# Project 42: E-commerce Sales Analysis

**Team Members:   
1. Radhika Kalluri (263565)**

**2. Sonu Banoth (239932)**

**3. Nagarani Tummala(262767)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Description:**  
Create an ETL pipeline to collect sales data from e-commerce platforms, transform it, and load it into a database for business intelligence analysis.

Milestone 1: Sales Data Extraction

* Extract sales data from e-commerce platforms via their APIs.
* Handle data extraction for different types of transactions.

Milestone 2: Data Transformation

* Clean and transform sales data into a consistent format.
* Calculate key metrics like revenue, profit margins, and customer acquisition cost.

Milestone 3: Database Loading and Reporting

* Choose a database (e.g., MySQL, PostgreSQL) and design a schema.
* Load transformed data into the database and create reports for business analysis.  
  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Milestone 1: Sales Data Extraction**Step-by-Step Process to Extract Sales Data from E-commerce Platforms via APIs

We will use Amazon's Selling Partner API (SP-API) as our example to extract sales data. This guide includes all necessary steps and source code examples.

**Step 1:** Set Up API Access

Register as a Developer on Amazon Seller Central:

Sign in to Amazon Seller Central.

Navigate to Apps & Services > Manage Your Apps.

Click on Register as a developer.

Create an IAM User:

Go to the AWS Management Console.

Navigate to IAM > Users > Add User.

Set User type to Programmatic access.

Attach the policy AmazonSPAPIPolicy to this user.

Create and Configure Security Profile:

In Amazon Seller Central, go to Apps & Services > Develop Apps > Add New Developer Profile.

Fill in the necessary information to create the security profile.

Note down the Client ID, Client Secret, and Security Profile ID.

Authorize the Application:

Generate an authorization URL using the Client ID and Redirect URI.

Direct the seller to this URL to authorize the application and retrieve a refresh token.

**Step 2:** Obtain API Keys and Tokens

Obtain AWS Credentials:

After creating the IAM user, download the Access Key ID and Secret Access Key.

Get LWA (Login with Amazon) Access Token:

python

import requests

def get\_lwa\_access\_token(client\_id, client\_secret, refresh\_token):

url = "https://api.amazon.com/auth/o2/token"

payload = {

'grant\_type': 'refresh\_token',

'client\_id': client\_id,

'client\_secret': client\_secret,

'refresh\_token': refresh\_token

}

headers = {'Content-Type': 'application/x-www-form-urlencoded'}

response = requests.post(url, data=payload, headers=headers)

return response.json()['access\_token']

# Example usage

client\_id = 'YOUR\_CLIENT\_ID'

client\_secret = 'YOUR\_CLIENT\_SECRET'

refresh\_token = 'YOUR\_REFRESH\_TOKEN'

access\_token = get\_lwa\_access\_token(client\_id, client\_secret, refresh\_token)

Get STS (Security Token Service) Credentials:

python

import boto3

def get\_sts\_credentials(aws\_access\_key\_id, aws\_secret\_access\_key, role\_arn):

client = boto3.client('sts', aws\_access\_key\_id=aws\_access\_key\_id, aws\_secret\_access\_key=aws\_secret\_access\_key)

response = client.assume\_role(RoleArn=role\_arn, RoleSessionName='SPAPISession')

return response['Credentials']

# Example usage

aws\_access\_key\_id = 'YOUR\_AWS\_ACCESS\_KEY\_ID'

aws\_secret\_access\_key = 'YOUR\_AWS\_SECRET\_ACCESS\_KEY'

role\_arn = 'arn:aws:iam::YOUR\_AWS\_ACCOUNT\_ID:role/ROLE\_NAME'

sts\_credentials = get\_sts\_credentials(aws\_access\_key\_id, aws\_secret\_access\_key, role\_arn)

**Step 3:** Create API Request Scripts

Prepare Request Headers:

python

def get\_api\_headers(access\_token, sts\_credentials):

headers = {

'x-amz-access-token': access\_token,

'x-amz-security-token': sts\_credentials['SessionToken'],

'Content-Type': 'application/json'

