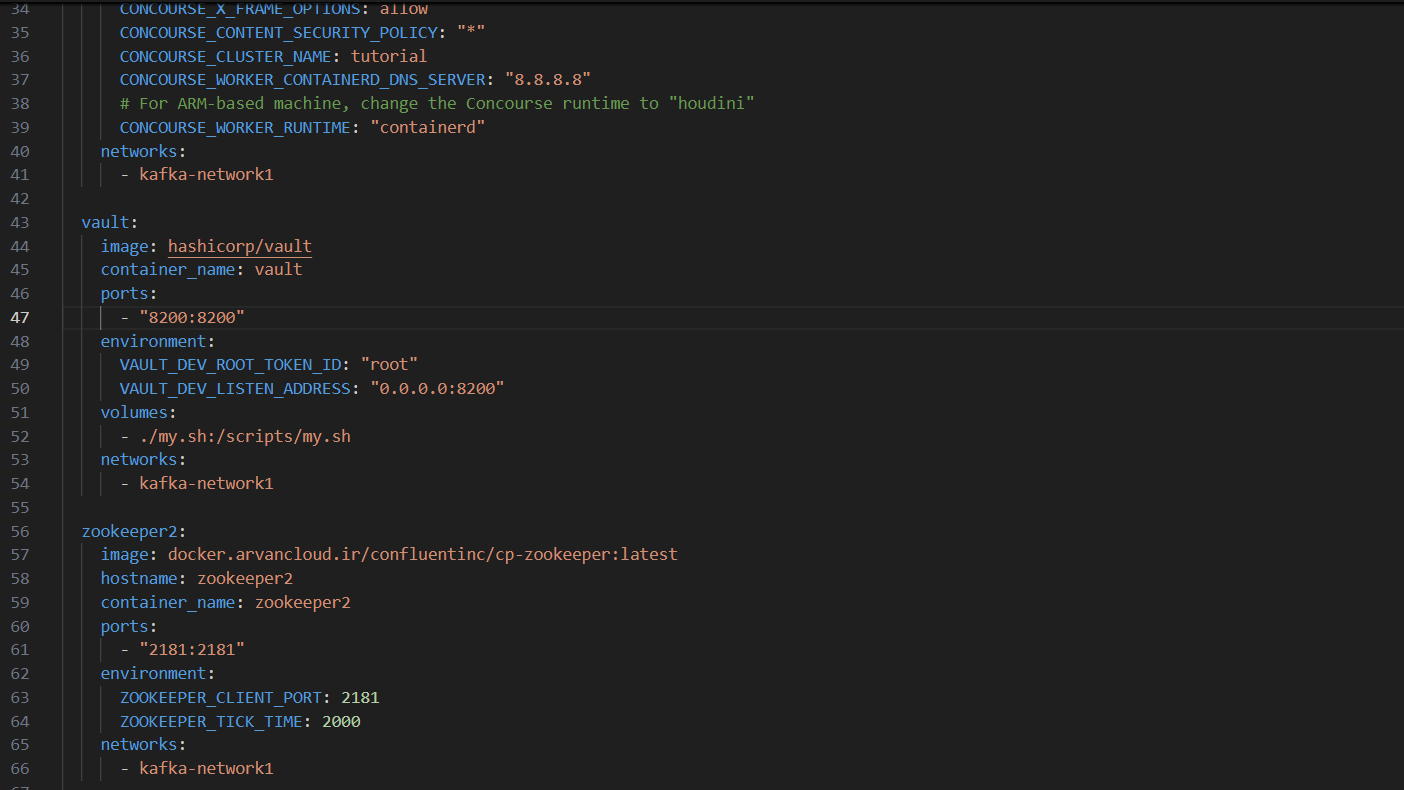
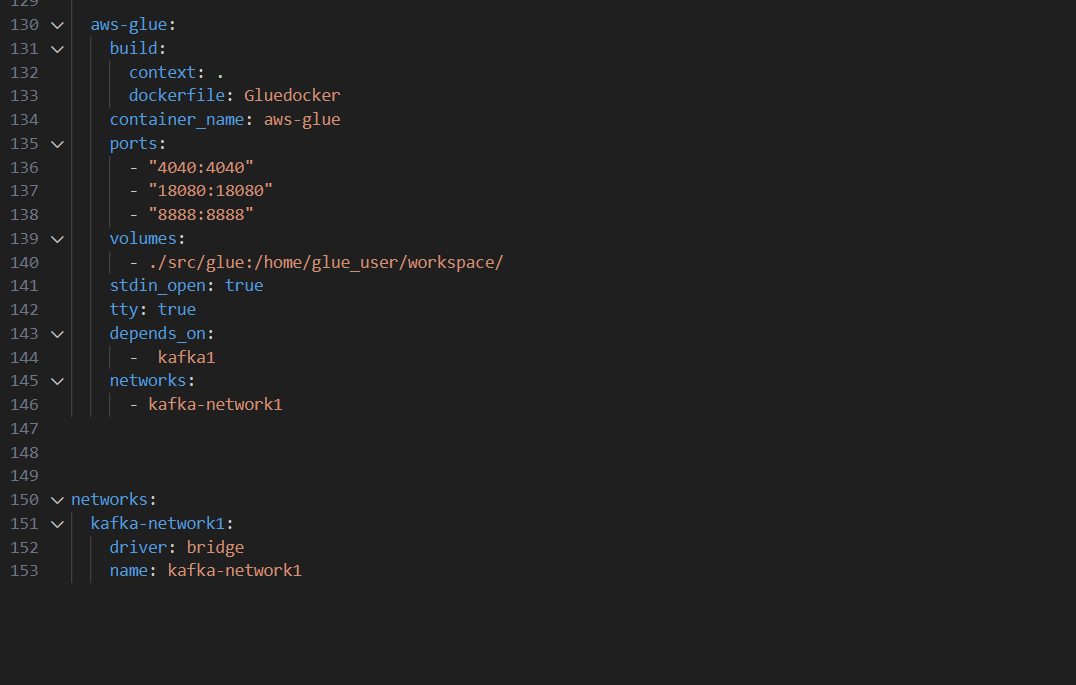
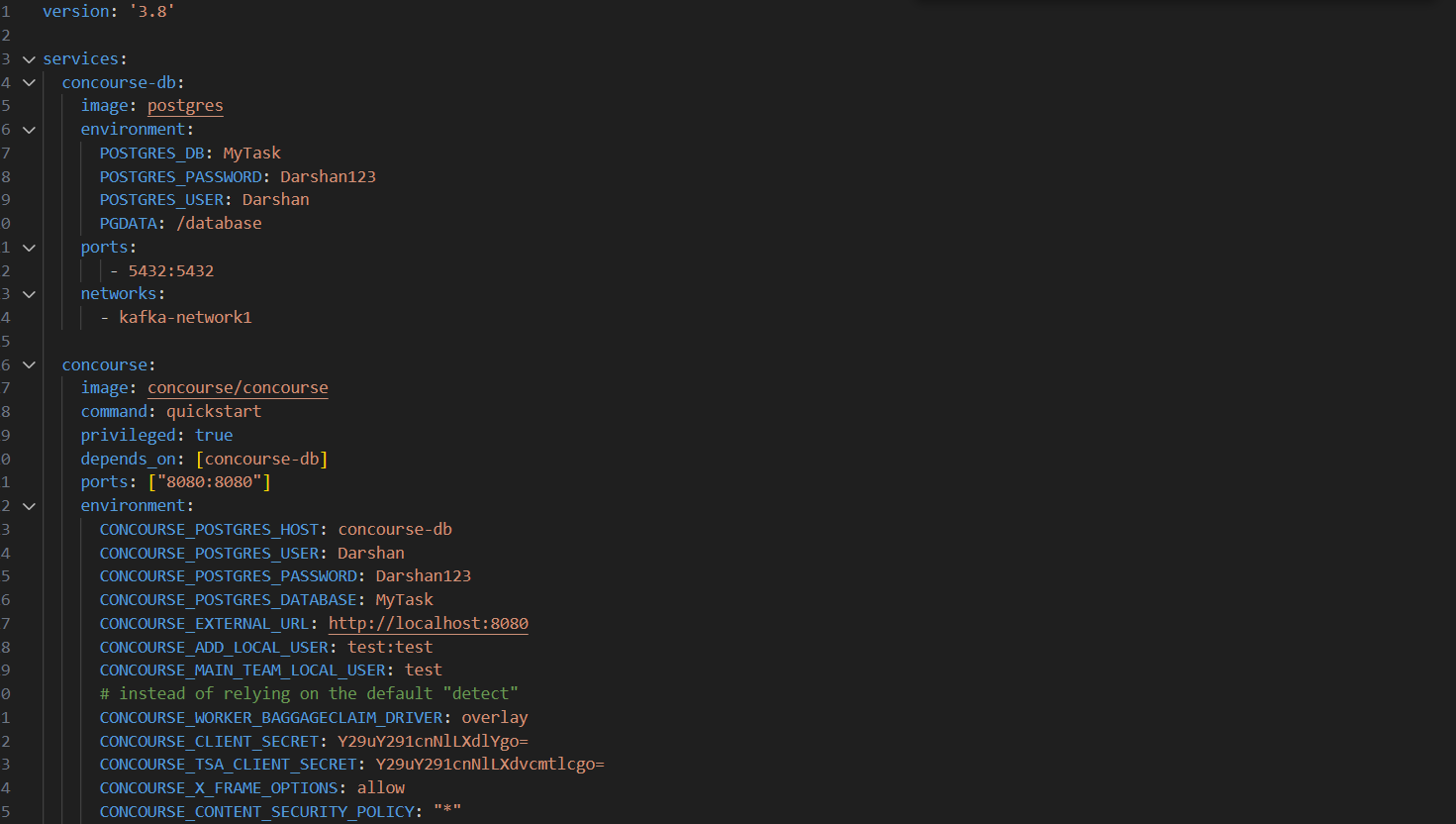
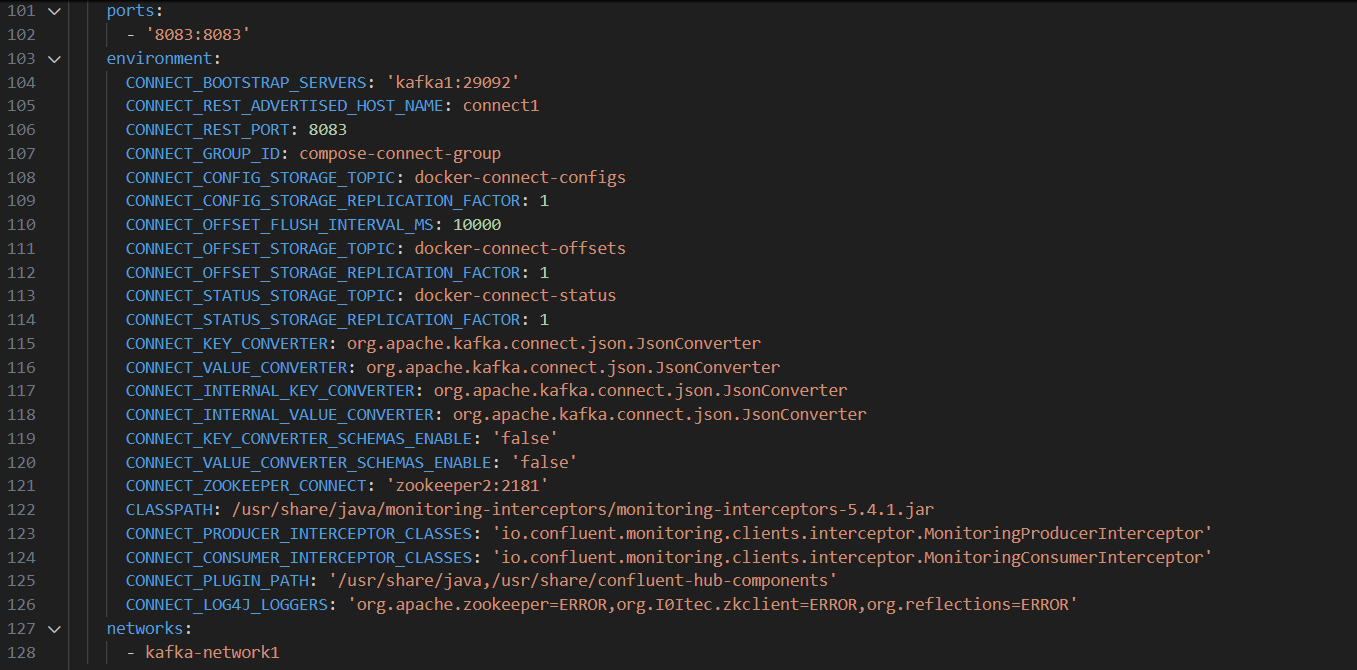
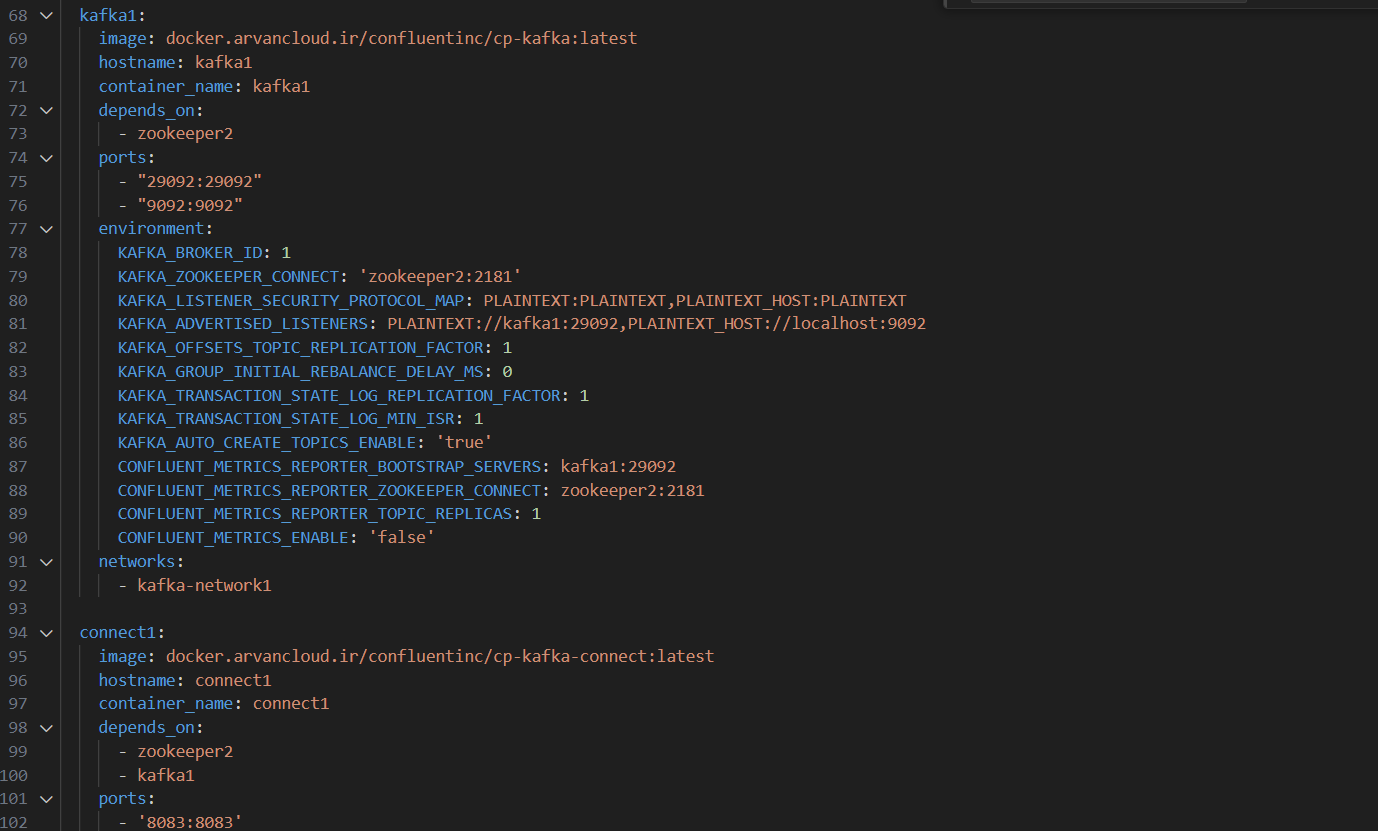
**Task 7 Documentation**

* Combine all tasks into one
* Use Kafka and Glue to stream screener data into Postgres tables.
* Before inserting data into sink table perform transformations (average of each row) on the data from the source.
* Only show transformed data in the sink table.

