**Concourse and Vault**



**Concourse**

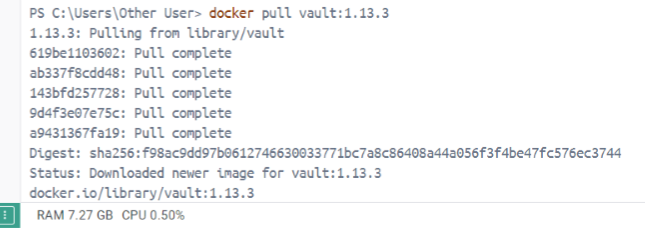
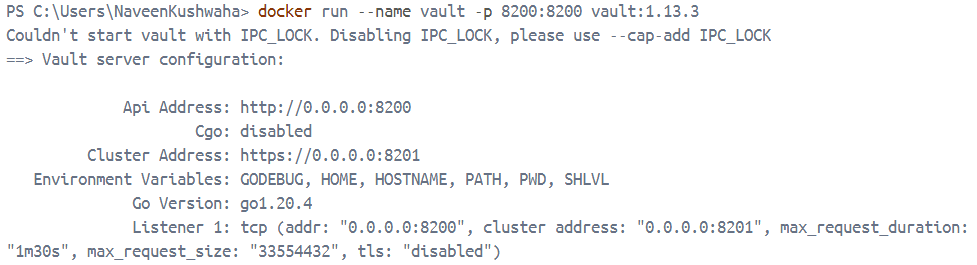
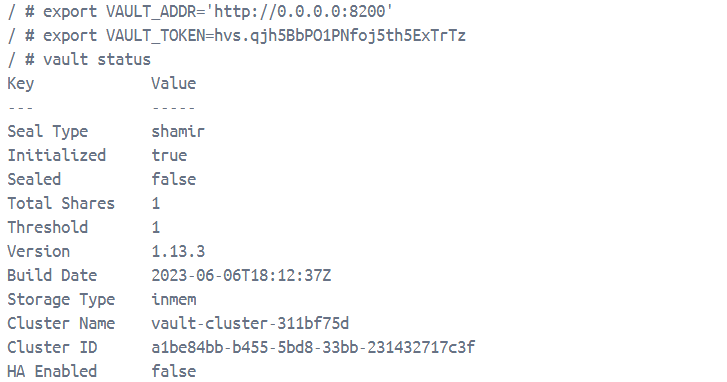
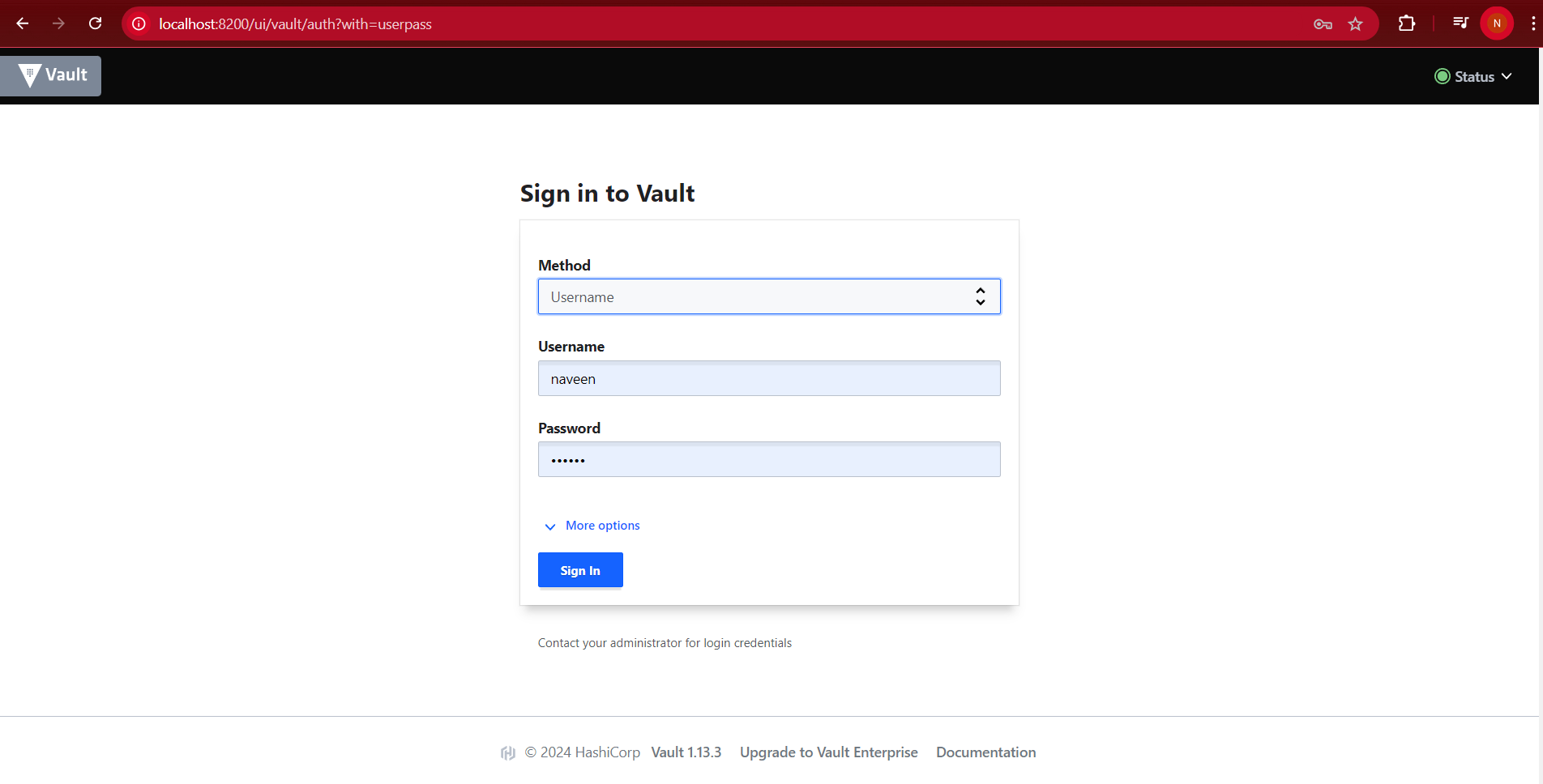
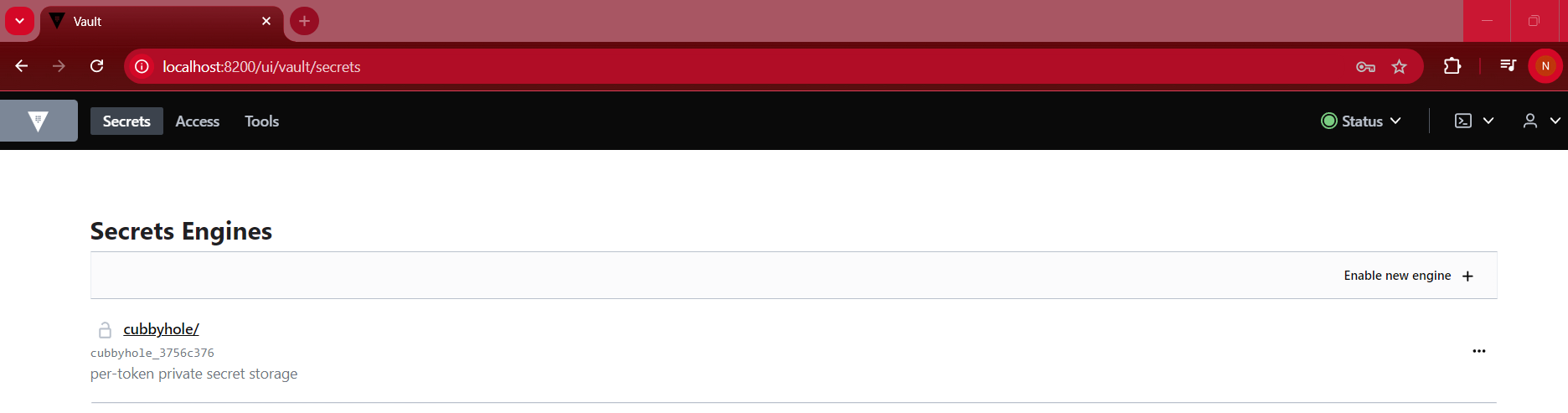
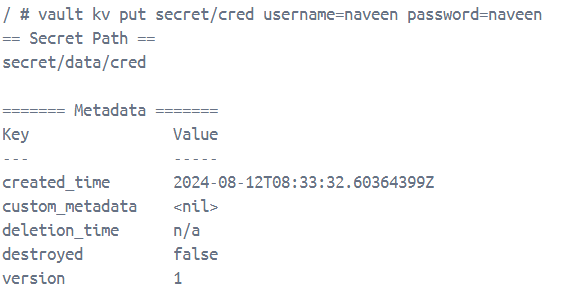
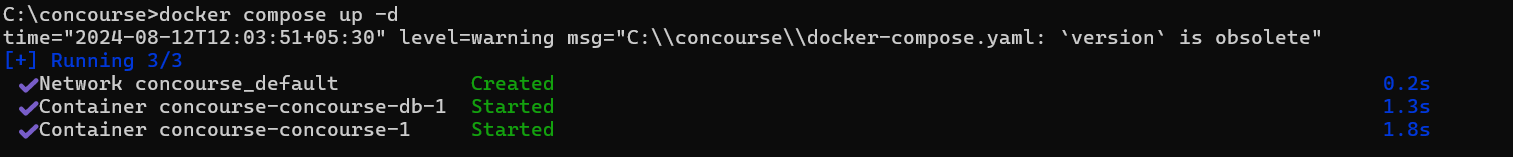
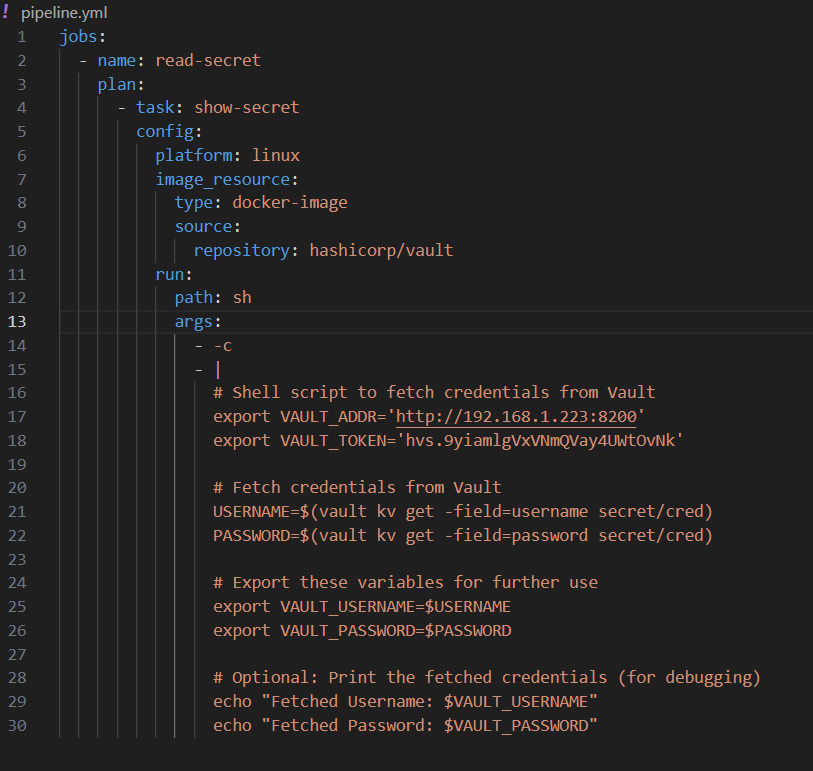
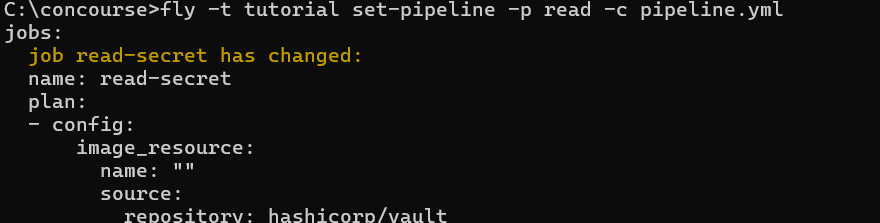
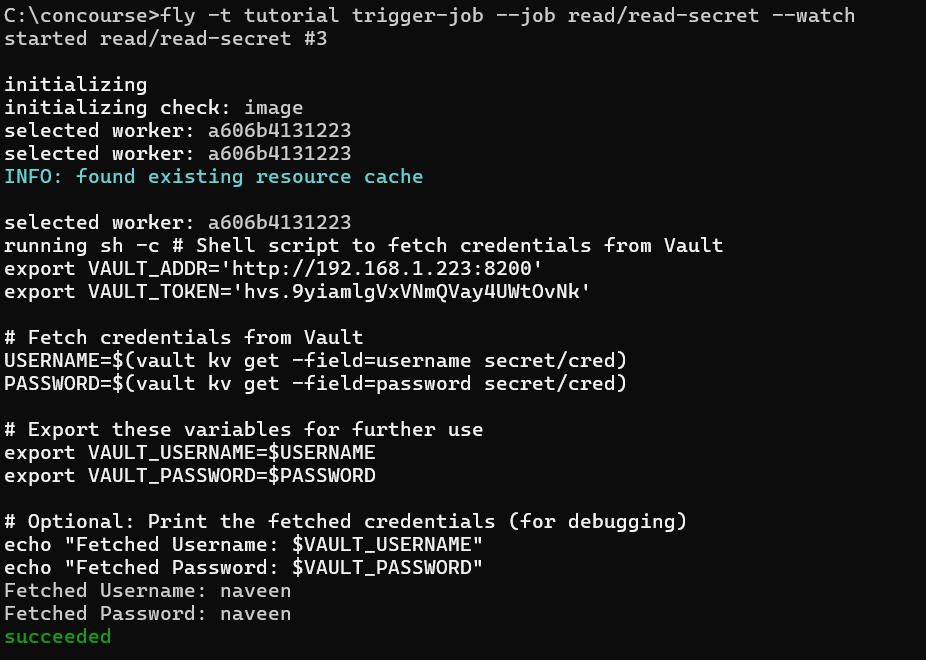
* **Overview of Concourse:**
  + Concourse is a free and Open-Source software project that relies on the contributions of sponsors and volunteers from around the world.  
    When working on a sizable project, having a pipeline to reliably test, deploy, and publish the product is crucial for rapid iteration.
  + Concourse CI/CD is simple enough to fully grok and easy to manage as your project grows; in both the complexity of the product and the size of your team.
* **Key Features of Concourse C:**
  + Provides a real-time view of the health and status of code moving through pipelines. Users can drill down for more information or troubleshoot errors.
  + Users can configure pipelines as code and visualize them.
  + Concourse uses containers to scale automation pipelines and ensure reproducibility.