}

return headers

Fetch Sales Data:

python

def fetch\_sales\_data(api\_url, headers, marketplace\_ids, created\_after):

params = {'MarketplaceIds': marketplace\_ids, 'CreatedAfter': created\_after}

response = requests.get(api\_url, headers=headers, params=params)

return response.json()

# Example usage

api\_url = 'https://sellingpartnerapi-na.amazon.com/orders/v0/orders'

headers = get\_api\_headers(access\_token, sts\_credentials)

marketplace\_ids = ['ATVPDKIKX0DER'] # Example for US marketplace

created\_after = '2024-01-01T00:00:00Z' # Example date

sales\_data = fetch\_sales\_data(api\_url, headers, marketplace\_ids, created\_after)

Handle Different Types of Transactions:

Modify the endpoint and parameters to handle different types of transactions such as refunds or cancellations.

Store Raw Data:

python

import pandas as pd

def save\_data\_to\_csv(data, file\_name):

df = pd.DataFrame(data['orders'])

df.to\_csv(file\_name, index=False)

# Example usage

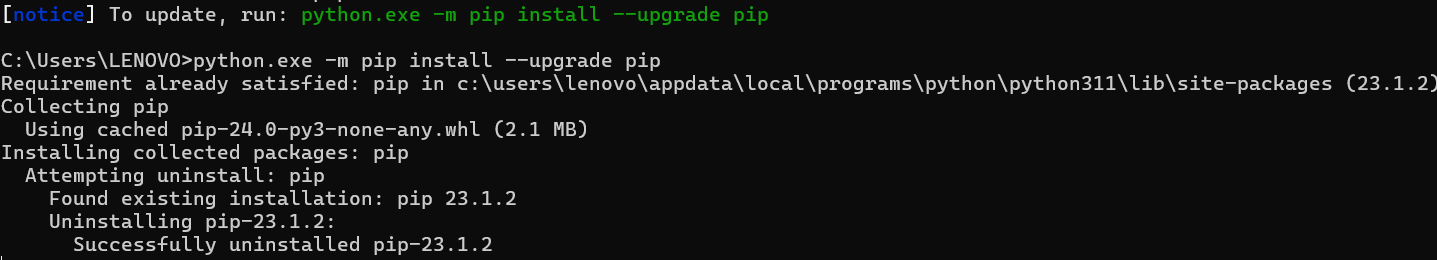
save\_data\_to\_csv(sales\_data, 'amazon\_sales\_data.csv')

kaggle datasets download -d thedevastator/unlock-profits-with-e-commerce-sales-data

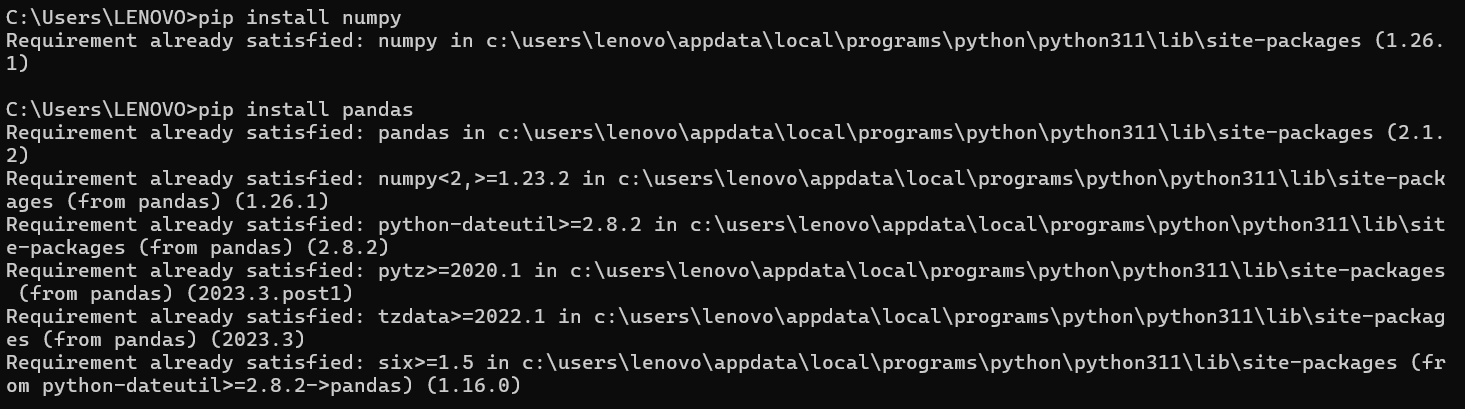
https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data