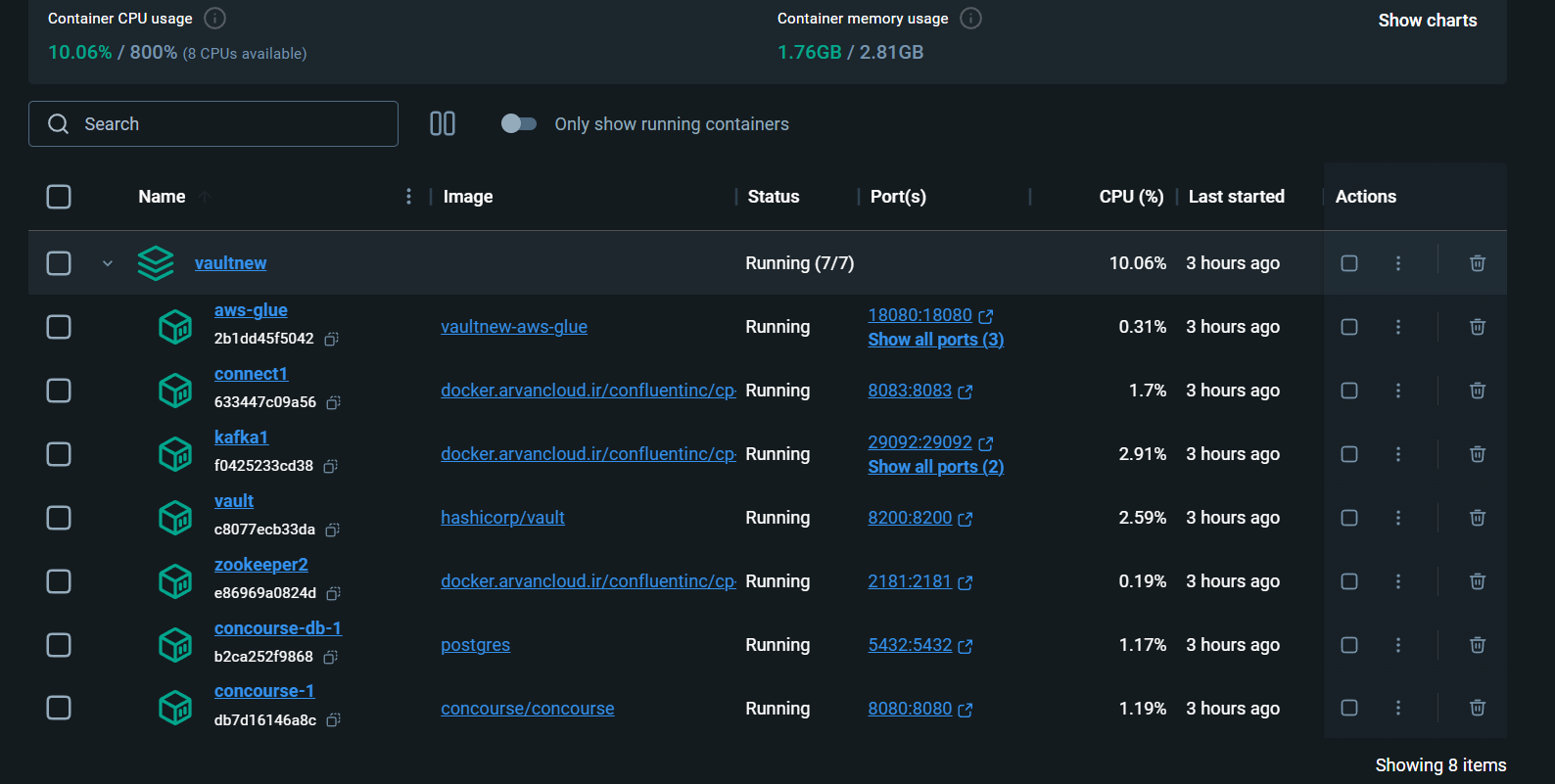
**Step1:**

**Update the docker-compose file to making sure that kafka broker and connect, aws-glue attached to it and as images were already there so just used following query.**

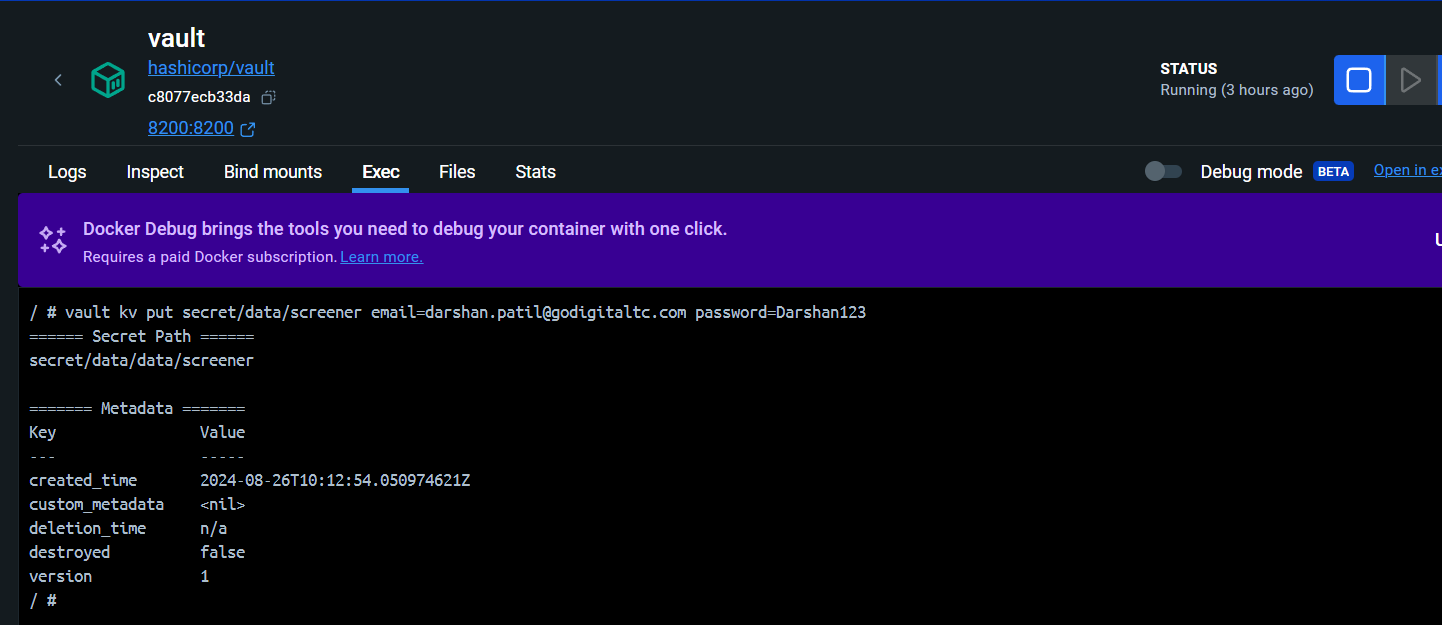
**Docker-compose:**

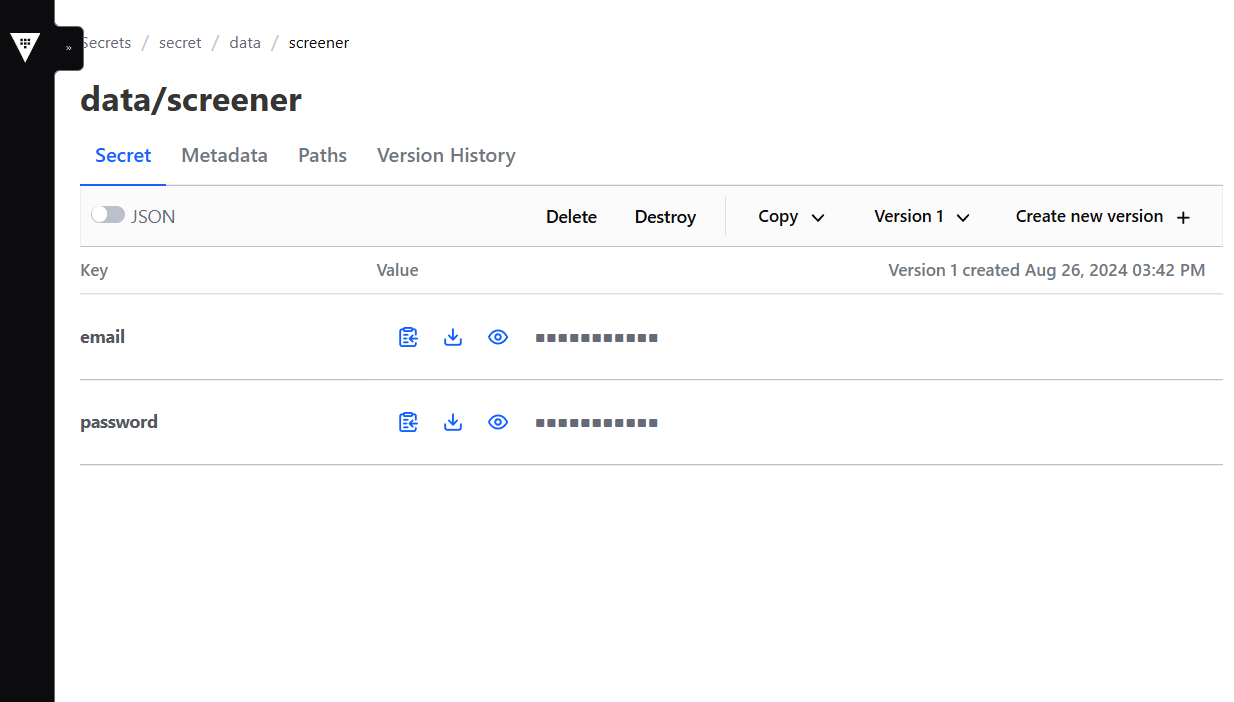
****

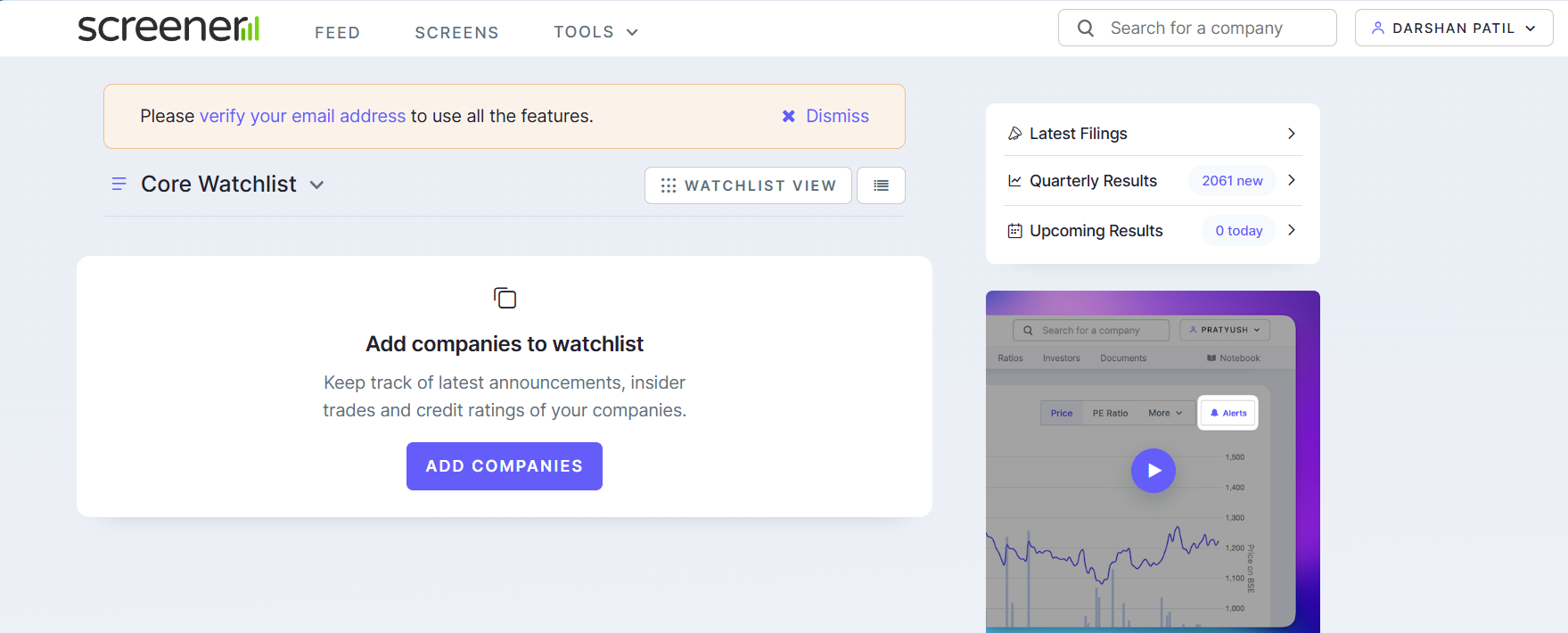
**docker-compose up -d --build**

****

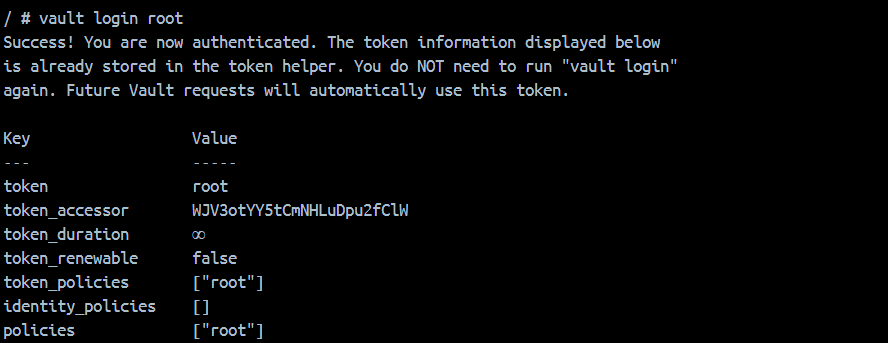
**Made a Vault Container and login into it with credentials and opened the port to url.**