**Vault**

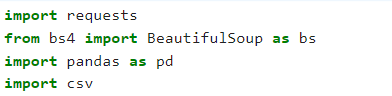
* **Overview of vault:**
  + Vault is a tool designed by HashiCorp for securely managing secrets and sensitive data. It is widely used to store and control access to tokens, passwords, certificates, API keys, and other secrets. Vault provides a unified interface to access secrets across various environments and supports dynamic secrets generation, encryption-as-a-service, and identity-based access control.
* **Key Concepts and Features of Vault:**
  + Secrets Management
    - Static Secrets: These are pre-defined secrets like passwords, API keys, and certificates that Vault stores securely and provides on request.
    - Dynamic Secrets: Vault can generate secrets on-demand for services.
  + Encryption as a Service
    - Vault allows encryption and decryption of data without the need to manage encryption keys directly. This is useful for applications that need to encrypt sensitive data like credit card information.
  + Identity and Access Management
    - Authentication: Vault supports various authentication methods.  
      This allows users and machines to authenticate securely to Vault.
    - Authorization: Once authenticated, access to secrets and operations in Vault is controlled by policies.
  + High Availability (HA) and Disaster Recovery
    - Vault supports clustering for high availability.
    - Disaster recovery replication enables an additional Vault cluster to be kept in sync with the primary cluster, allowing quick recovery in case of catastrophic failure.