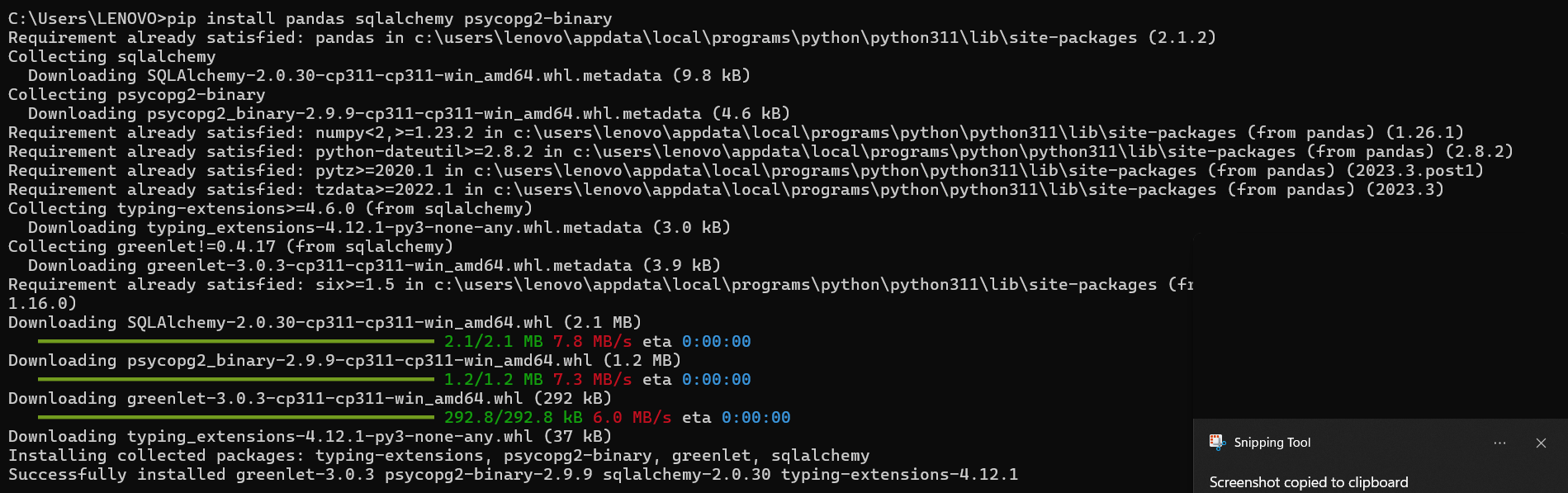
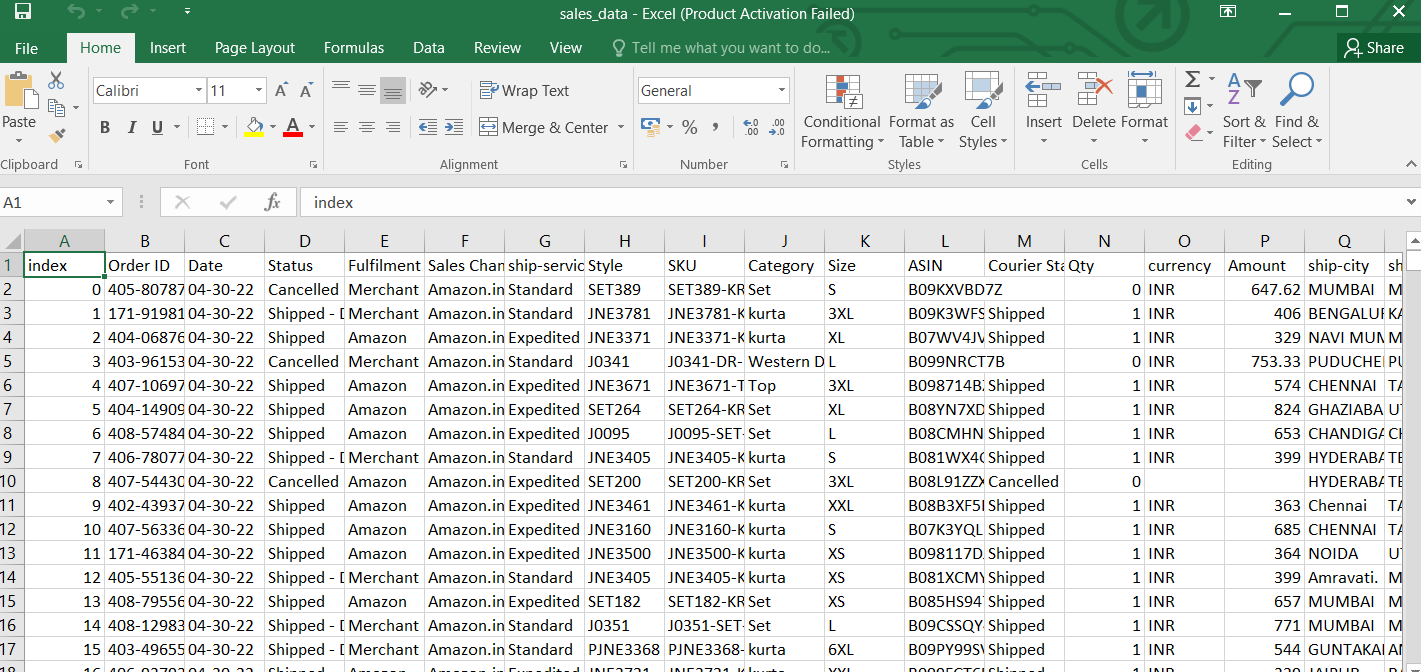
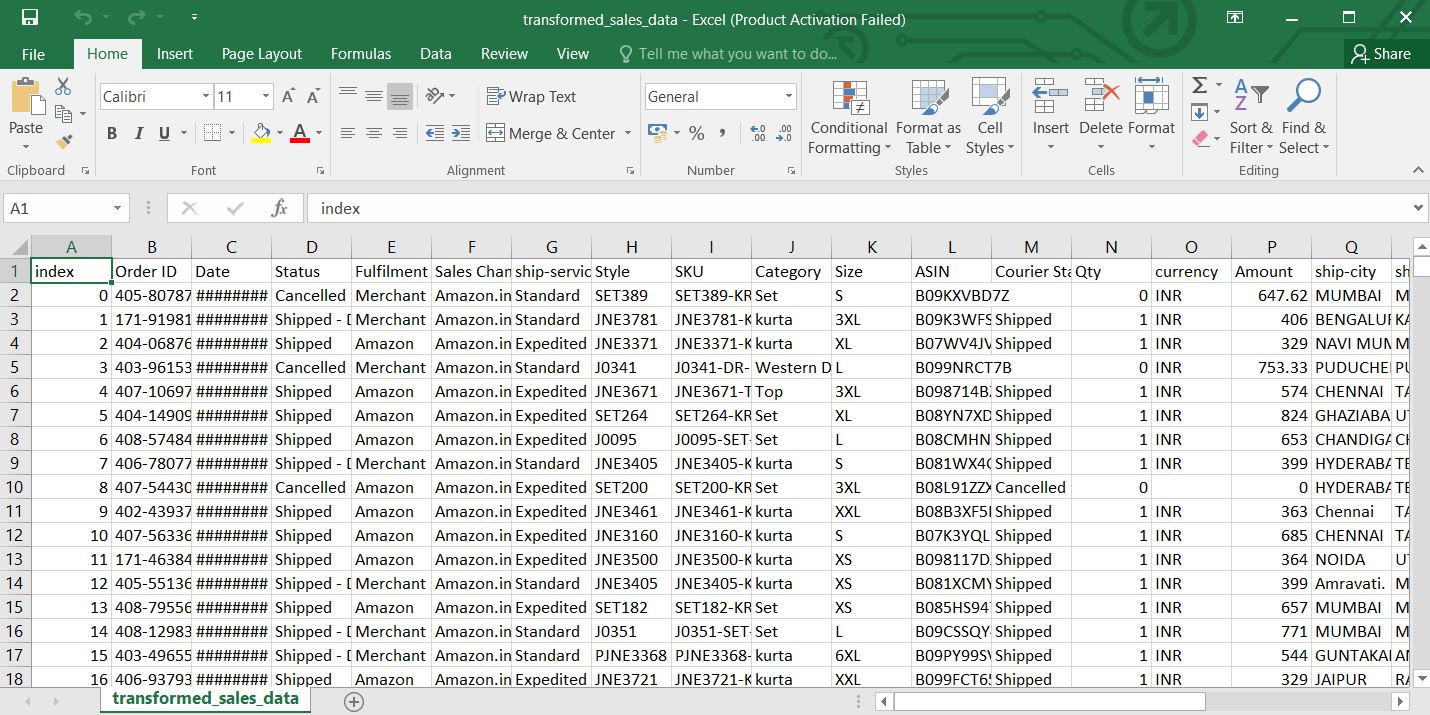
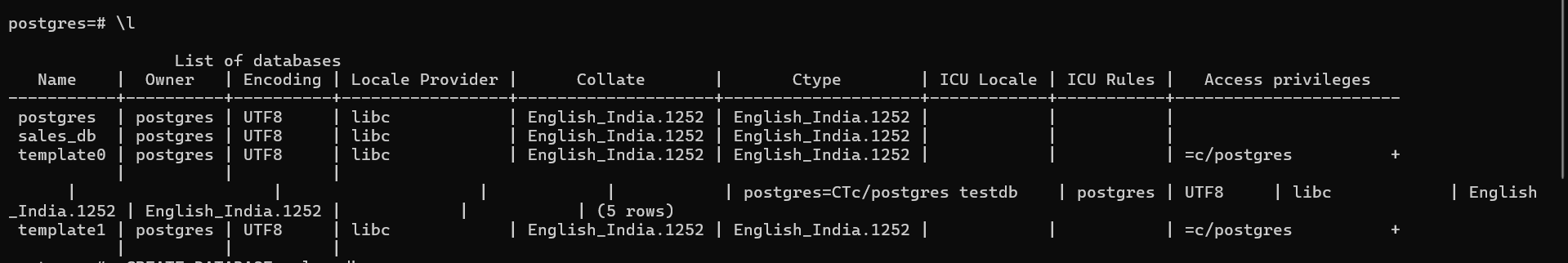
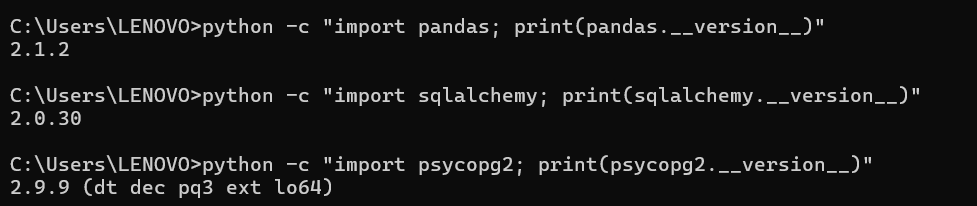
