INFRASTRUCTURE ENGINEER TASK

Problem Statement

You have been given a dataset of customer orders from an online store. The data is in a CSV file orders.csv with the following columns:

- order_id: Unique identifier for each order
- customer_id: Unique identifier for each customer
- order_date: Date when the order was placed
- product_id: Unique identifier for each product
- product_name: Name of the product
- product_price: Price of the product
- quantity: Quantity of the product ordered

Your task is to write a Python program that performs the following tasks:

1. **Compute the total revenue generated by the online store for each month in the dataset.**
2. **Compute the total revenue generated by each product in the dataset.**
3. **Compute the total revenue generated by each customer in the dataset.**
4. **Identify the top 10 customers by revenue generated.**

revenue-analysis

My Task

My Task is a containerized application designed for processing and analyzing revenue data from a CSV file. It provides details on:

- Monthly revenue
- Product revenue
- Customer revenue
- Top customers based on spending

The application is built using Docker to ensure a consistent development and testing environment.

Prerequisites

Before you start, ensure you have the following installed:

- **Docker**: For containerizing the application.
- **Docker Compose**: For managing multi-container Docker applications.

Getting Started

Follow these steps to set up and run the application:

Build Docker Images

docker-compose build

Running the Application

To start the application:

docker-compose up app

Running Tests

To run tests:

docker-compose up test

This command builds and starts the test container, running your test suite.

Stopping and Cleaning Up

When finished, stop and remove all containers:

docker-compose down

Cleans up containers defined in docker-compose.yml.

Troubleshooting

If you encounter issues:

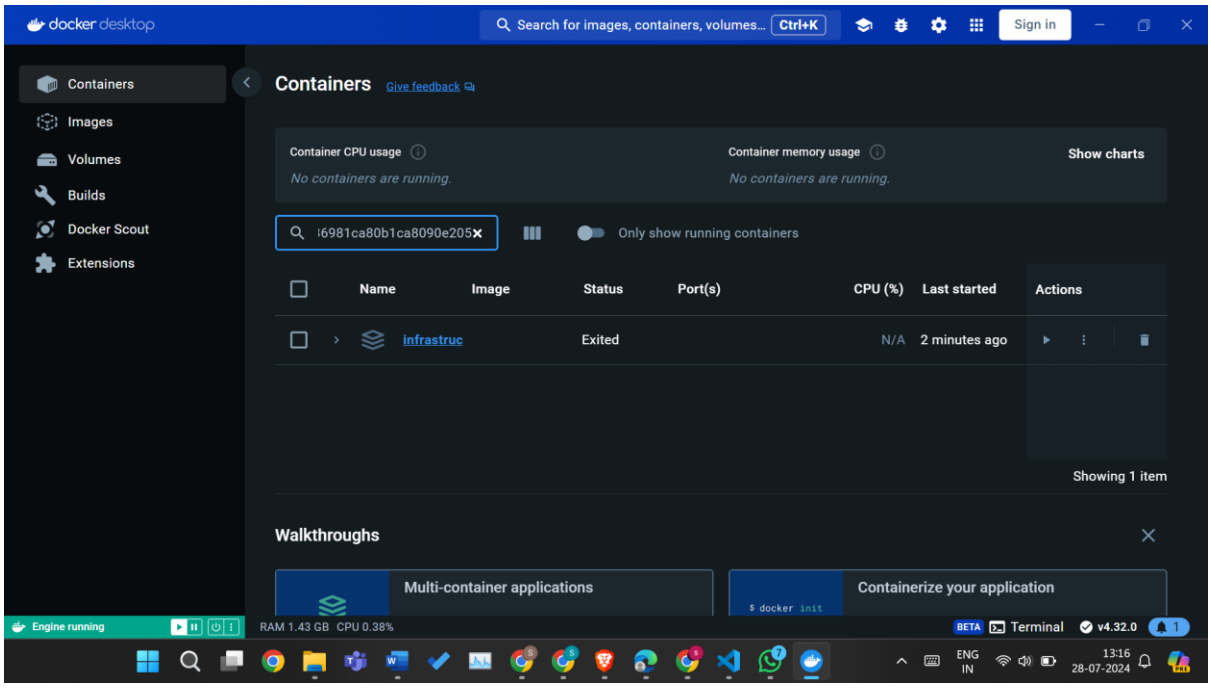
- **Docker Daemon Issues:** Ensure Docker Desktop is running.
- **Permission Problems:** Run Docker commands as administrator or adjust permissions.
- **Build Failures:** Check Dockerfile for correct dependencies.