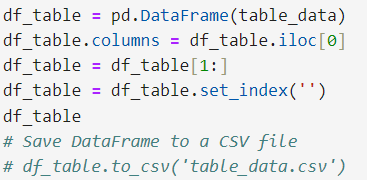
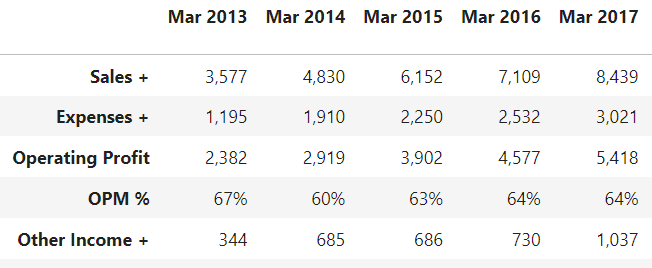
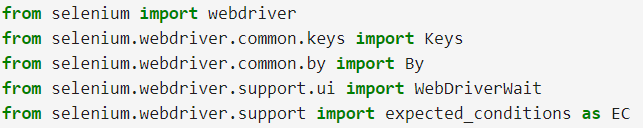
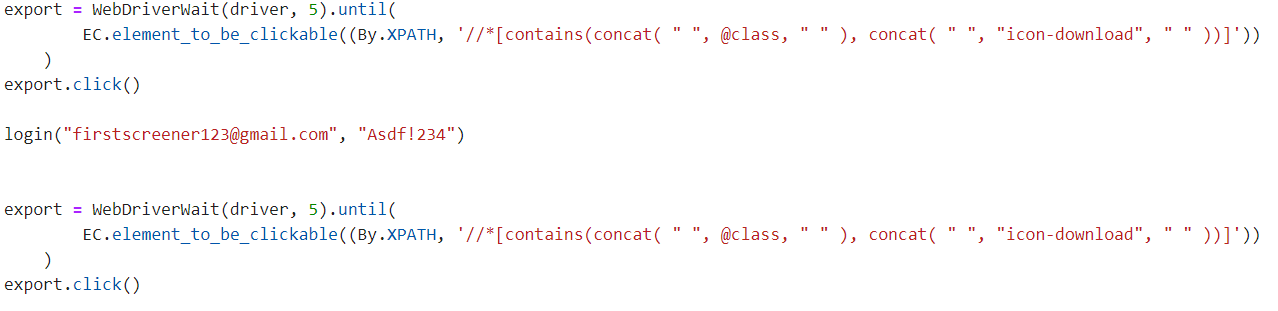
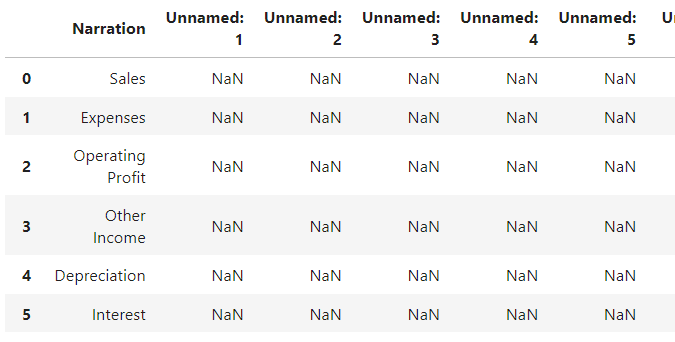
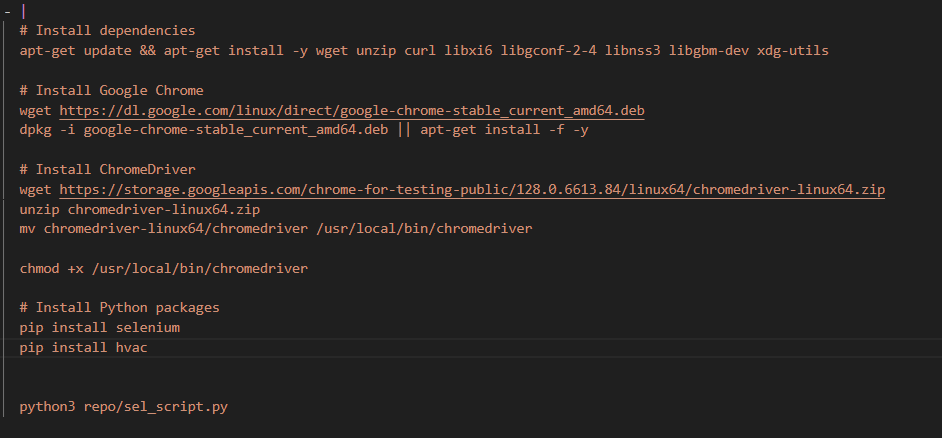