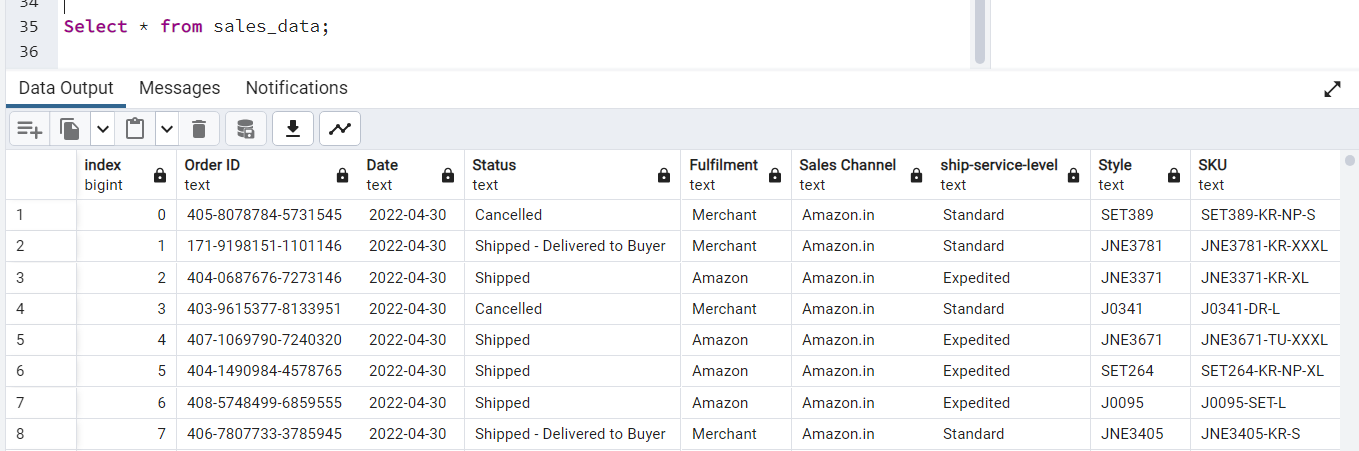
<https://www.kaggle.com/code/jaysonli/e-commerce-sales-analysis>  
Since we're not using an API and the data is already available in the CSV file from kaggle website. The file sales\_data.csv is located at C:\Users\LENOVO\Downloads\Ecommerce.  
**---------------------------------------------------------------------------------------------------**