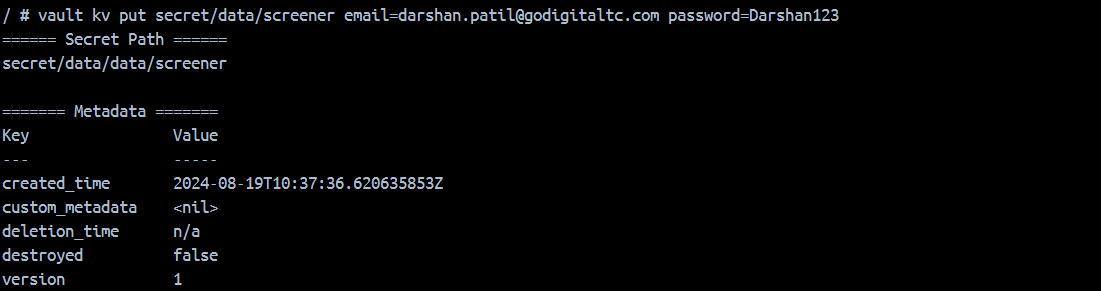
****

****

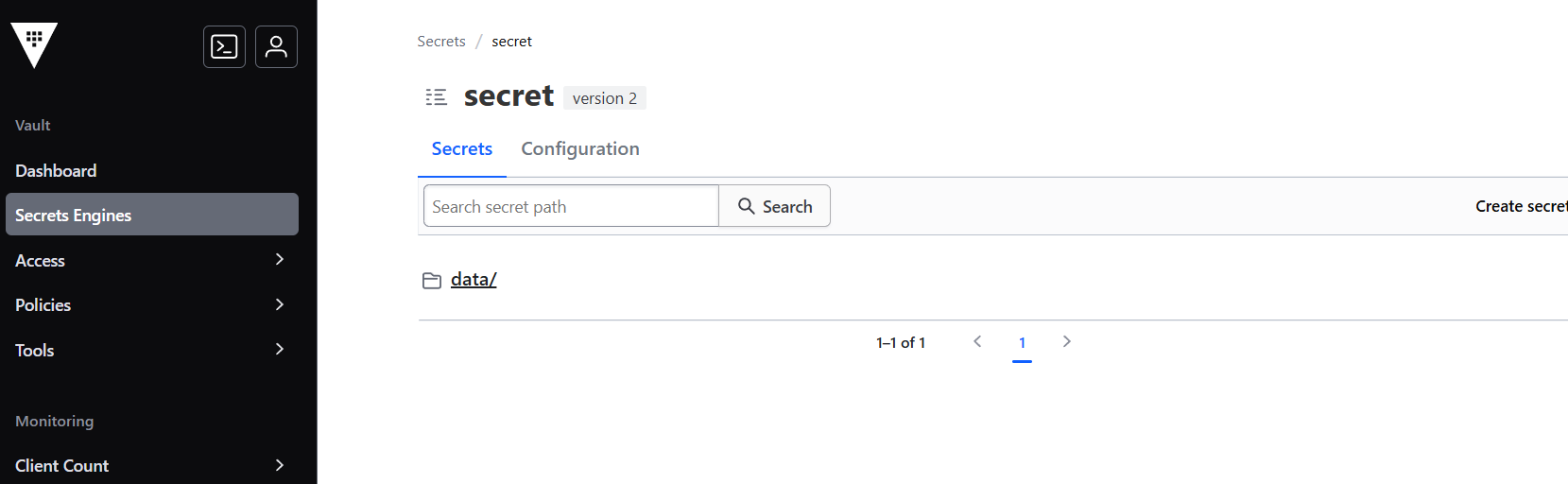


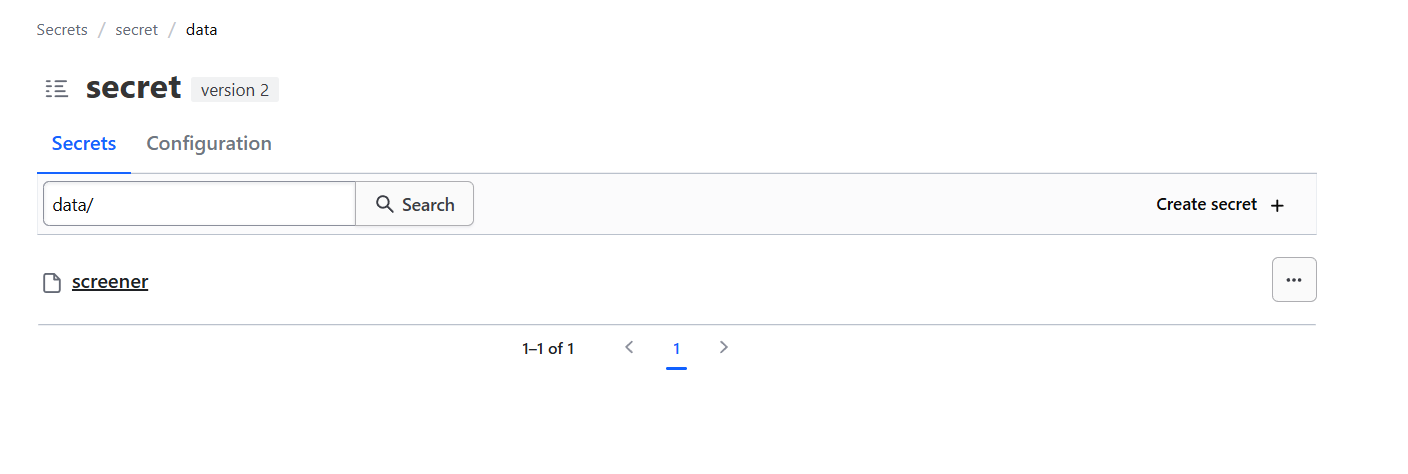
**Now come again in vault shell:**



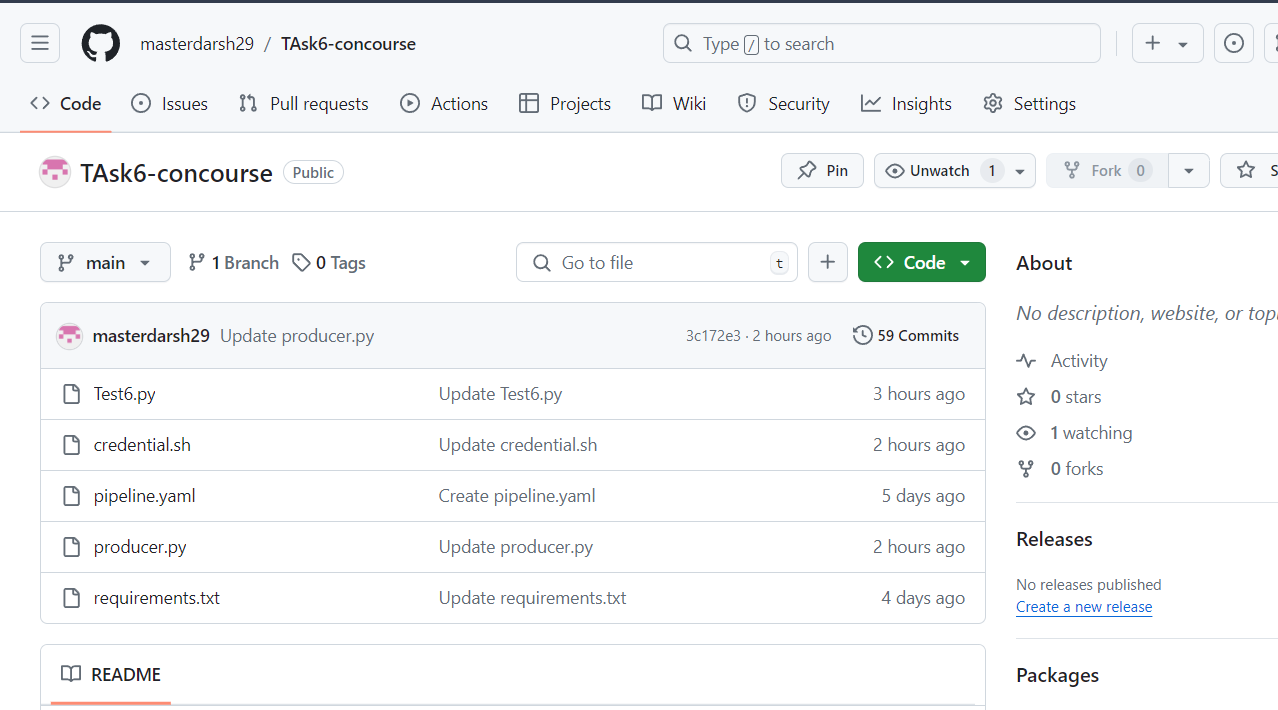


**As a result, I can see those credentials into vault:**

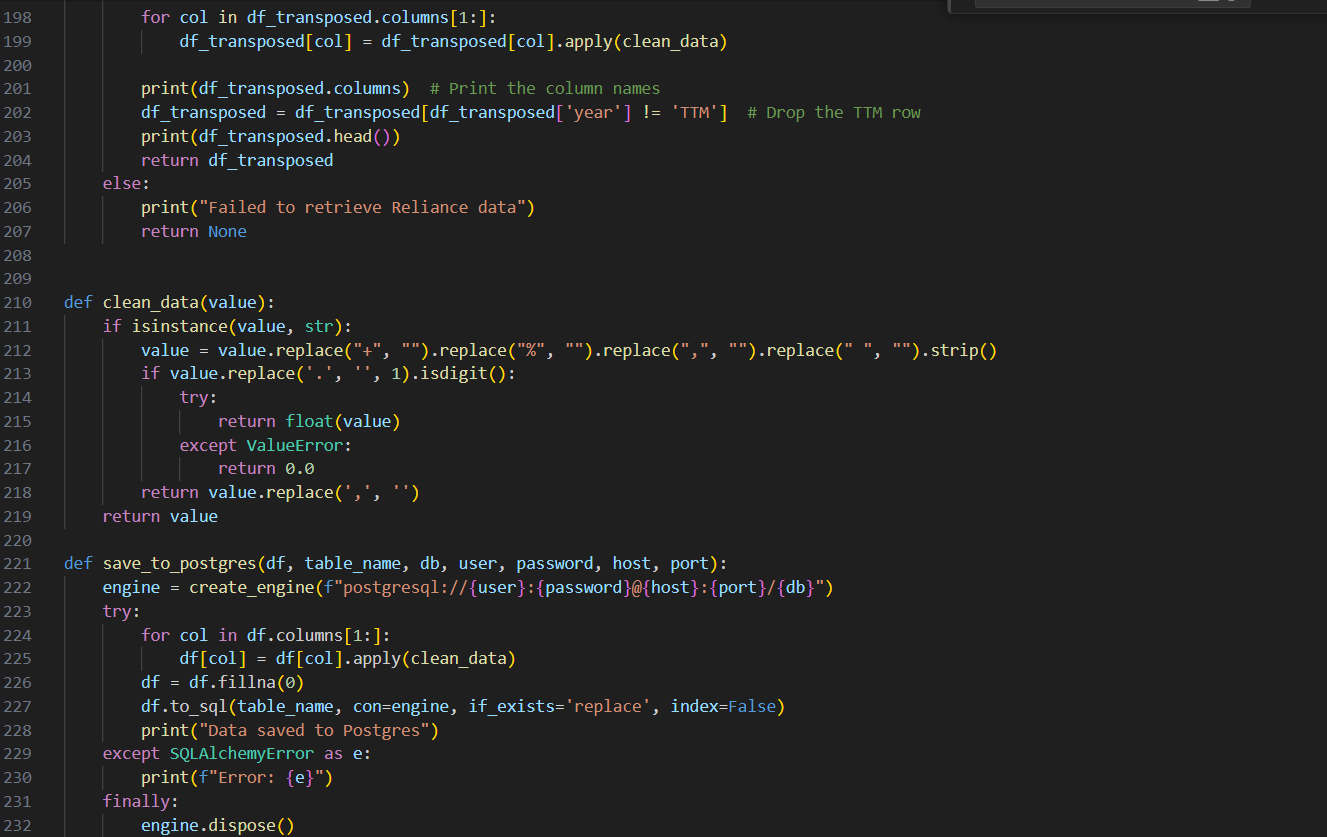
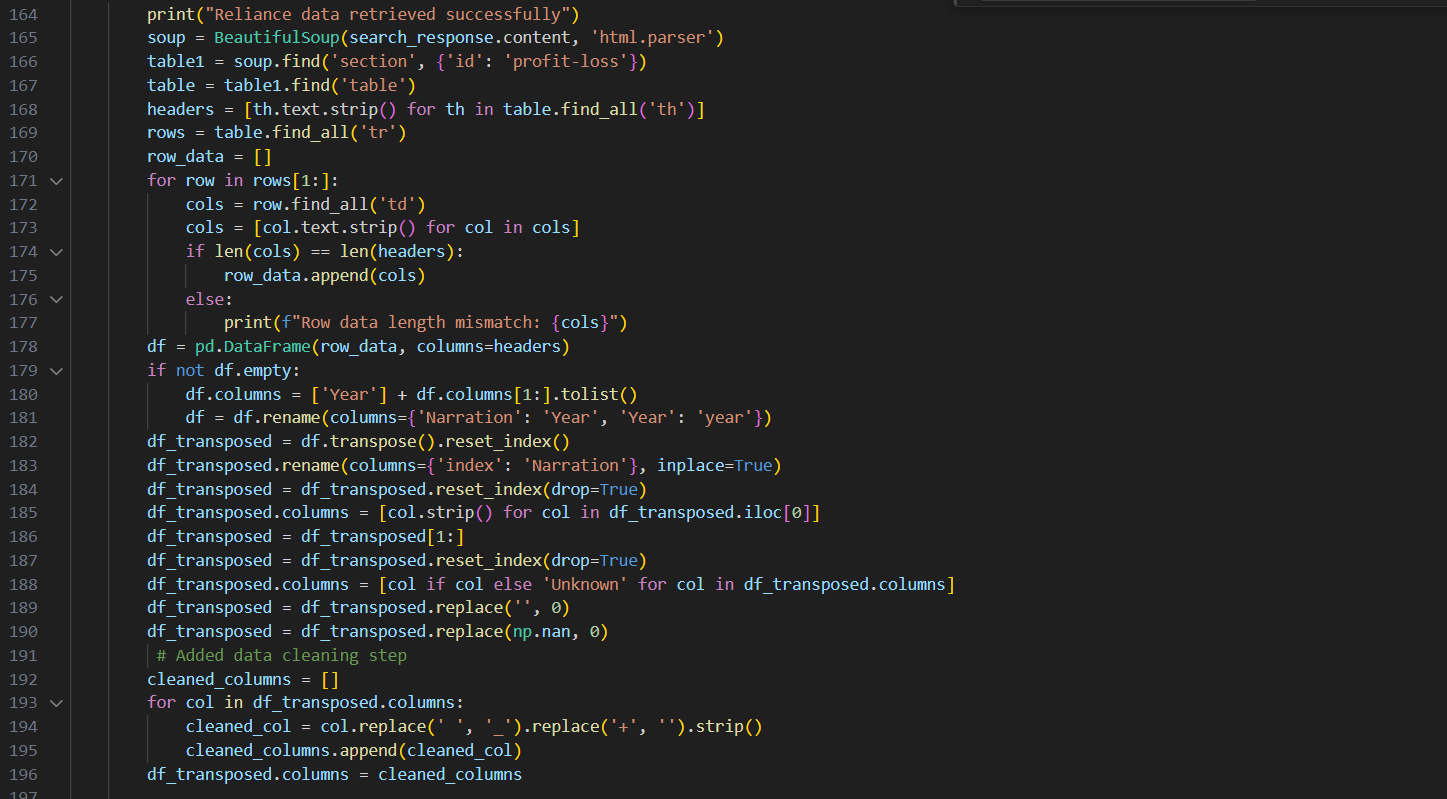
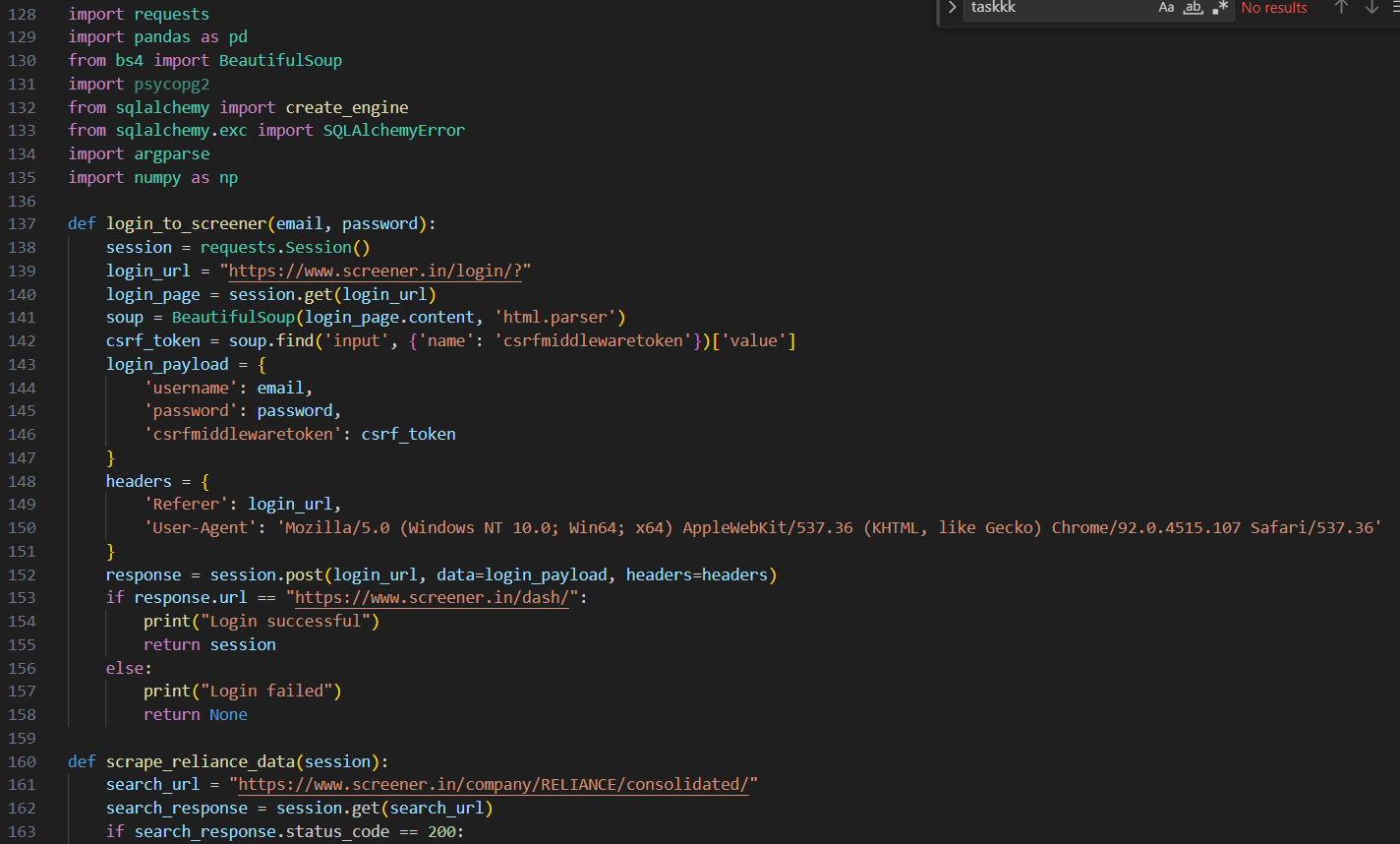