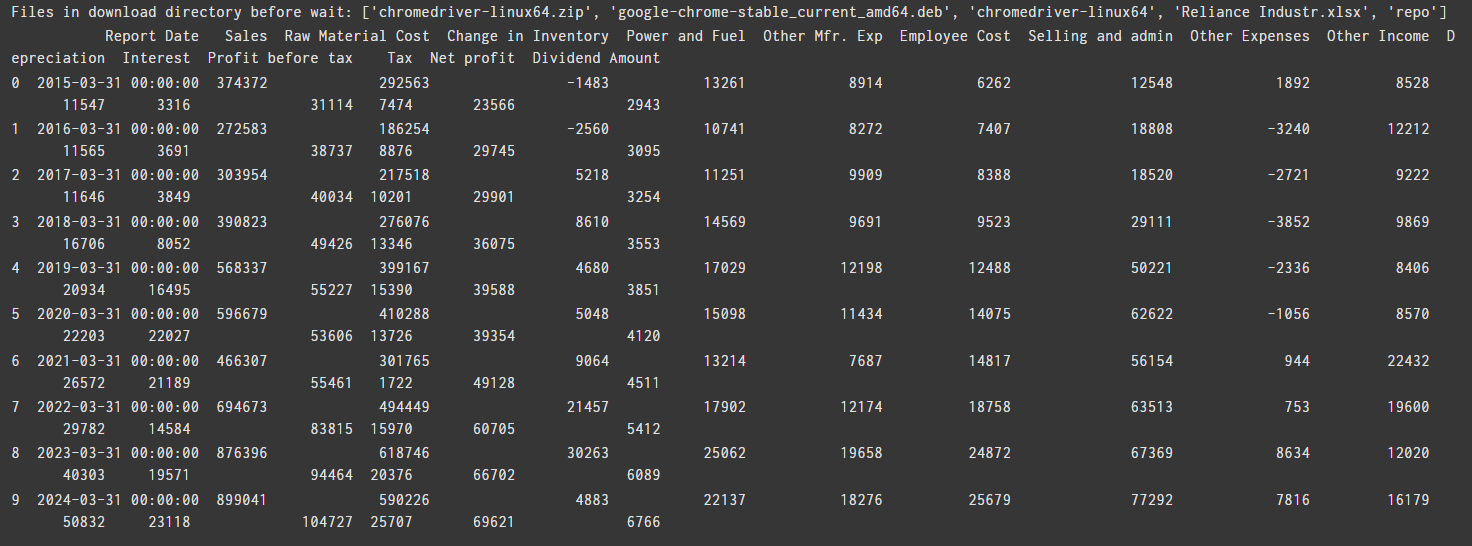
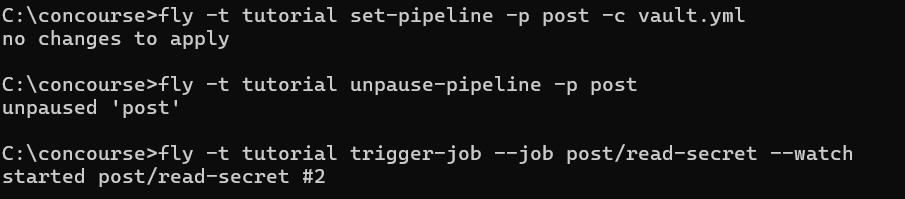
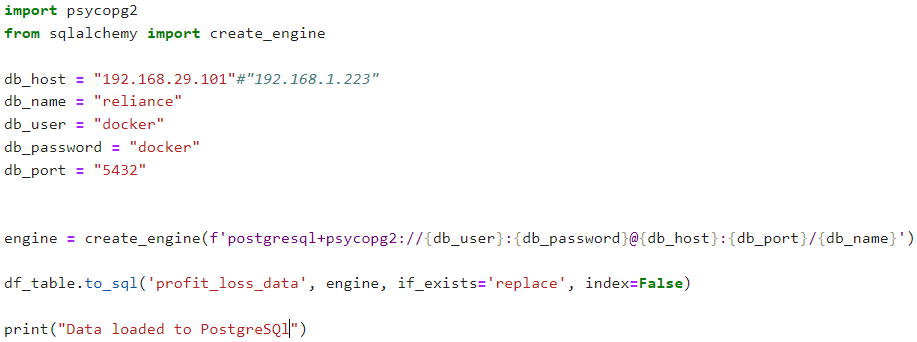
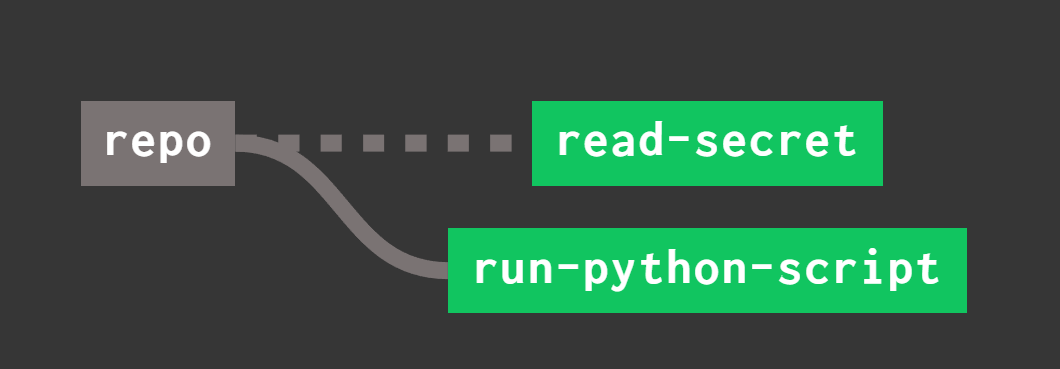
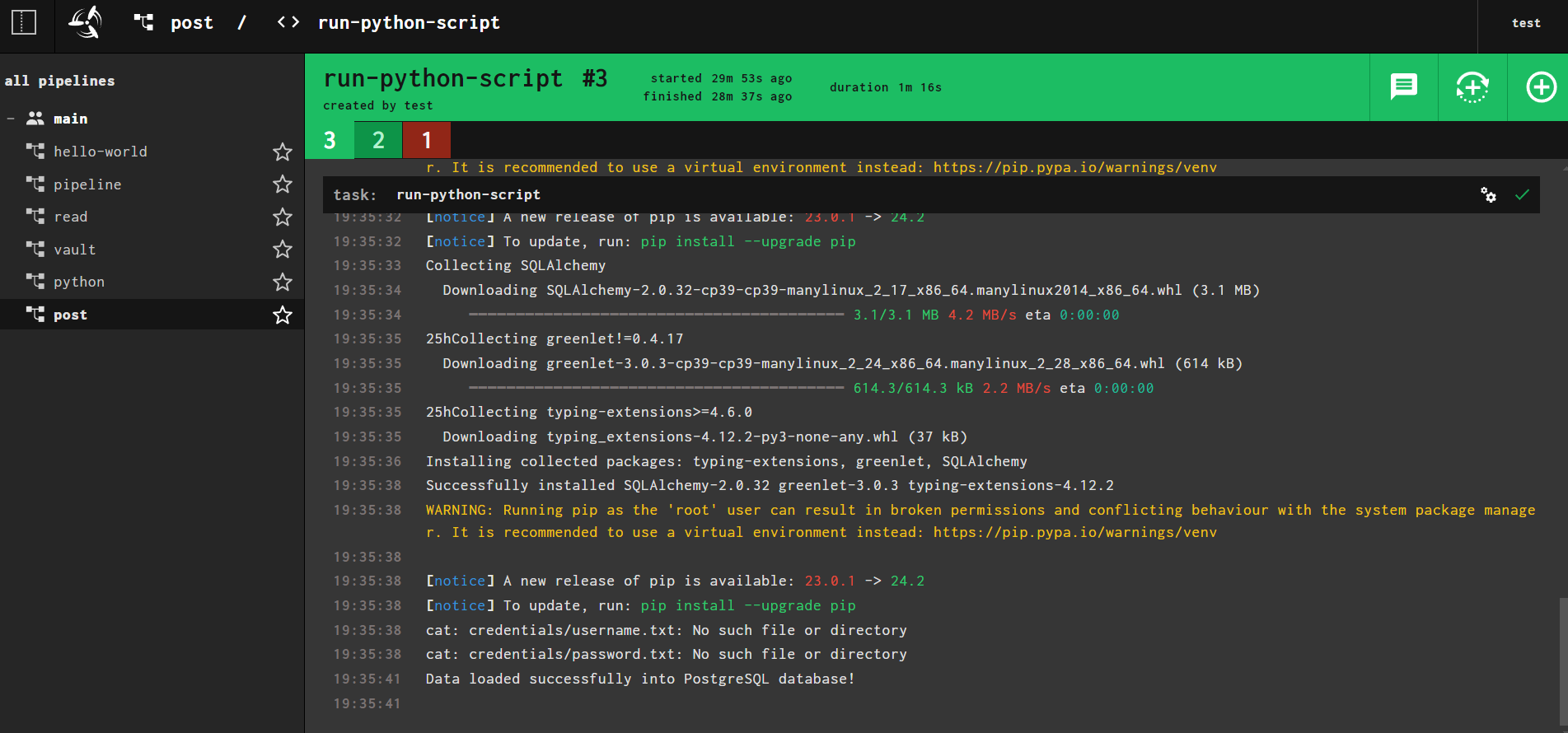