Key Files

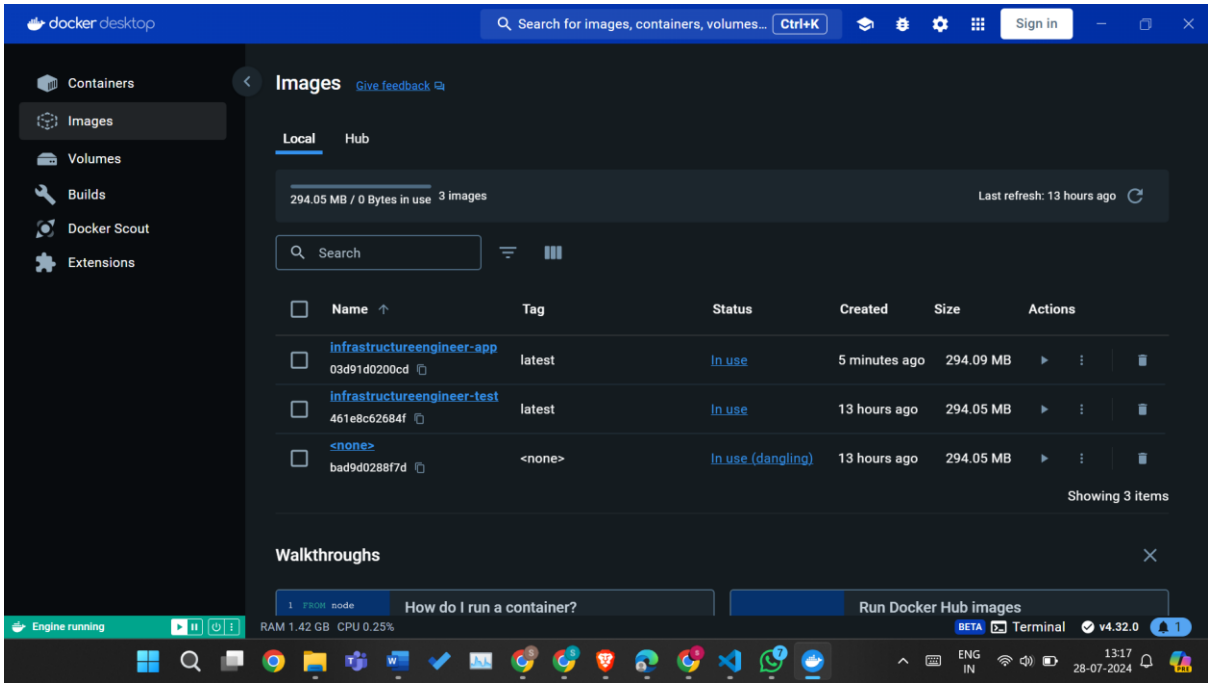
- **Dockerfile:** Defines the application image build.
- **Dockerfile.test:** Defines the test image build.
- **docker-compose.yml:** Docker Compose configuration.
- **requirements.txt:** Lists Python packages needed for the app.
- **app.py:** Contains the main application logic.
- **test_app.py:** Contains the test cases for the application.