* **As per task 5&6 Vault is running on same credentials, so I’m taking same for futher task.**
* **Data is well scrapped with the use of beautifulsoup library.**
* **And as far that all the credentials and files were included into pipeline and it had given into git-repo.**

****

**Python\_script\_scrapping:**

****

**Then use fly-commands to start the concourse pipelining:**

* **commands**

**fly -t tutorial login -c http://localhost:8080 -u test -p test**

**As followed by   
destroy:**

**fly -t tutorial destroy-pipeline -p vault-demo**

**set:**

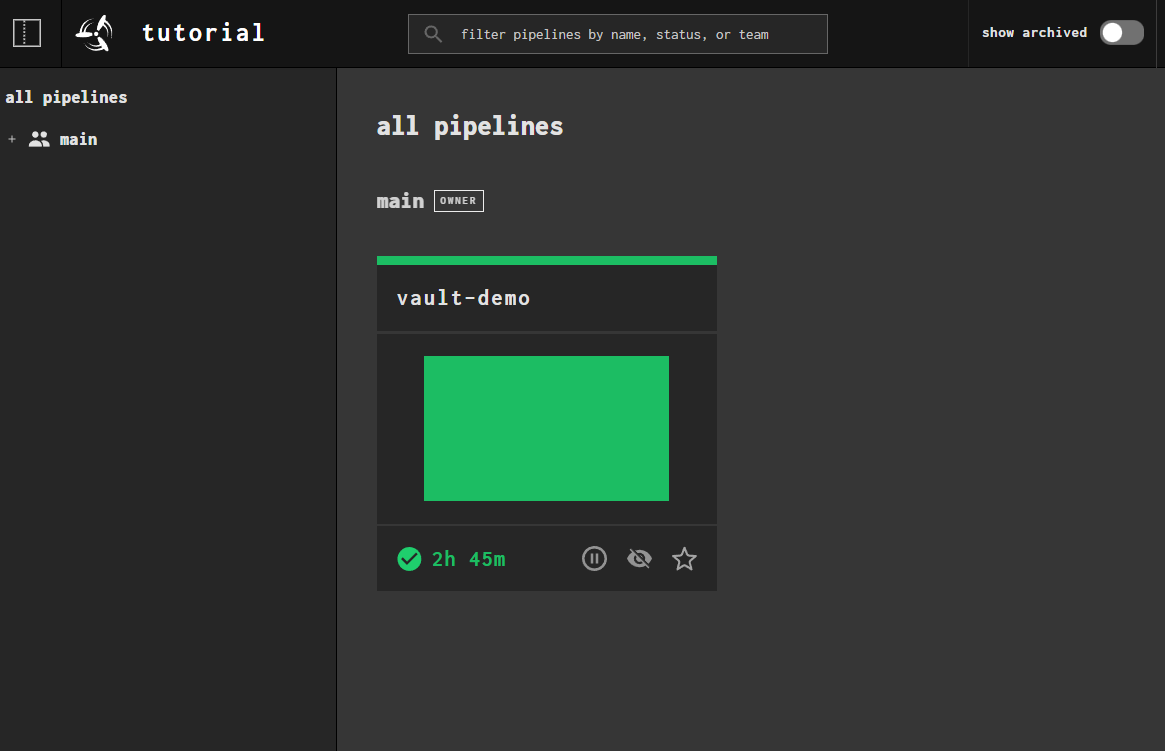
**fly -t tutorial set-pipeline -p vault-demo -c “C:/Vaultnew/pipeline.yaml”**

**unpause:**

**fly -t tutorial unpause-pipeline -p vault-demo**

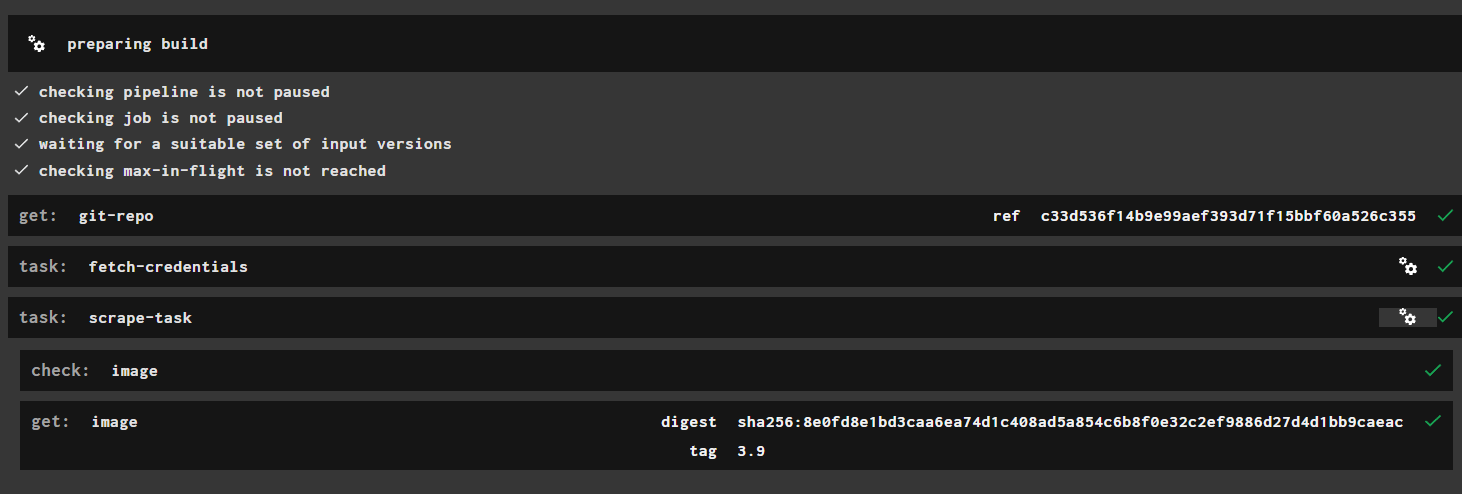
**Step5:**

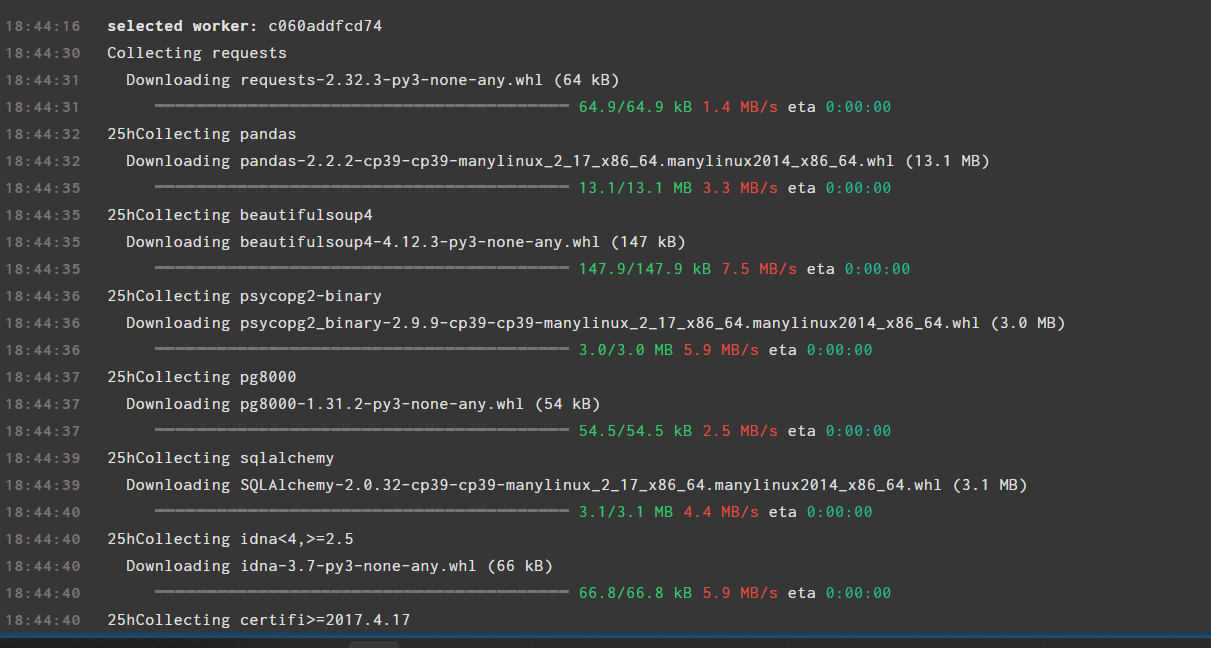
**Now open vault-demo**

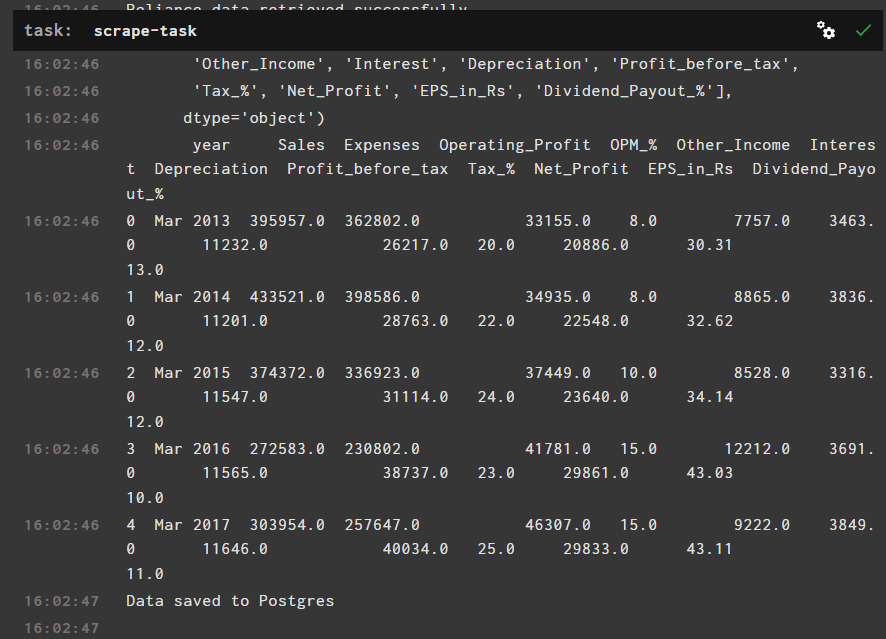


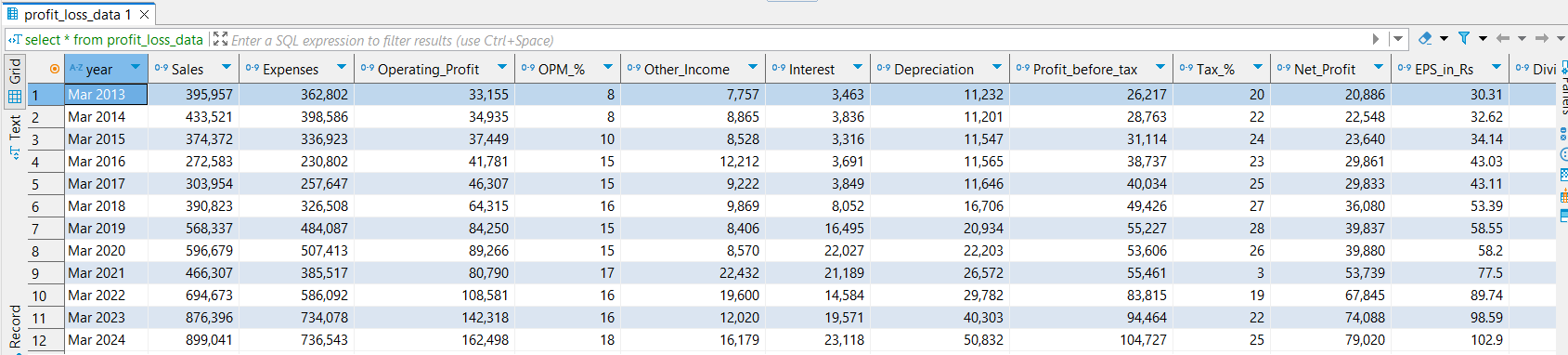


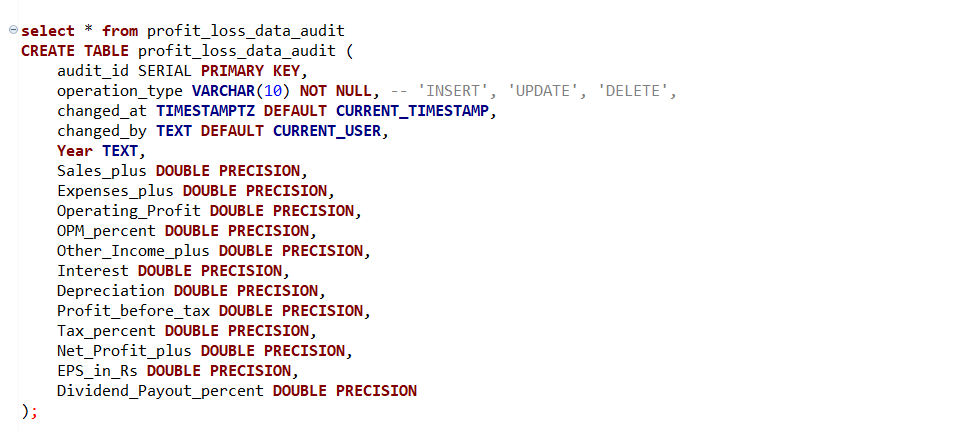
**Try to scrap-and-store your pipeline and fetch data into your postgres connection database.**



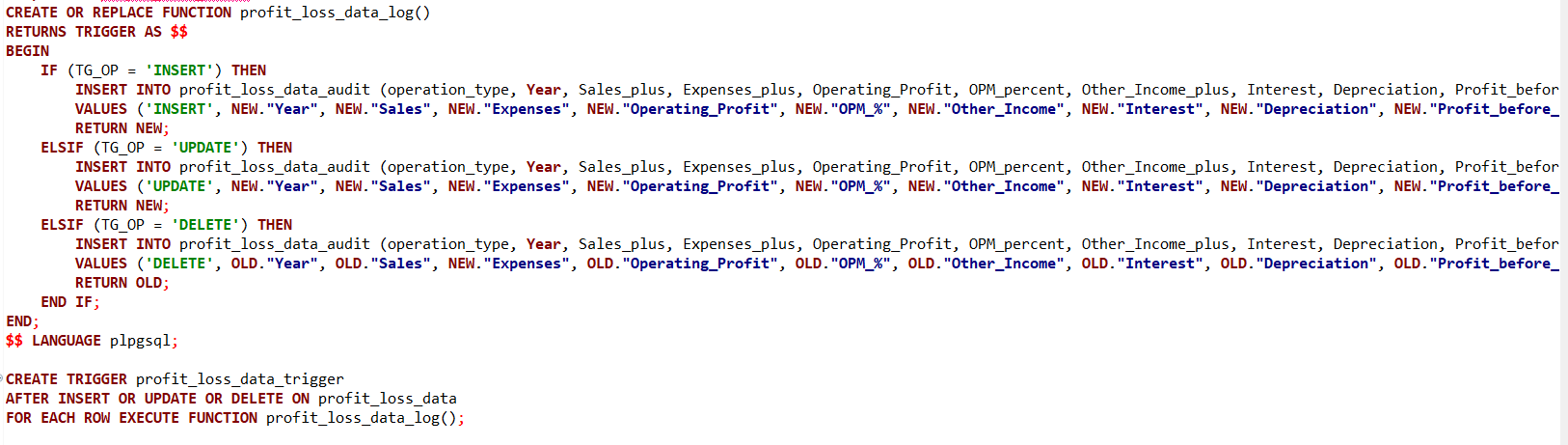




* **We can see the scrapped data into our postgres database in ‘profit\_loss\_data’ table.**
* ****
* **After that I have created audit-table with same schema just added operations such as audit\_id, changes\_at(timestamp), operation\_type.**