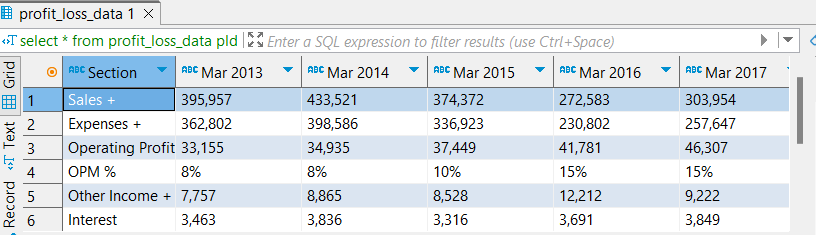
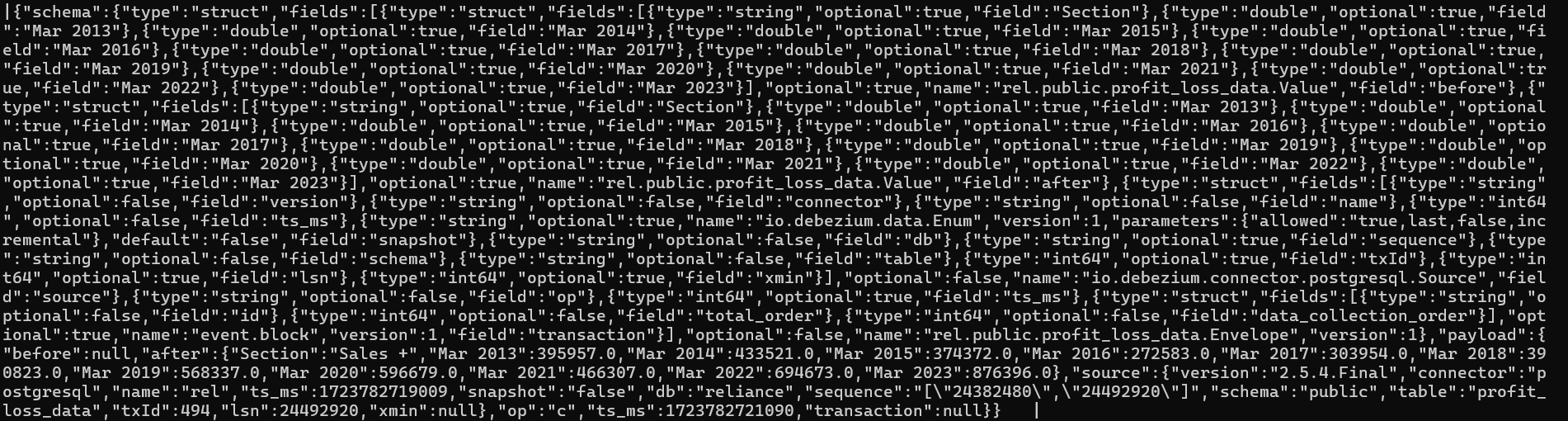
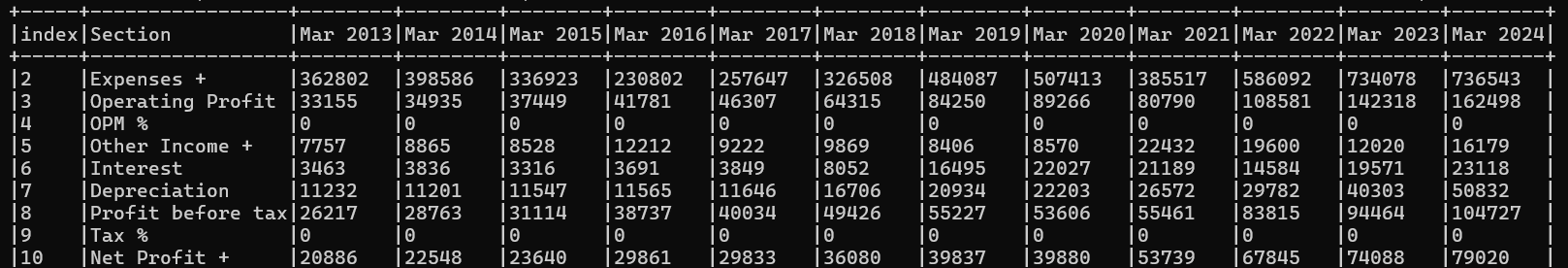
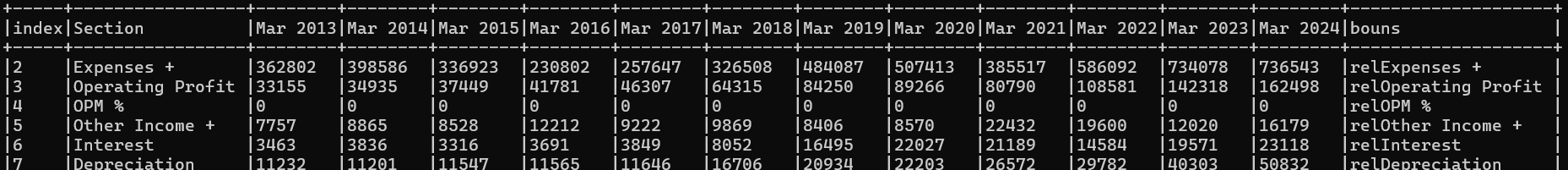
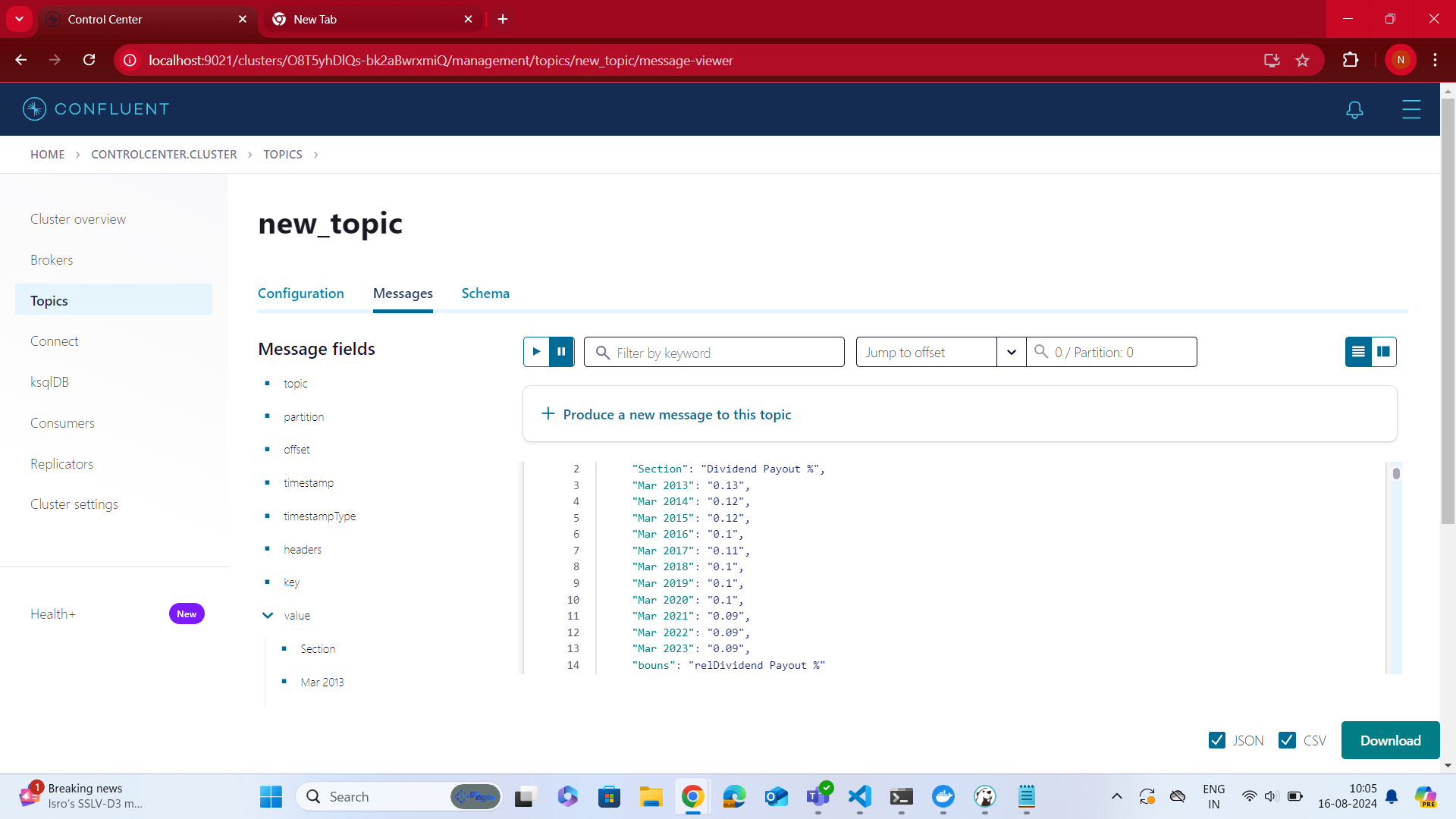
These are some of the key features of vault