DOCKER

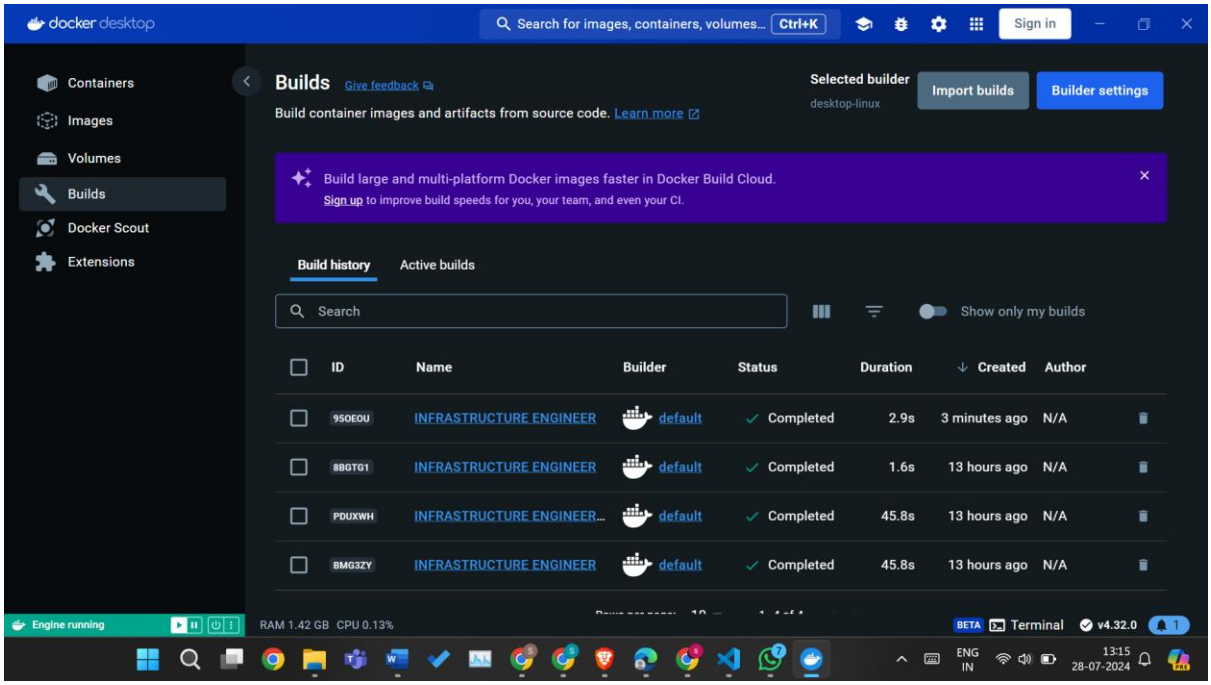
CONTAINERS



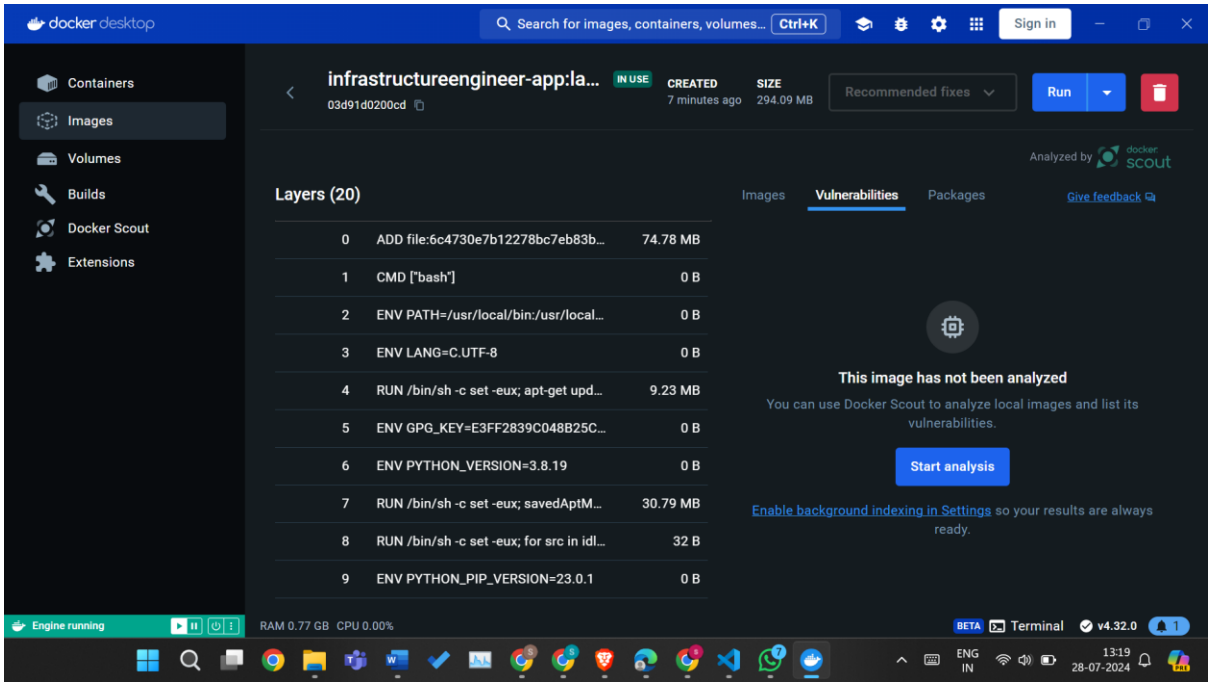
IMAGES



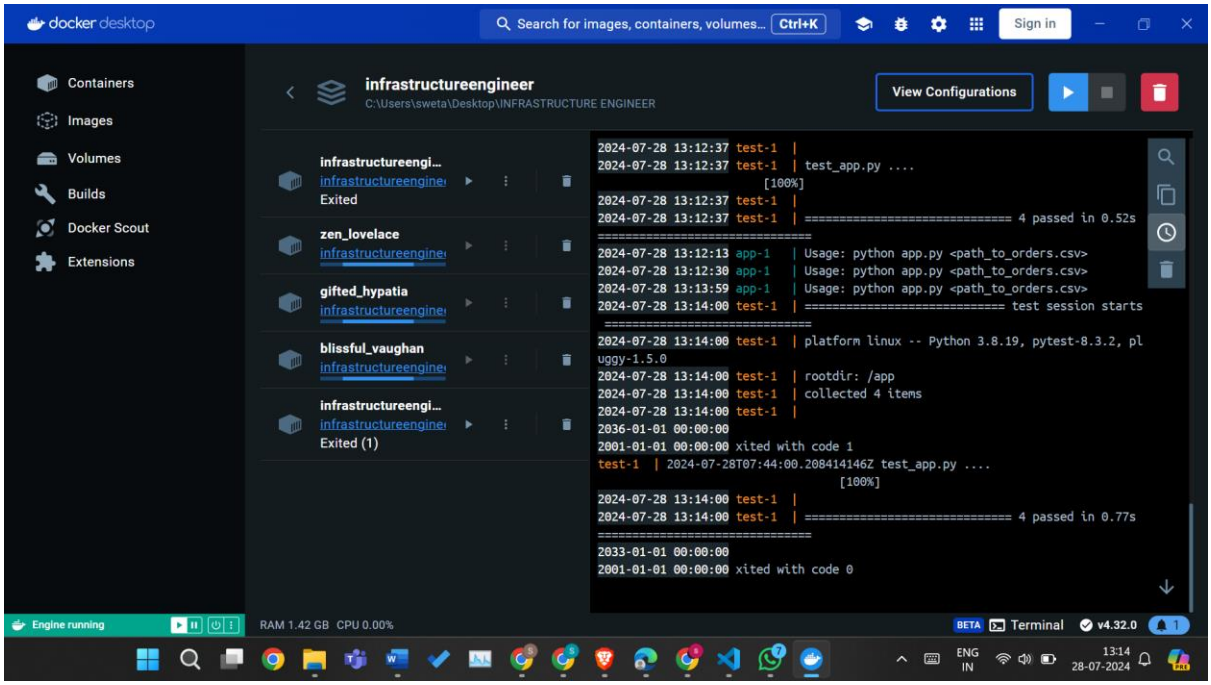
BUILDS

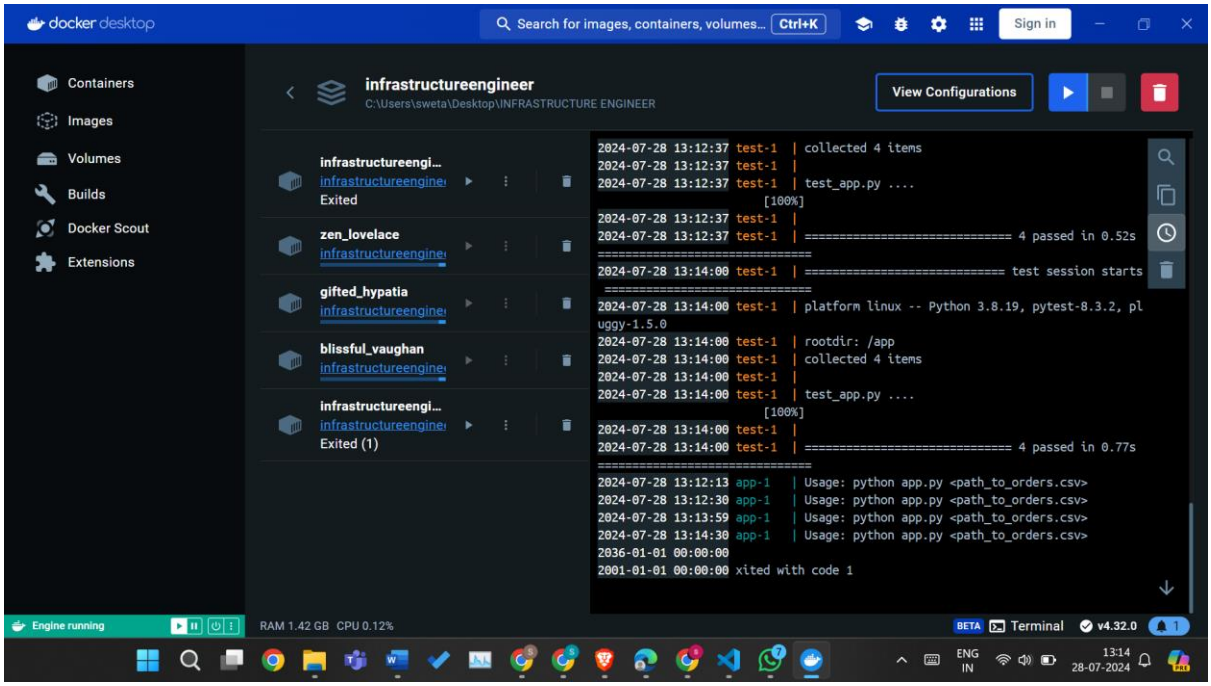


LAYERS



LOGS OF APP.PY AND TEST_APP.PY





RESULT OF app.py and test_app.py

app.py

1. Importing Pandas

- **import pandas as pd**: We use Pandas, a powerful library for handling data, to read and analyze the CSV file.

2. Reading the Data

- **read_data(file_path)**: This function takes the path to our CSV file and loads the data into a Pandas DataFrame. If something goes wrong (like if the file isn't found), it raises an error.

3. Calculating Monthly Revenue

- **compute_monthly_revenue(df)**: This function figures out how much revenue we made each month. It does this by:
    - Converting the order dates into a format that Pandas can work with.
    - Extracting just the month from these dates.
    - Calculating the revenue for each order (price times quantity).
    - Summing up the revenue for each month and converting the month column into a readable format.

4. Calculating Revenue by Product

- **compute_product_revenue(data)**: This function calculates the total revenue for each product. It groups the data by product name and sums up the revenue for each.

5. Calculating Revenue by Customer

- **compute_customer_revenue(data)**: This function calculates how much revenue each customer has generated. It groups the data by customer ID and sums up their revenue.

6. Finding Top Customers