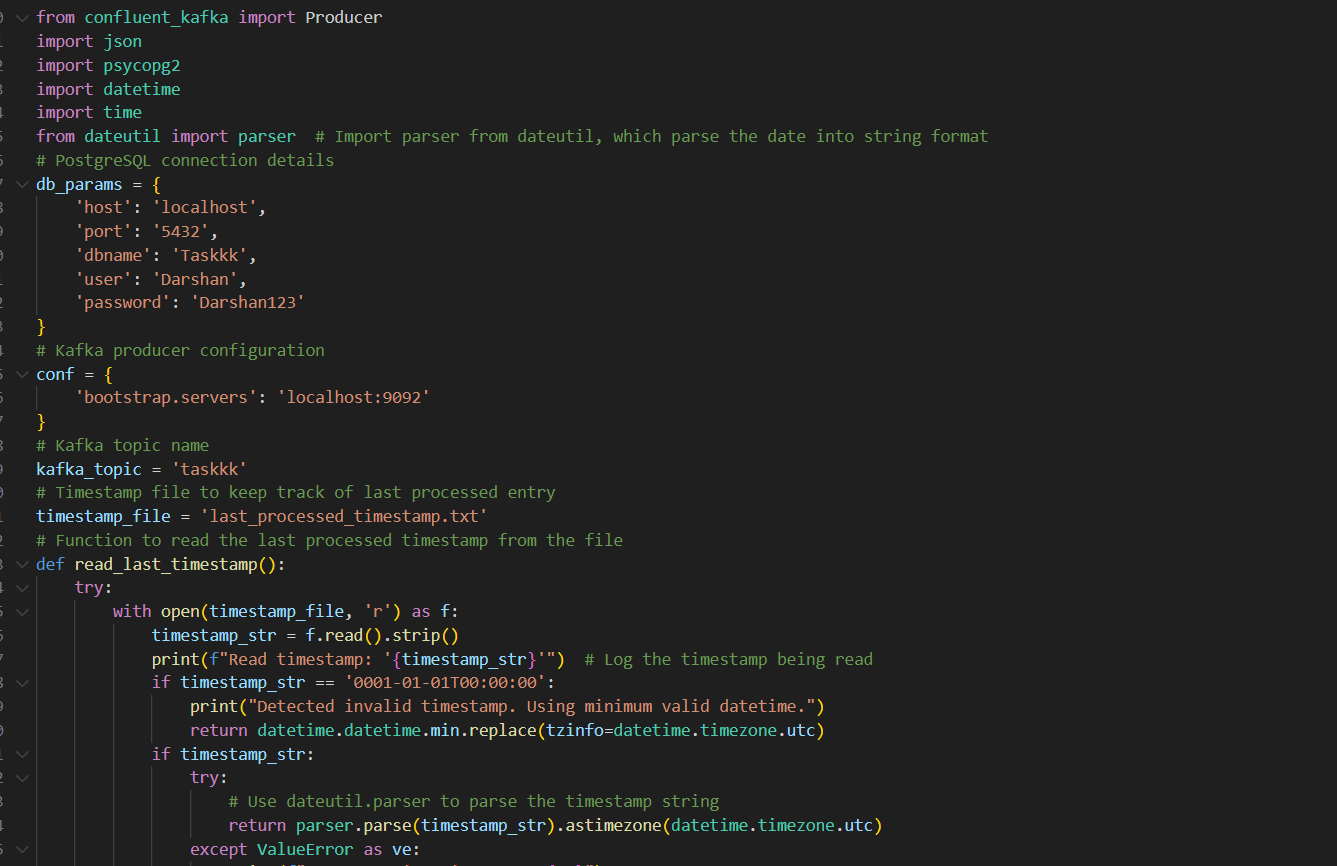
****

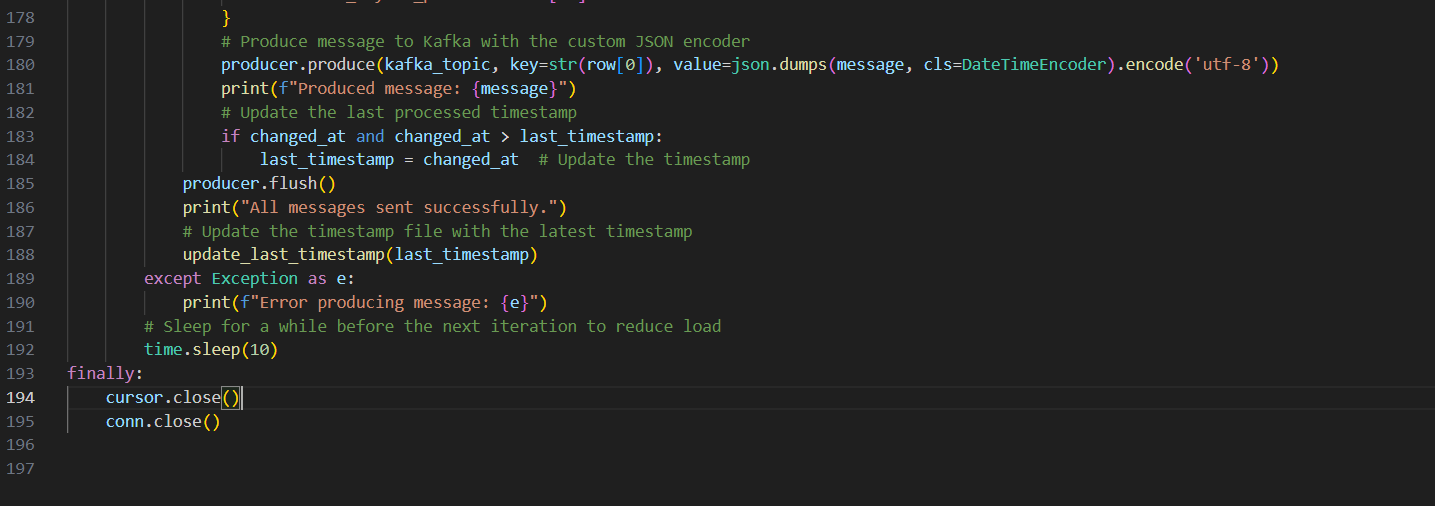
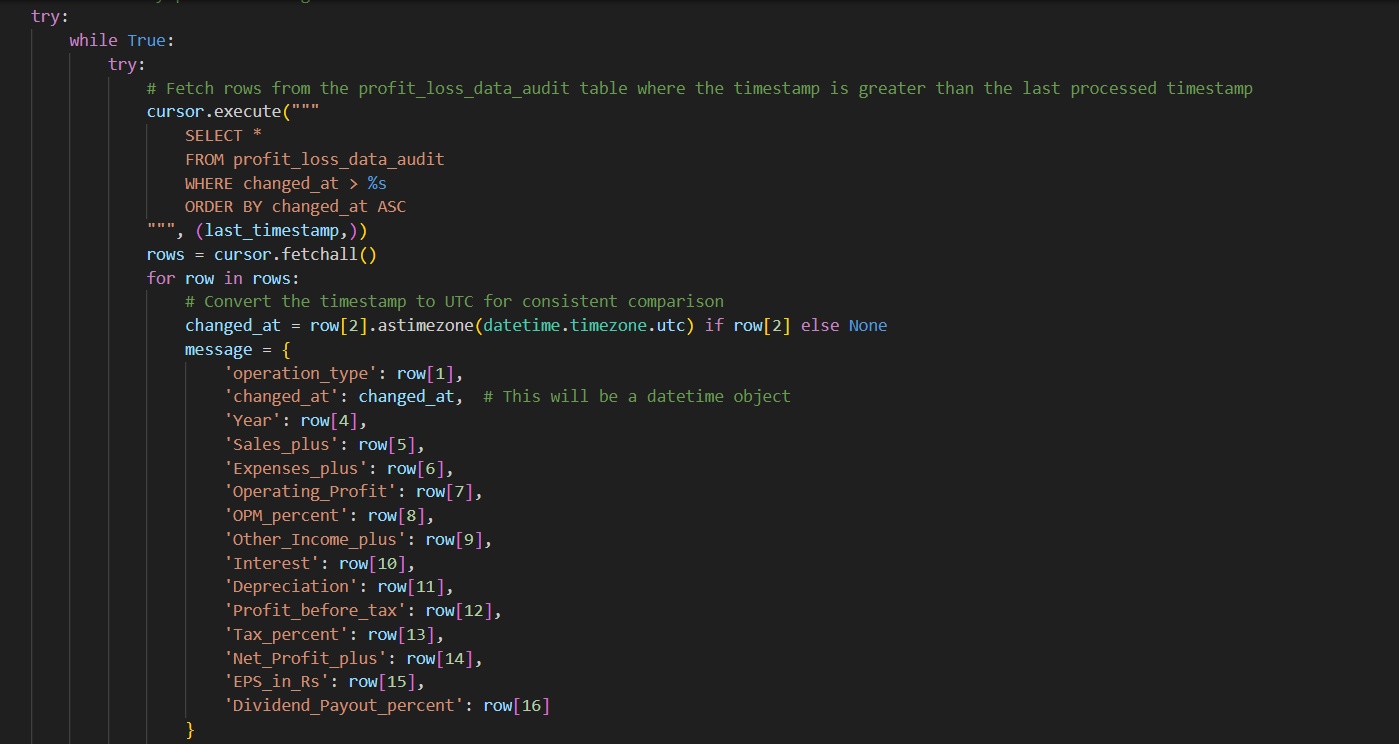
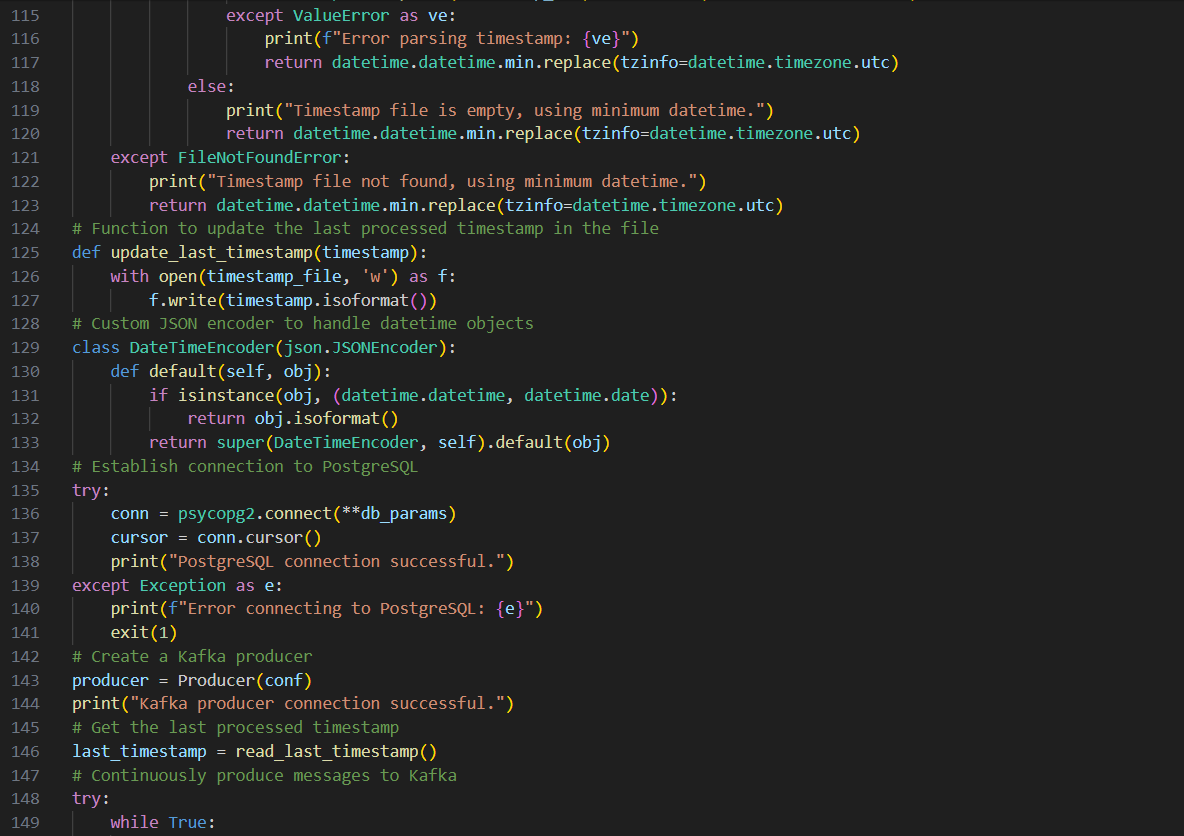
* **Then created a audit-log to monitor the operations such as insert, update, delete happens on main table and created trigger to identify that operations.**

****

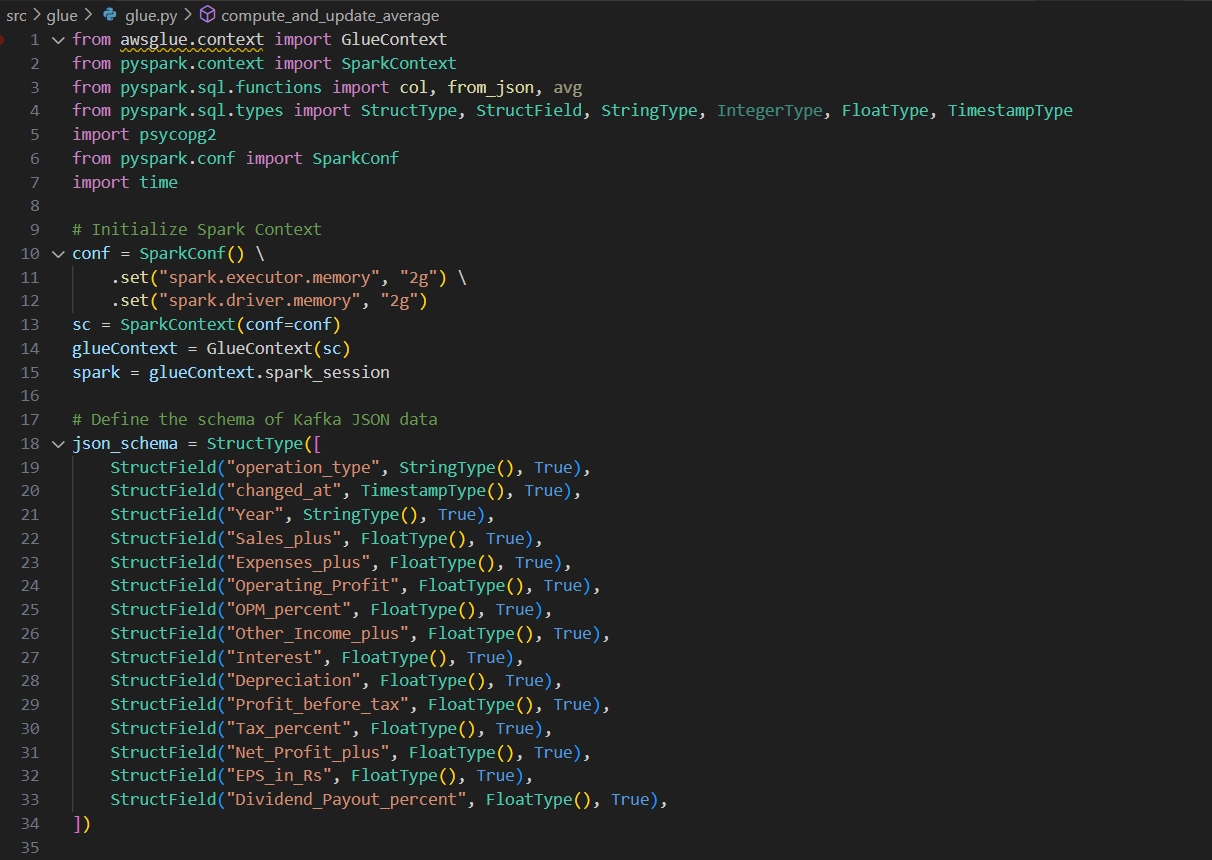
* **Then made a producer file, where created a timestamp file, in which made a function which checks the rows entry greater than previous timestamp.**