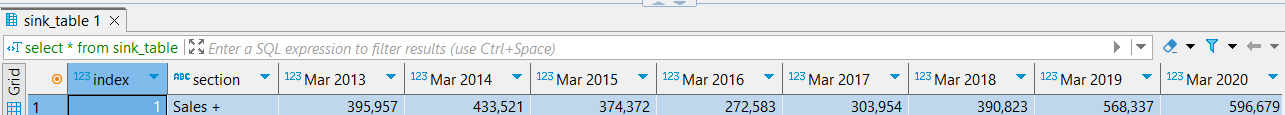
**Vault On Docker**

* Step 1:
  + Pull the docker image on of Vault  
    “docker pull vault:1.13.3”
* Step 2:
  + Create a container   
    “docker run --name vault -p 8200:8200 vault:1.13.3”  
    
  + Below you will be able to root token and make sure not exit or interrupt the command.
* Step 3:
  + Now you must open docker vault terminal, or ubuntu terminal and the following commands.
    - “export VAULT\_ADDR='http://0.0.0.0:8200'”
    - “export VAULT\_TOKEN=<root token>”
    - Vault status
  + On successful execution you would see similar output as below image.  
    
  + Now run “vault auth enable userpass”  
      
    This method allows users to authenticate with Vault using a username and password.
  + Now we set the username and password for that the command is as follow:  
    “vault write auth/userpass/users/<username> password=<password> policies=default”
* Step 4:
  + Open a browser and search for “localhost:8200”  
    and you a similar interface as shown below:  
    
  + Select username method for sign in
  + After signing in you interface as below image:
  + To create kv engine use below command;  
    “vault kv put secret/cred username=<username> password=<password>”
* Step 5:
  + Create a docker compose for concourse. For docker compose file [click here](https://concourse-ci.org/docker-compose.yml)
  + Download fly cli and store it in the same folder as the docker compose file. To download [Click here](https://github.com/concourse/concourse/releases/download/v7.11.2/fly-7.11.2-windows-amd64.zip)
* Step 6:
  + Use fly login to log into your local Concourse as the test user:
  + Now you can login to your concourse web-ui using username: test and password: test
* Step 7:
  + Create job to read the data from Vault
* Step 8:
  + Using fly command to login in concourse and run yml file having the pipeline to execute the task.  
    “fly -t tutorial set-pipeline –p <set pipeline name> -c <file name>.yml”  
    
  + Pipelines are paused when first created. To unpause use below command  
    “fly -t tutorial unpause-pipeline –p <pipeline name>”  
    
  + Trigger the job and watch it run to completion  
    “fly -t tutorial trigger-job –job <pipeline name>/<job name> --watch”  
    