- **top_customers_by_revenue(customer_revenue, top_n=10)**: This function identifies the top customers based on their total revenue. It sorts the customers by revenue in descending order and picks the top N (default is 10) customers.

7. Putting It All Together

- **main(file_path)**: This function orchestrates everything. It:
    - Reads the data from the CSV file.
    - Calculates monthly revenue, product revenue, and customer revenue.
    - Finds the top customers by revenue.
    - Returns all these results in a structured format.

8. Running the Script

- **Command-Line Execution**: If you run this script directly from the command line, it expects one argument: the path to the CSV file. It then processes the file and prints out:
    - Monthly revenue
    - Revenue by product
    - Revenue by customer
    - The top customers by revenue

This script helps in analyzing revenue data by breaking it down into meaningful insights such as monthly trends, product performance, customer contributions, and identifying top customers.

```
PS C:\Users\sweta\Desktop\INFRASTRUCTURE ENGINEER> python app.py order.csv
Monthly Revenue:
     month  revenue
0  2024-01      735
1  2024-02     1020
2  2024-03     1135
3  2024-04      515

Product Revenue:
   product_name  revenue
0     Product A      300
1     Product B      120
2     Product C      350
3     Product D      240
4     Product E      120
5     Product F      240
6     Product G       90
7     Product H      120
8     Product I      100
9     Product J      100
10    Product K      110
11    Product L      195
12    Product M       90
13    Product N       75
14    Product O      170
15    Product P      200
16    Product Q      240
17    Product R      105