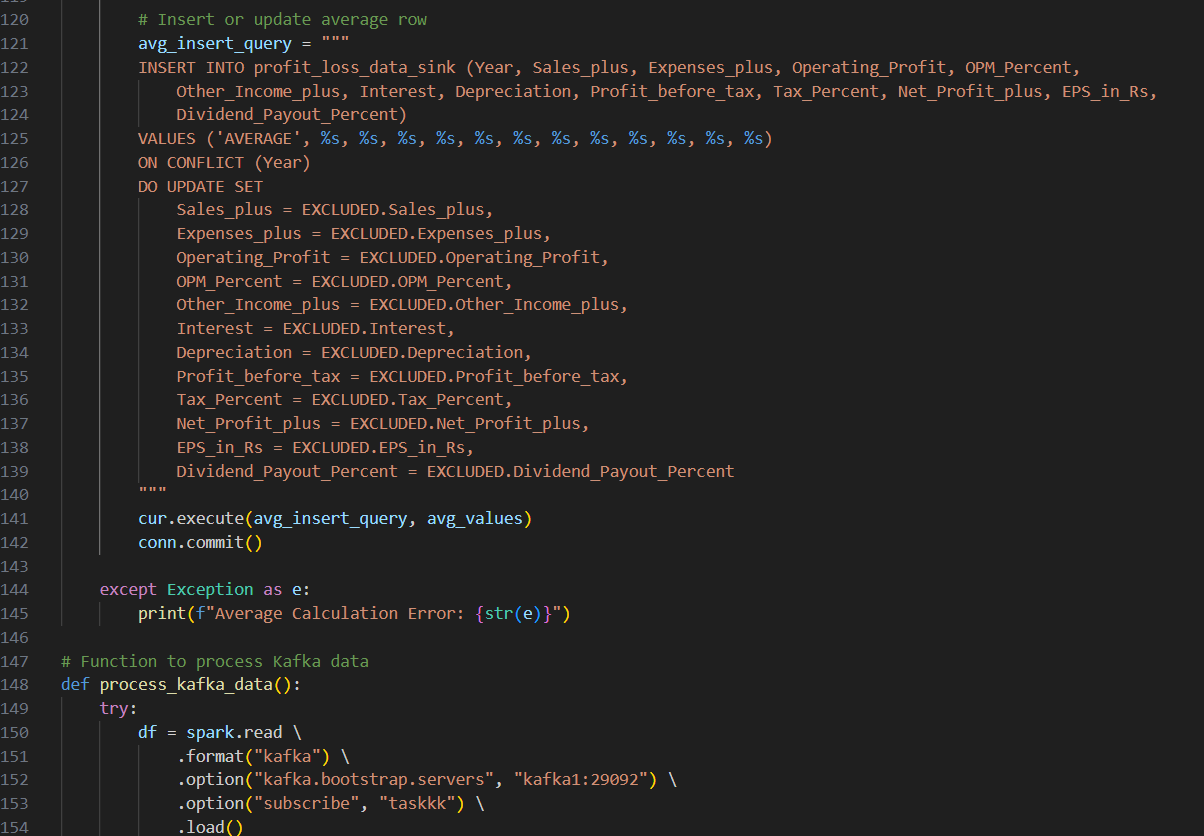
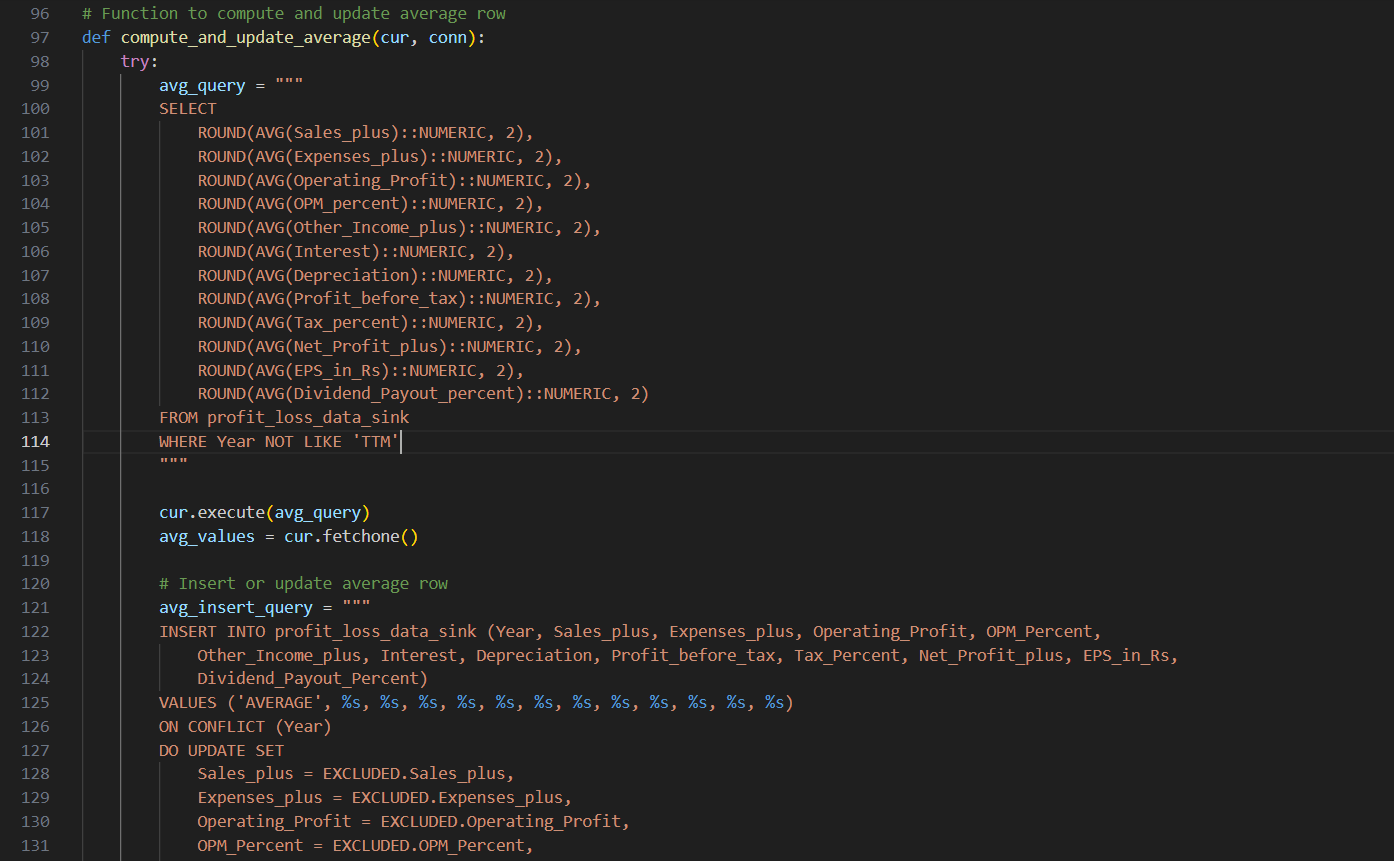
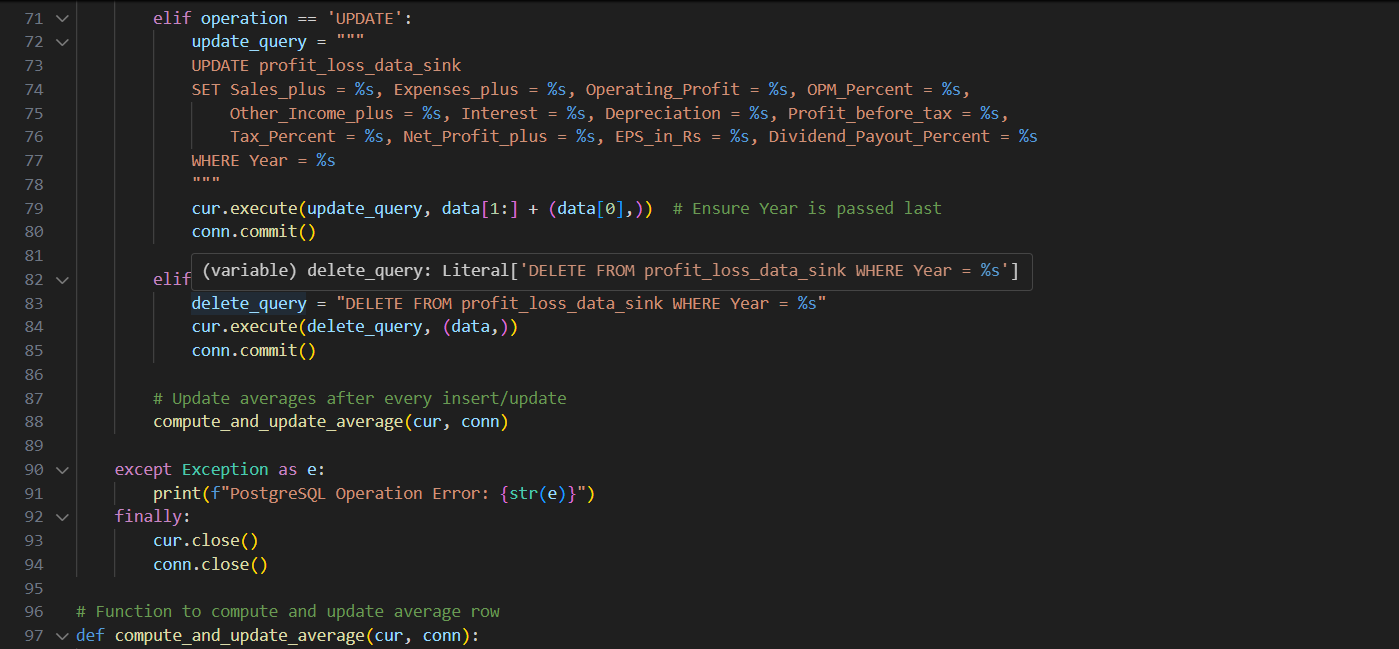
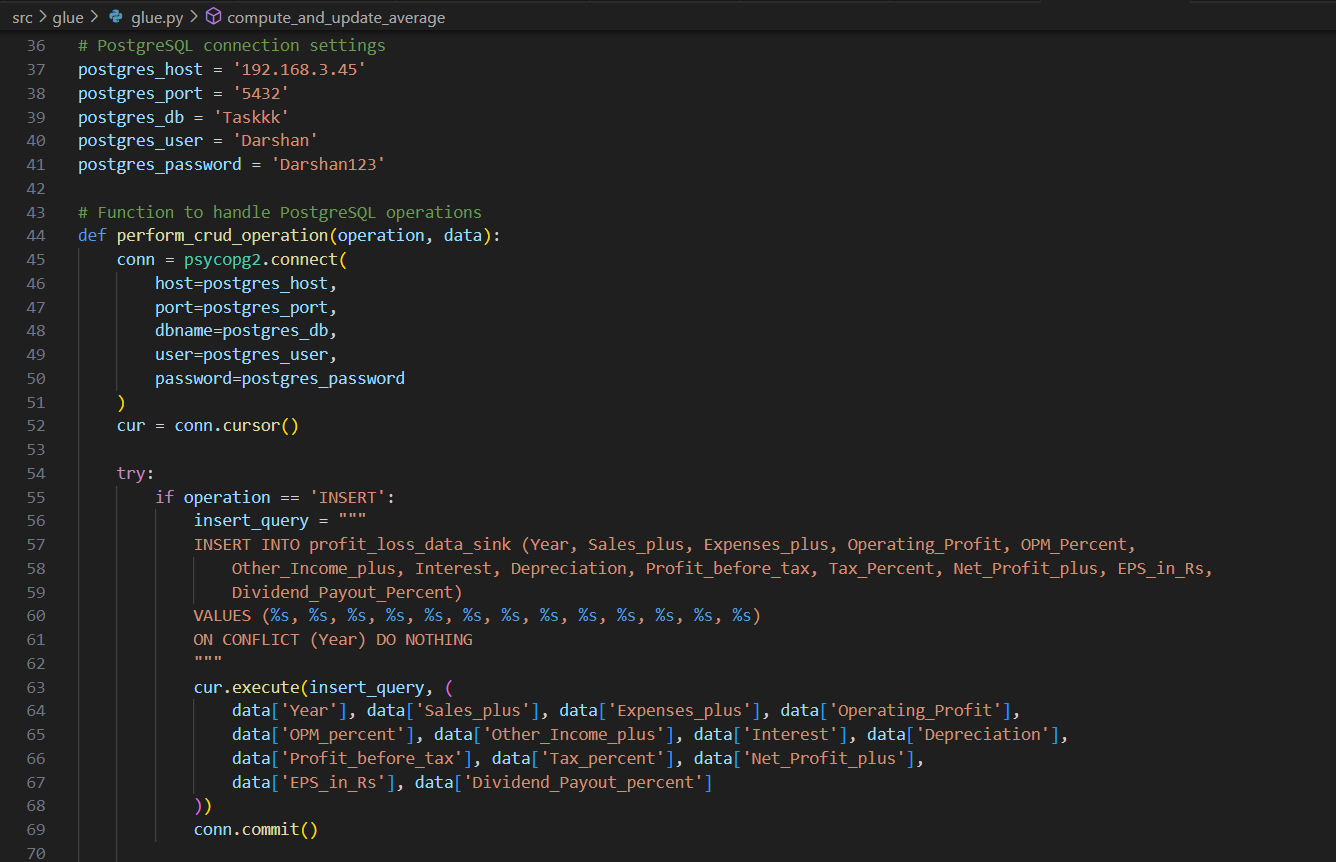
**Produer\_script:**

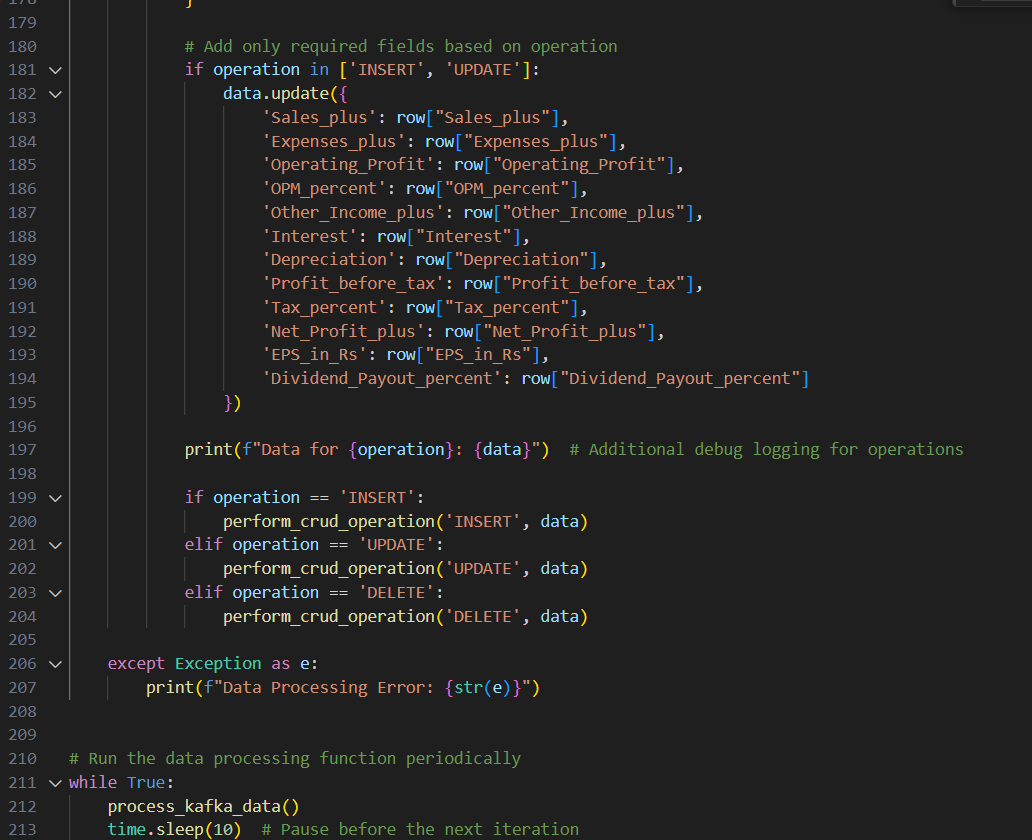
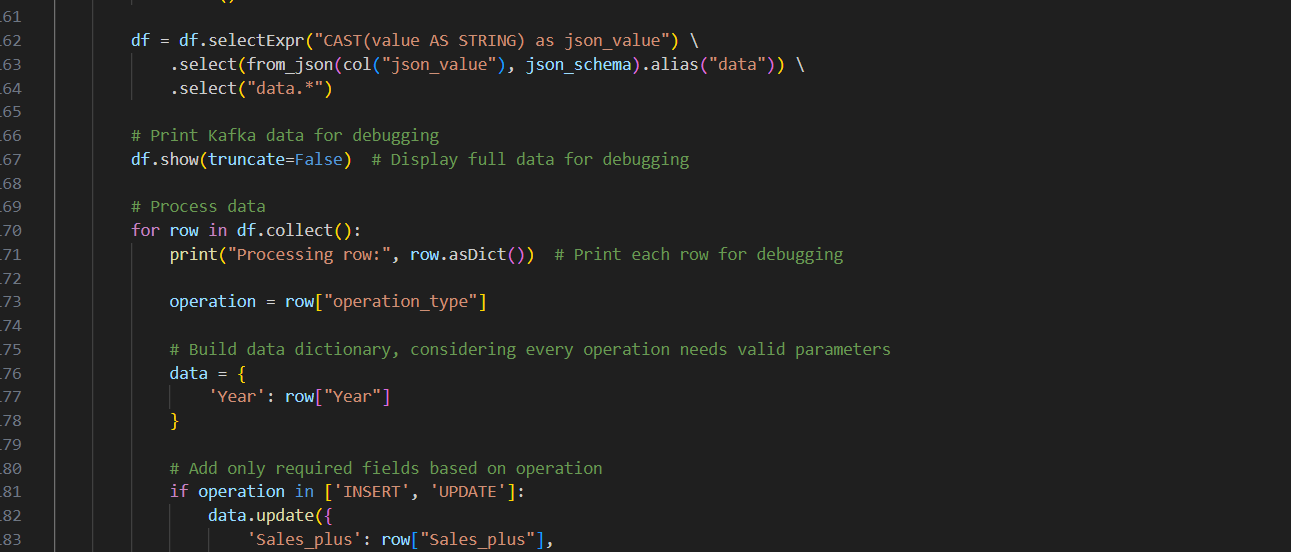
****