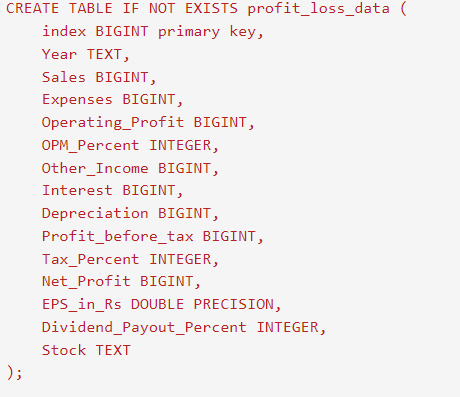
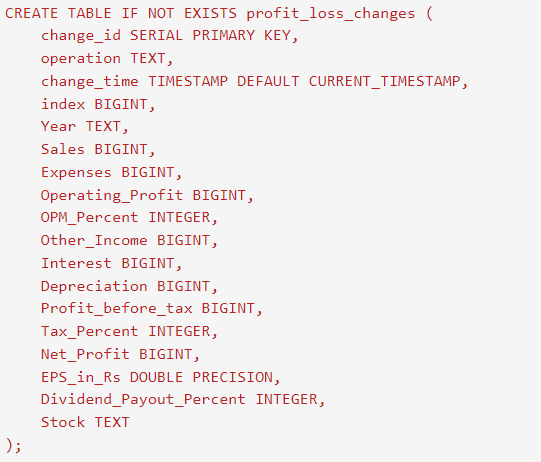
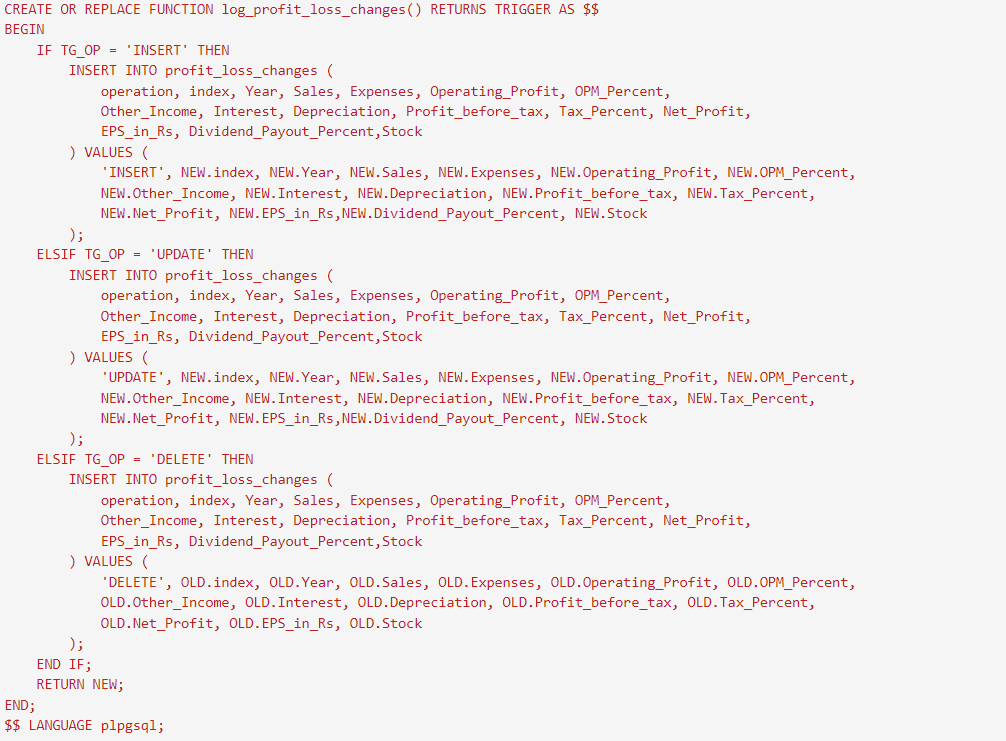
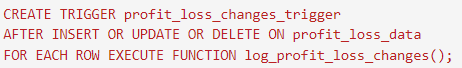
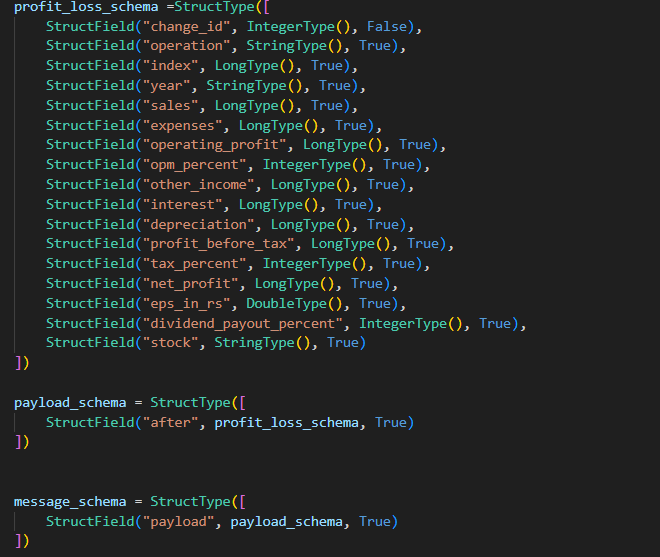
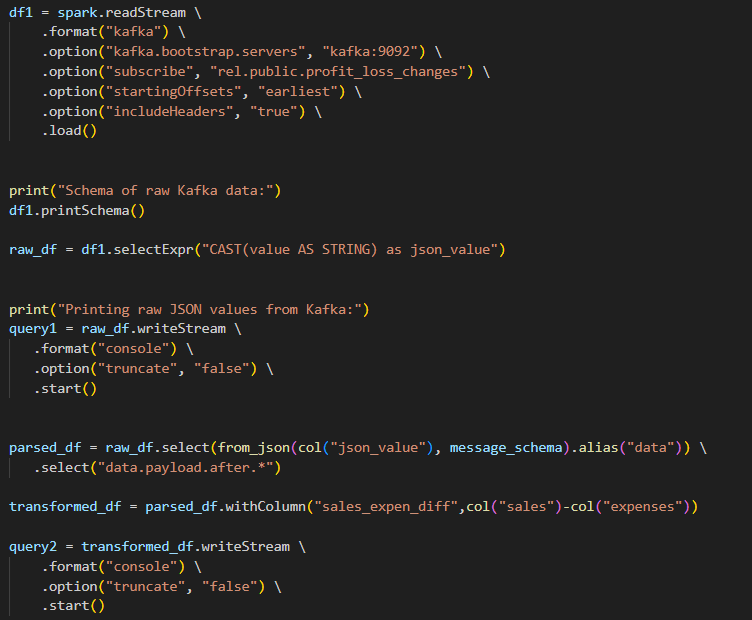
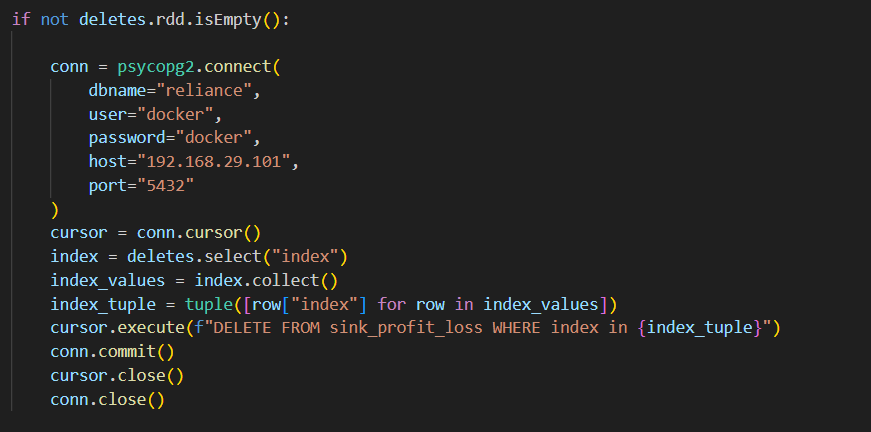
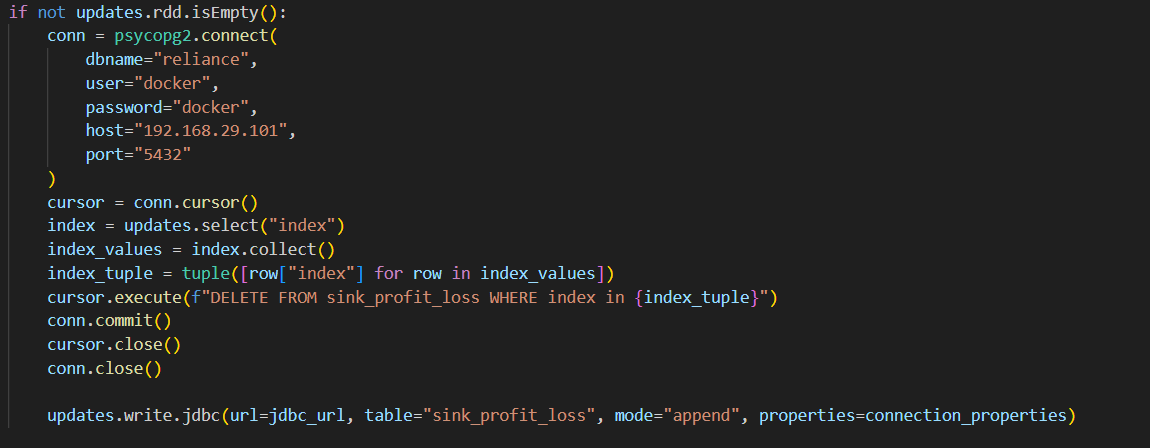
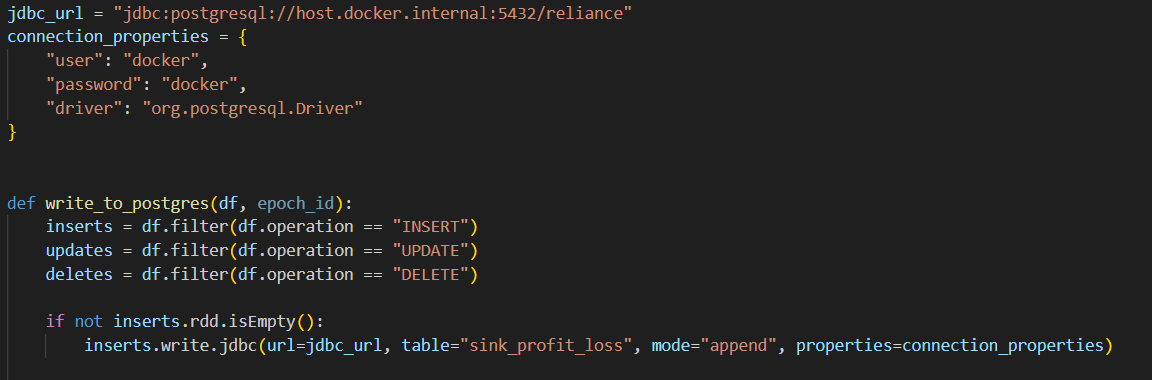
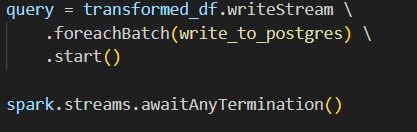
**SCRIPT FOR SCRAPPING**