18    Product S      300
19    Product T      140
```

```
Customer Revenue:
    customer_id  revenue
0           101      300
1           102      120
2           103      350
3           104      240
4           105      120
5           106      240
6           107       90
7           108      120
8           109      100
9           110      100
10          111      110
11          112      195
12          113       90
13          114       75
14          115      170
15          116      200
16          117      240
17          118      105
18          119      300
19          120      140

Top Customers:
   customer_id  revenue
0          103      350
1          101      300
2          119      300
3          104      240
4          106      240
5          117      240
6          116      200
7          112      195
8          115      170
9          120      140
```

test_app.py

How test_app.py Works with app.py

1. **Imports Functions**:
    - test_app.py imports functions from app.py so it can test them.
2. **Defines Tests**:
    - It creates specific tests for each function, checking if they give the right results.
3. **Provides Sample Data**:
    - test_app.py uses example data to test the functions.
4. **Compares Results**:
    - It runs the functions with the sample data and compares the actual output to the expected results.

```
5          105       120
PS C:\Users\sweta\Desktop\INFRASTRUCTURE ENGINEER> python -m unittest test_app
....
----------------------------------------------------------------------Ran 4 tests in 0.017s

OK
PS C:\Users\sweta\Desktop\INFRASTRUCTURE ENGINEER>
```