****

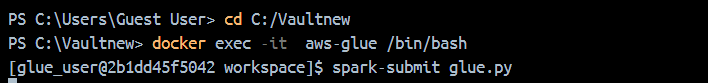
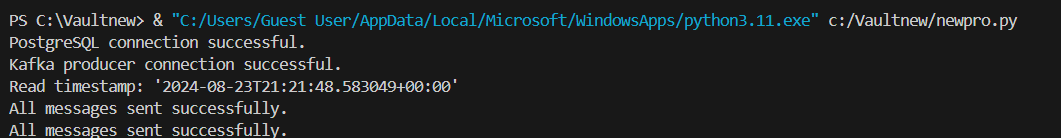
* **In this file we are taking the operation of audit-log from audit-table (profit\_loss\_data).**
* **File will be run continuously as we will make any changes it we reflect in topic (‘task7’).**
* **After that made a glue script for to make a sink table ( profit\_loss\_data\_sink ) and all data and changes made will be reflect there.**

**Glue\_script: **

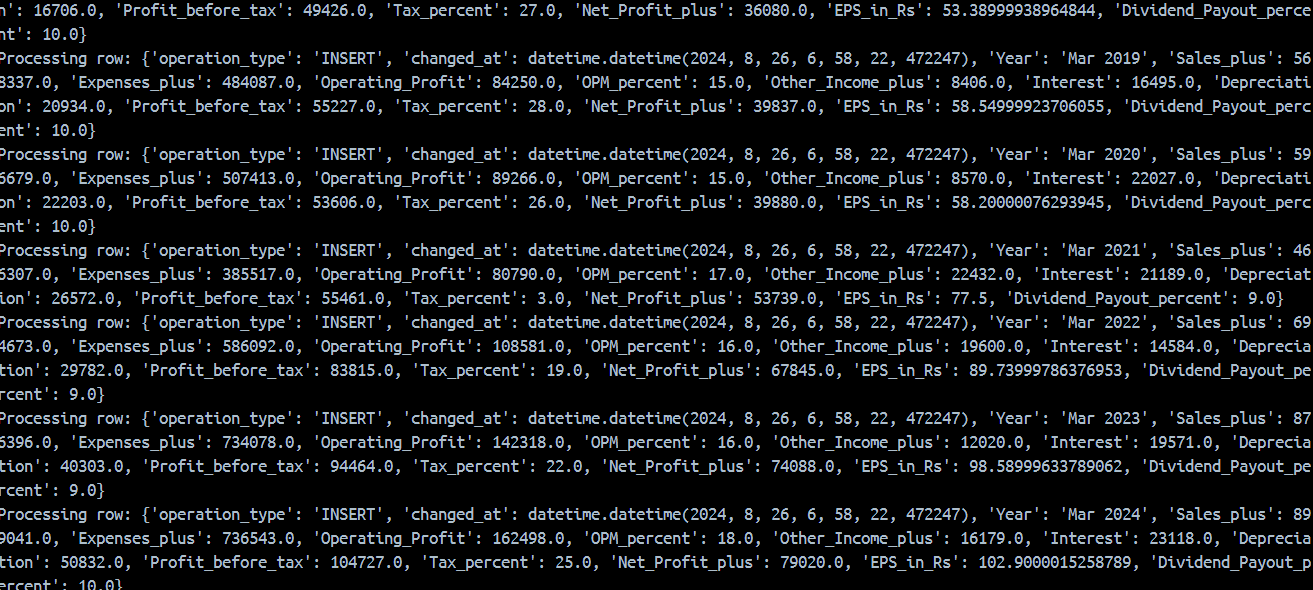
****

****

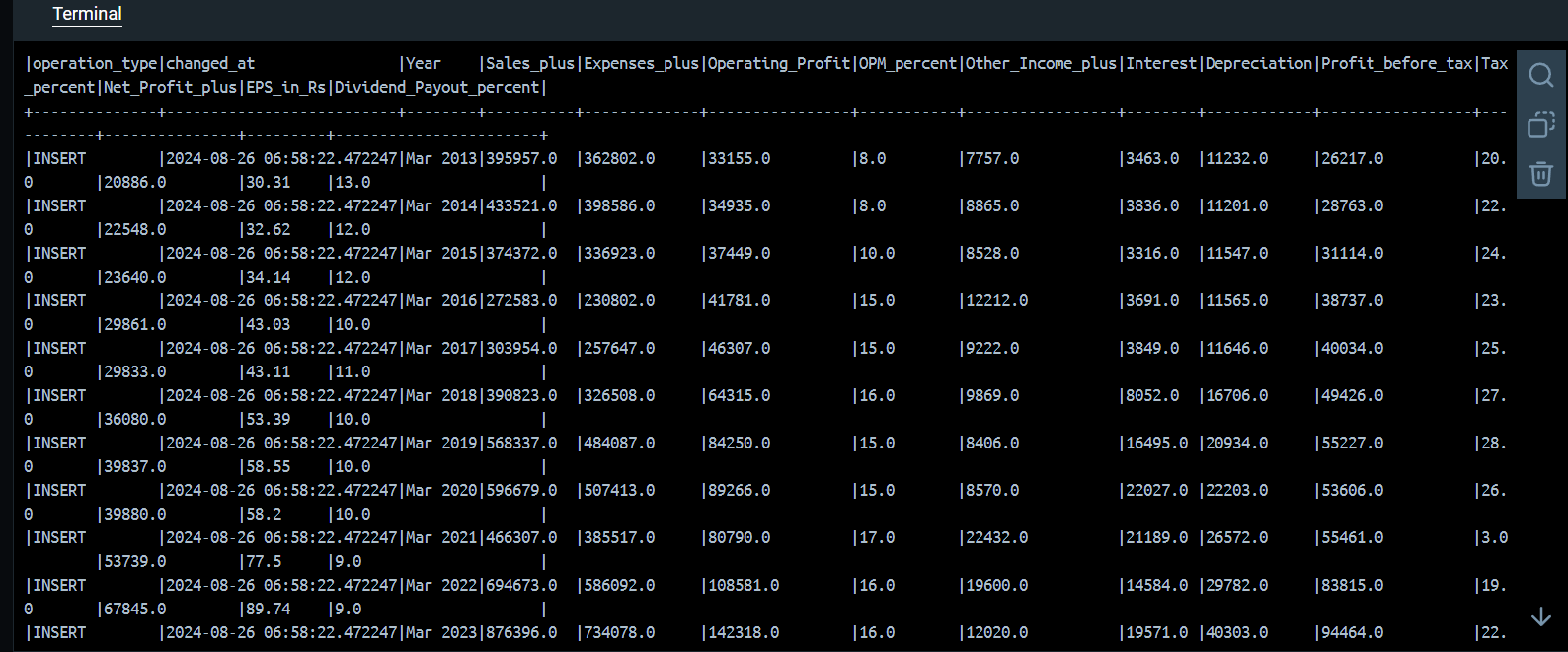
* **In this glue\_script, gave a tcp connection of IPv4 as postgres\_host and all configuration of database.**
* **After that update a script according the insert, update, delete operation and to execute average operation made a function regading that.**
* **Futher all the data is saved into topic ‘task7’.**

****

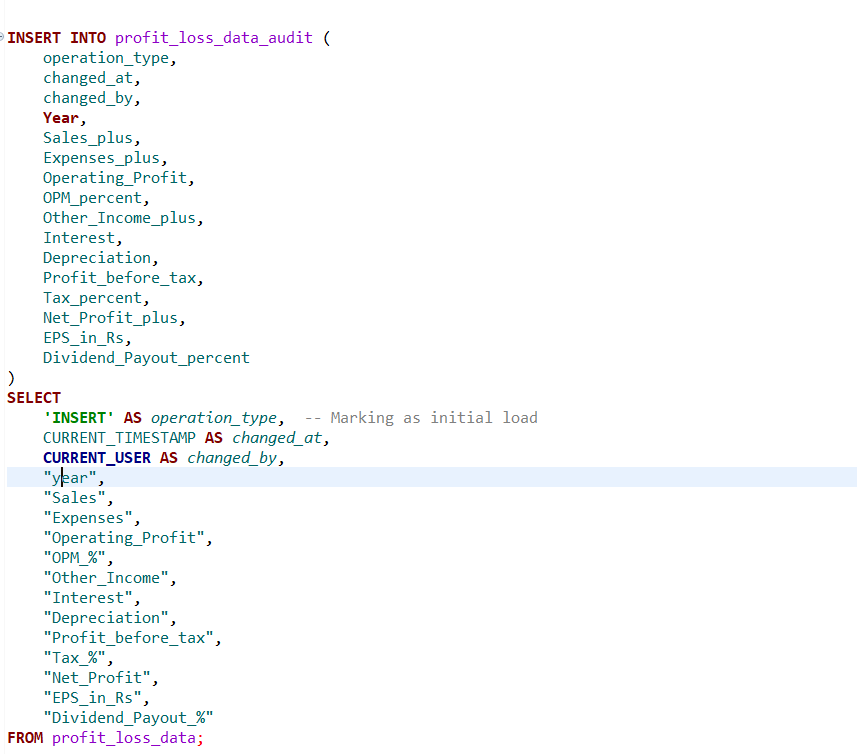
**Now both producer and glue script are running and set at time.sleep(10) running in while loop like it will continuously take the changes nd directly send to the target (sink) table.**

****

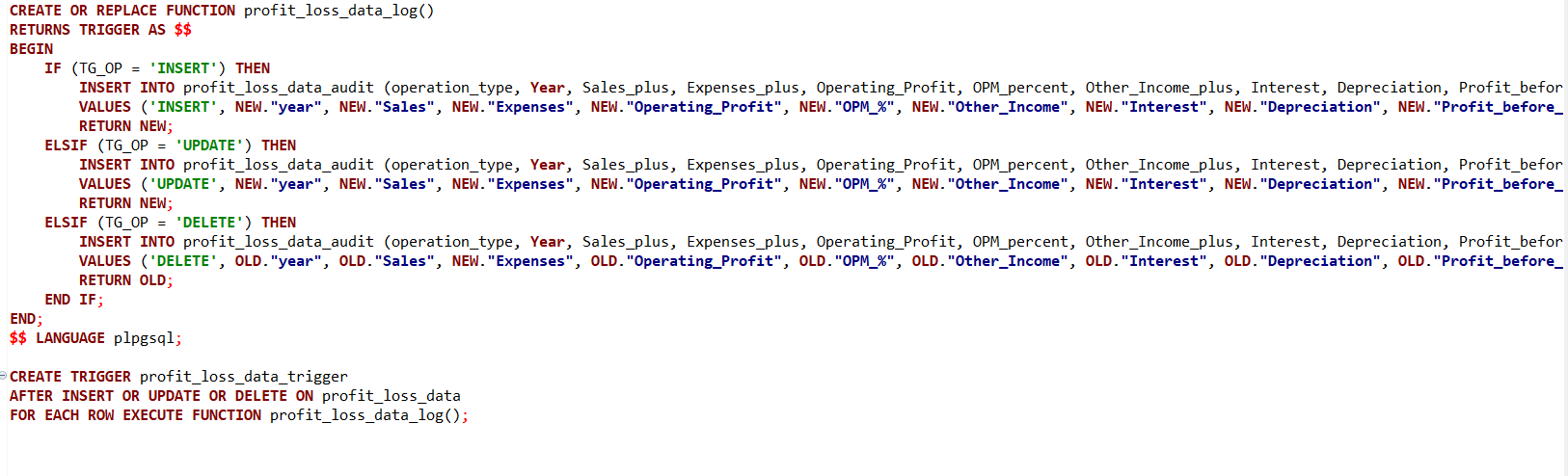
**These kind of logs we can see into glue file as it is processing data into sink table.**