* Scrapping profit and loss data of RELIANCE from screener via bs4 (beautifulsoup)
  + Importing the required libraries:
  + Providing the link for reliance web page:
  + Using requests to send HTTP get request after request we will parse a HTML document using BeautifulSoup  
    
  + Below code will fetch the data from html document by performing find operation and storing all the details in list format.  
    

* + Below code will remove the unnecessary rows and column  
    then set the columns and index   
    
  + Final Data frame output
* **Downloading excel file from web using selenium**
  + Importing the required libraries:
  + Creating a function to for login purpose:Here we are using webdriver for locating the button and entering the required details
  + Setting chrome as default browser to open link  
    
  + Providing a set of tasks to be performed in a certain sequence
  + After the success, we will see the excel in our download folder.
  + Excel file consist of different sheet within it so now we must write a code to read the excel file using pandas  
    Error: While reading the excel data via pandas we are getting nan is all columns  
    
* Using Concourse to run selenium script
  + Creating a pipeline code run selenium script  
    On our first we were getting an error of chrome and chrome driver version mismatch for that we use below link to download the chrome driver  
    “[https://storage.googleapis.com/chrome-for-testing-public/128.0.6613.84/linux64/chromedriver-linux64.zip”](https://storage.googleapis.com/chrome-for-testing-public/128.0.6613.84/linux64/chromedriver-linux64.zip ”)
  + 
  + Fetching the code from repo and running the concourse pipeline  
    [click here](https://github.com/Naveen-GDTC/repo.git) (sel\_script.py)  
    -Output of concourse console  
    
* **Scrapping and** **Loading to Postgres Database via Concourse**
  + Step 1:  
    Creating a pipeline code for running the python script jobs.
    - In pipeline code we will use a resource that will access the git repository where we will store our python script and Pyspark code for streaming data.   
      For code [Click here](https://github.com/Naveen-GDTC/repo.git)
  + Step 2:  
    Running the pipeline using fly command in terminal
  + Step 3:  
    In scrap script we will add a code that will load the scrapped data on postgresql  
    Psycopg and sqlalchemy will be used to connect to the database and create a table for profit and loss data.
  + Step 4:  
    Goto the concourse ui to view the process/execution of tasks or also stay on command terminal  
    After the successful run of pipeline job on concourse ui you will see a similar interface as below image 
  + You see the profit and loss data on the Postgres database.
  + Step 5:  
    Using source connector for connecting postgres database and kafka topic.  
    Data inside the topic will be a JSON and inside will have schema as below image.  
    
  + Step 6:  
    Now we will use a pyspark script to read the streaming which will retrieve live flowing data from topic
    - Extracting the raw Data and creating a dataframe   
      
    - Transforming Data
  + Step 7:  
    Sending the transformed data to another topic.  
    Note: Data sent to new topic just have table data in JSON format but doesn't have schema details   
    
  + Step 8:  
    Using a connector to establish connection between the new topic and postgres
    - We tried using sink connector but as the second topic did not had schema to it, so we were not able establish a sink connector

So, we created a pyspark script to read the data from new topic and load the data to postgres database

**Extra**

* **CDC In target table using Trigger Log**
  + Step 1:  
    Create a source table for storing the actual data and another table to store logs for all operations performed in the source table.  
    Creating the 1st table  
       
    Creating the second table for logs of 1st table  
    
  + Step 2:  
    Create a function to update the required data on logs table.
  + Step 3:  
    Create a trigger to capture all update, delete and insert operation on source table to logs table   
    
  + Step 4:  
    Create a source connector to send the data to topic and read it through pyspark code   
    - Source connector  
    -Code to read the data from kafka topic  
      
    -Code print the extract and transformed onto console  
      
      
      
      
      
      
      
      
      
      
      
      
      
      
      
      
      
      
    -Code to perform update, delete and insert on Postgres database  
      
      
      
      
      
    -For running the function for every new data and also to avoid code termination  
    
  + Step 5:  
    Once the data get inserted onto new table/ target table, before performing any update and delete set replica identity to full, after this you can see all updates deletes and inserts on the source table will get reflected on target table  
    