****

**First I have put the previous data into audit table.**

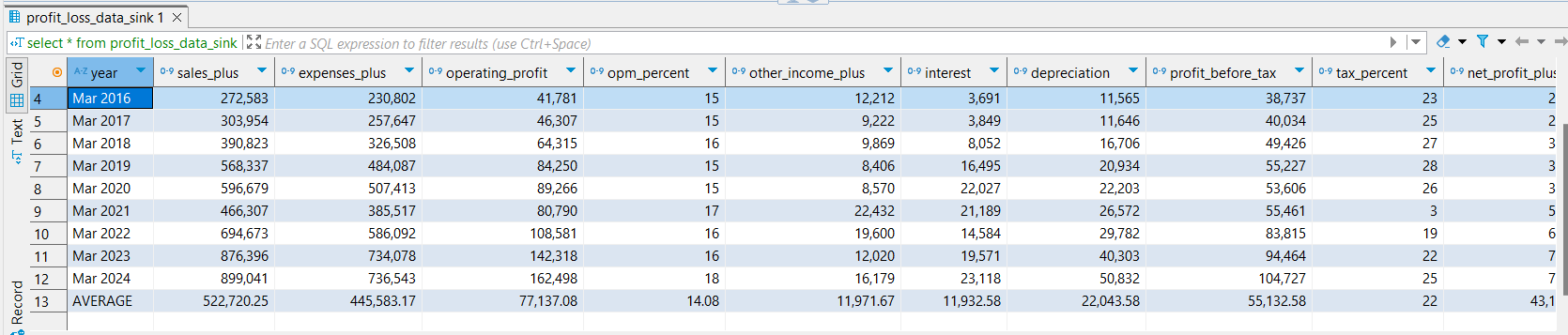
****

**Created a function and trigger for recognise the operations;**

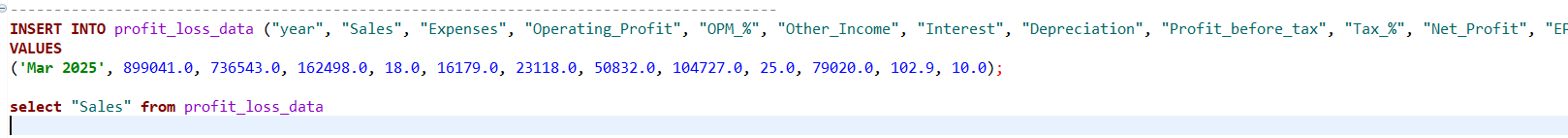
****

**After running a glue script:**

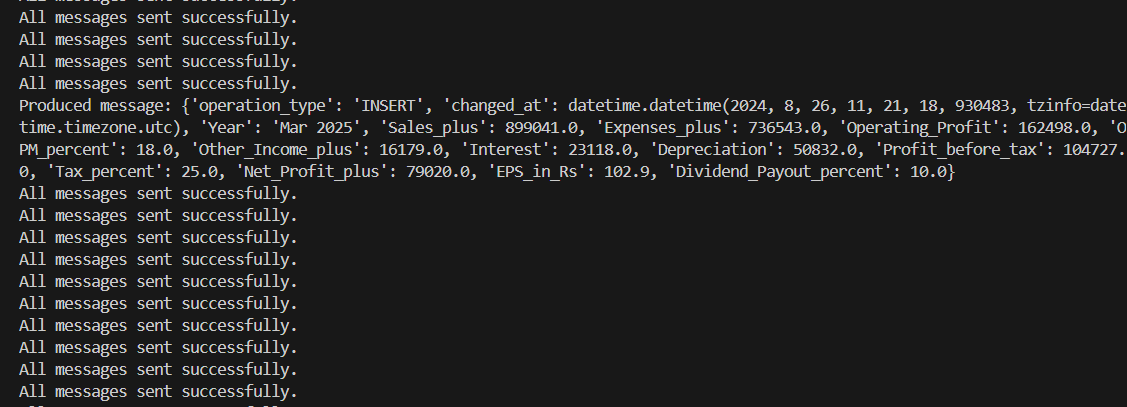
**Data is reflected into sink table**

****

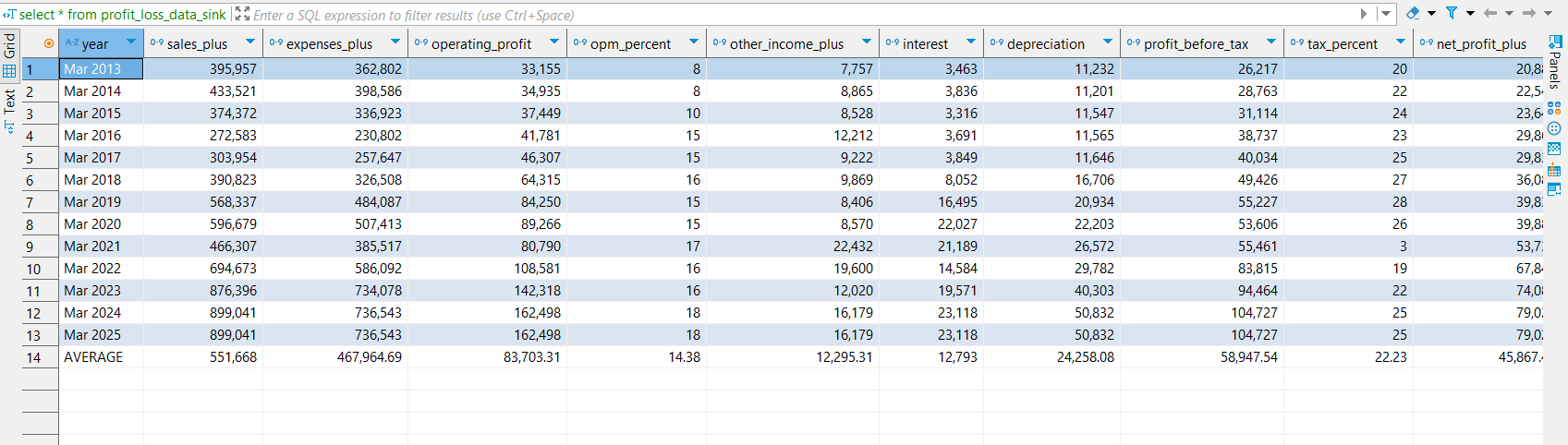
**Insert Operation:**

****

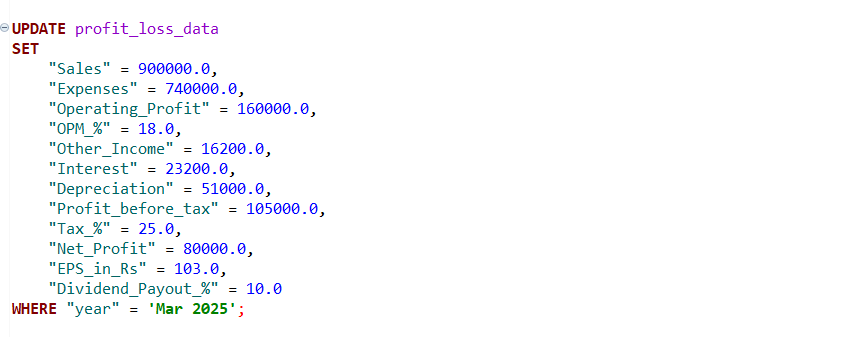
**Now, we can check the topic:**

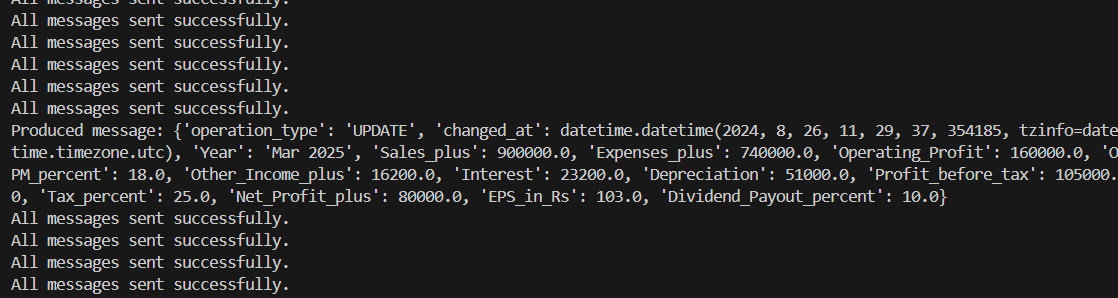
****

**Now, will check into sink table:**

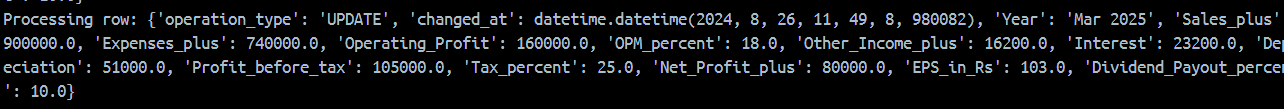
****

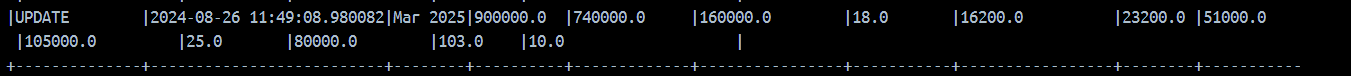
**Update Operation:**

****

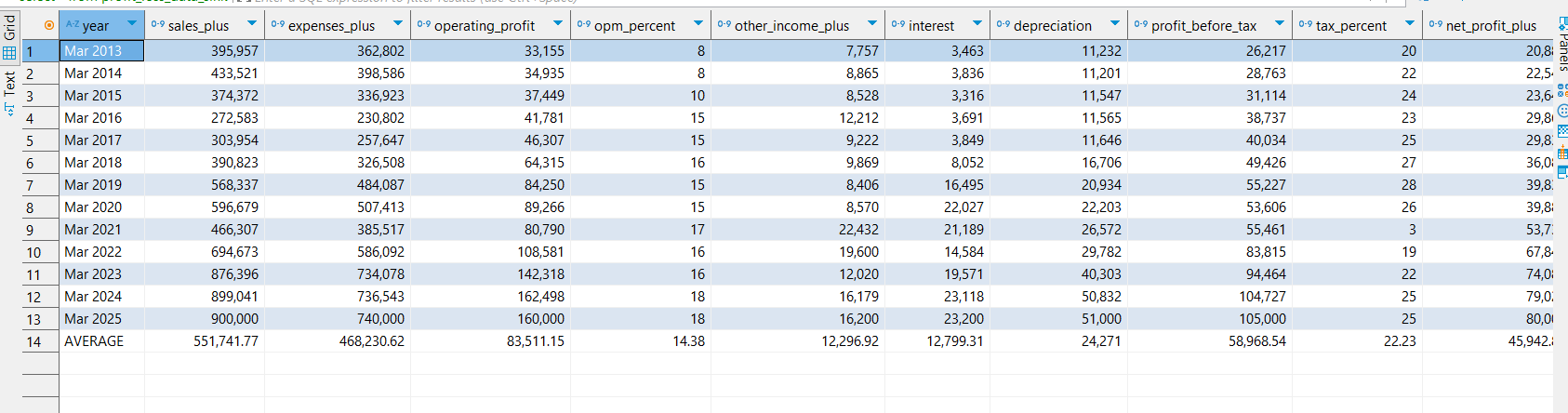
****

**As producer file is on time sleep of 10sec so it will take data into topic as operation proceeds.**

****

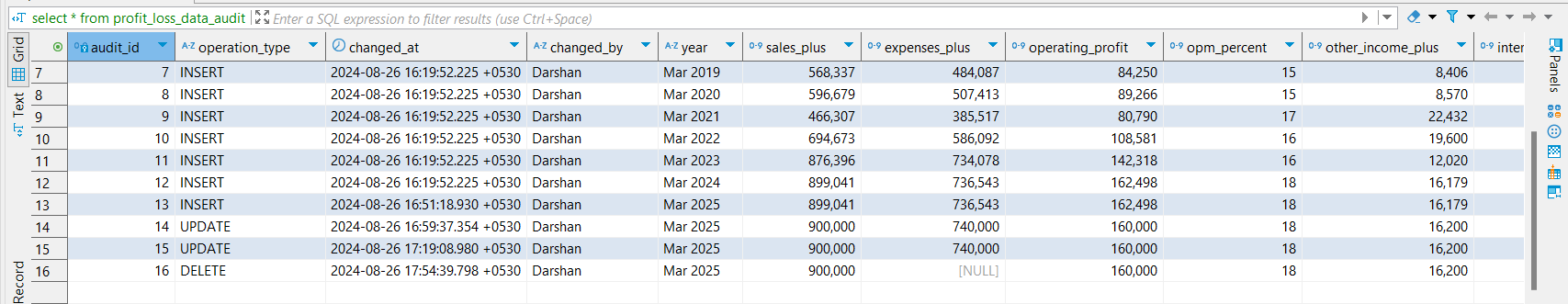
****

**Check the data in sink table:**

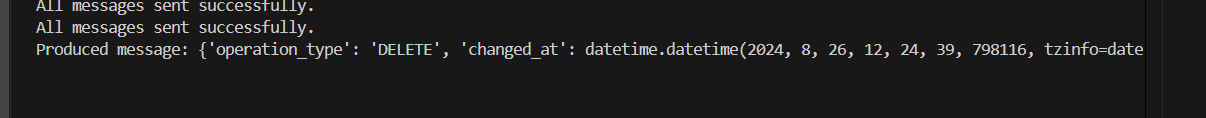
****

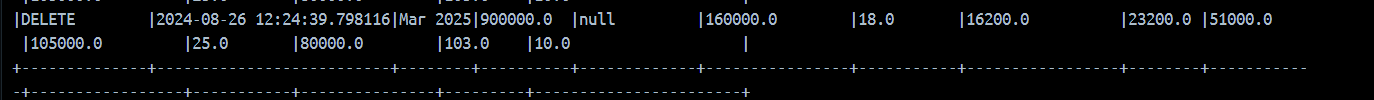
**Delete Operation:**

****

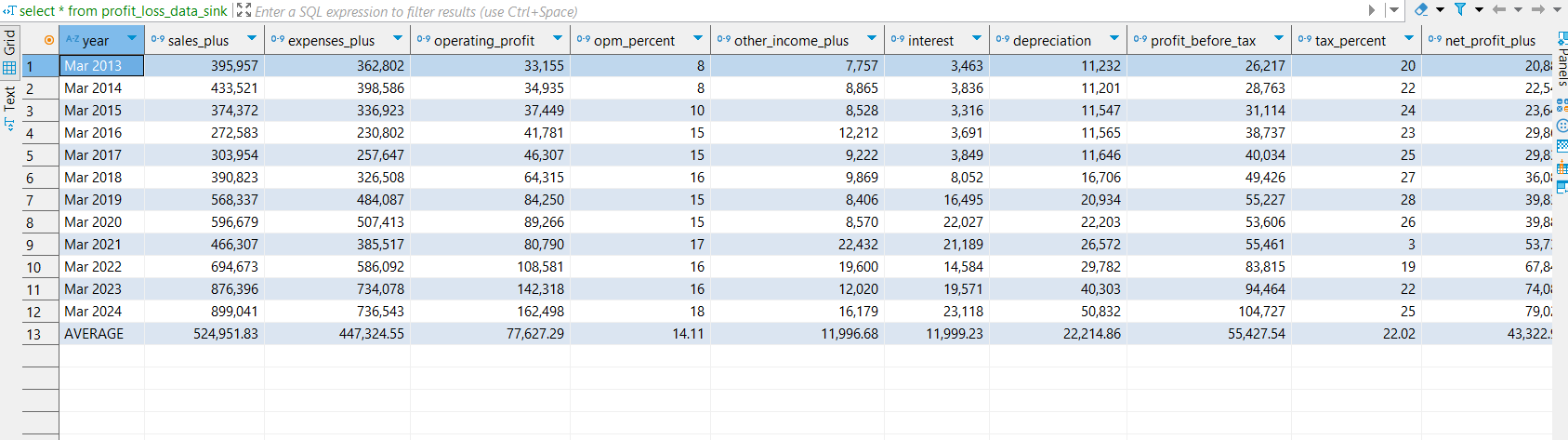
****

**You can see the Delete operations in audit table logs.**

****

****

**Now we can see the changes after delete the row where year in ‘Mar 2025’**

****

**-------------------------------------------------------Task 7 completed-------------------------------------------------------**

**Issues Got:**

1. **At the time of making dokcer-compose file while detaching aws-glue in this direccory issues of ‘psycopg2’ module not found got so, inside docker terminal , inside aws-glue shell run**

**“pip install psycopg2 –binary’.**

1. **Table transformation issues: Data types were at random , so makes them into order like**

**In ‘profit\_loss\_data’ table except ‘year’ column all columns were in double precision, so created audit table and sink table on same data type.**

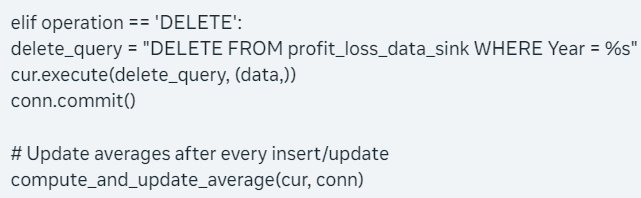
1. **More than one triggers were mistacally run while doing operations day by day**

**So main table has more than one gtables so it was appending every operations 3 times like insert operation shows insert two times same data.**

**I debug it while deleteing other triggers of same table, only one triggers left at the moment.**

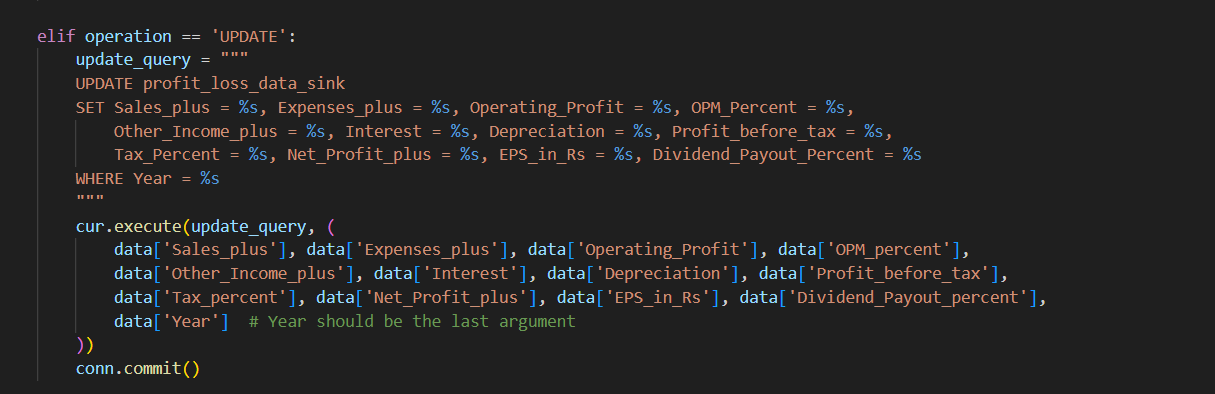
1. **While Updating data into ‘profit\_loss\_data’**

**Error: Unhashed – “slice’ occur , so I debufg that into glue script as**

**Before: **

**After:**

**Instead of slicing the data dictionary, we construct the parameter list directly using dictionary keys in the cur.execute() calls.**

****

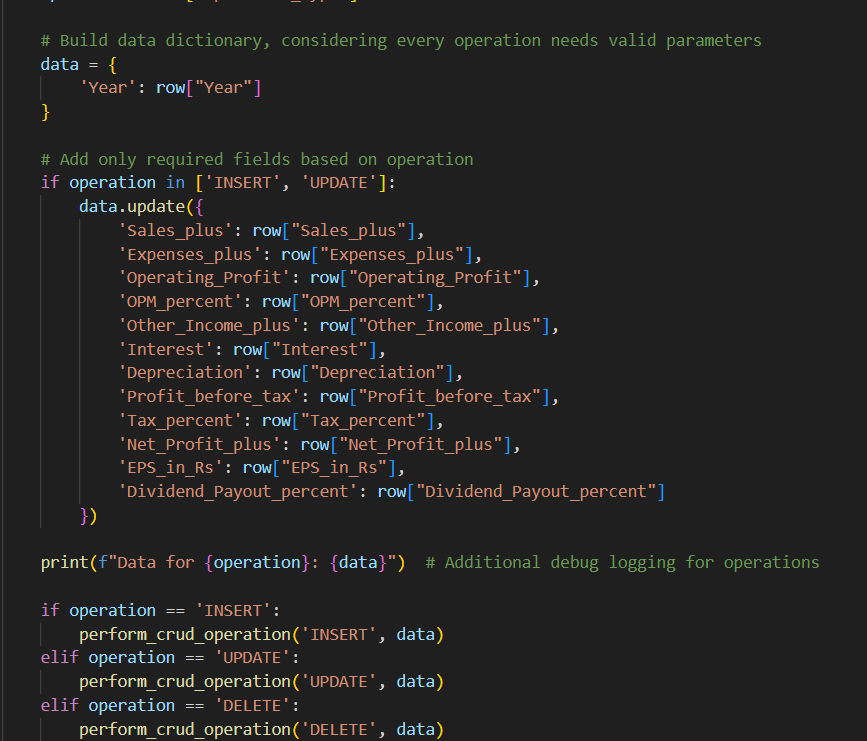
1. **While deleting any row, I was getting error as**

**Error: string indices must be integers**

**So, I debug it by using**

**I removed the unnecessary df.show(truncate=False) line in the process\_kafka\_data function, as it's not needed for the data processing pipeline.**

**Improved error handling: I added more specific error messages in the except blocks to help with debugging.**

****