**Milestone 2: Data Transformation**

We'll clean and transform the sales data from the CSV file. The transformation includes handling missing values, normalizing date formats, calculating revenue, cost, and profit margins. 

**Libraries to be used:**

* pandas
* numpy  
  

**Code for Data Transformation:**

import pandas as pd

import numpy as np

# Define the path to the CSV file

file\_path = r'C:\Users\LENOVO\Downloads\Ecommerce\sales\_data.csv'

# Step 1: Load the Data

df = pd.read\_csv(file\_path, low\_memory=False)

print("Original Data:")

print(df.head())

# Step 2: Clean the Data

# Remove duplicates

df = df.drop\_duplicates()

# Normalize Date Format to YYYY-MM-DD

df['Date'] = pd.to\_datetime(df['Date'], format='%m-%d-%y', errors='coerce').dt.strftime('%Y-%m-%d')

# Handle missing values in 'Amount' and 'Qty'

df['Amount'] = pd.to\_numeric(df['Amount'], errors='coerce').fillna(0)

df['Qty'] = pd.to\_numeric(df['Qty'], errors='coerce').fillna(0)

# Step 3: Transform the Data

# Calculate Revenue

df['Revenue'] = df['Amount'] \* df['Qty']

# Assuming a fixed cost percentage of 70% of the sale price

df['Cost'] = df['Amount'] \* df['Qty'] \* 0.7

df['Profit'] = df['Revenue'] - df['Cost']

print("Transformed Data:")

print(df.head())

# Step 4: Calculate Key Metrics

total\_revenue = df['Revenue'].sum()

total\_profit = df['Profit'].sum()

print(f"Total Revenue: {total\_revenue}")

print(f"Total Profit: {total\_profit}")

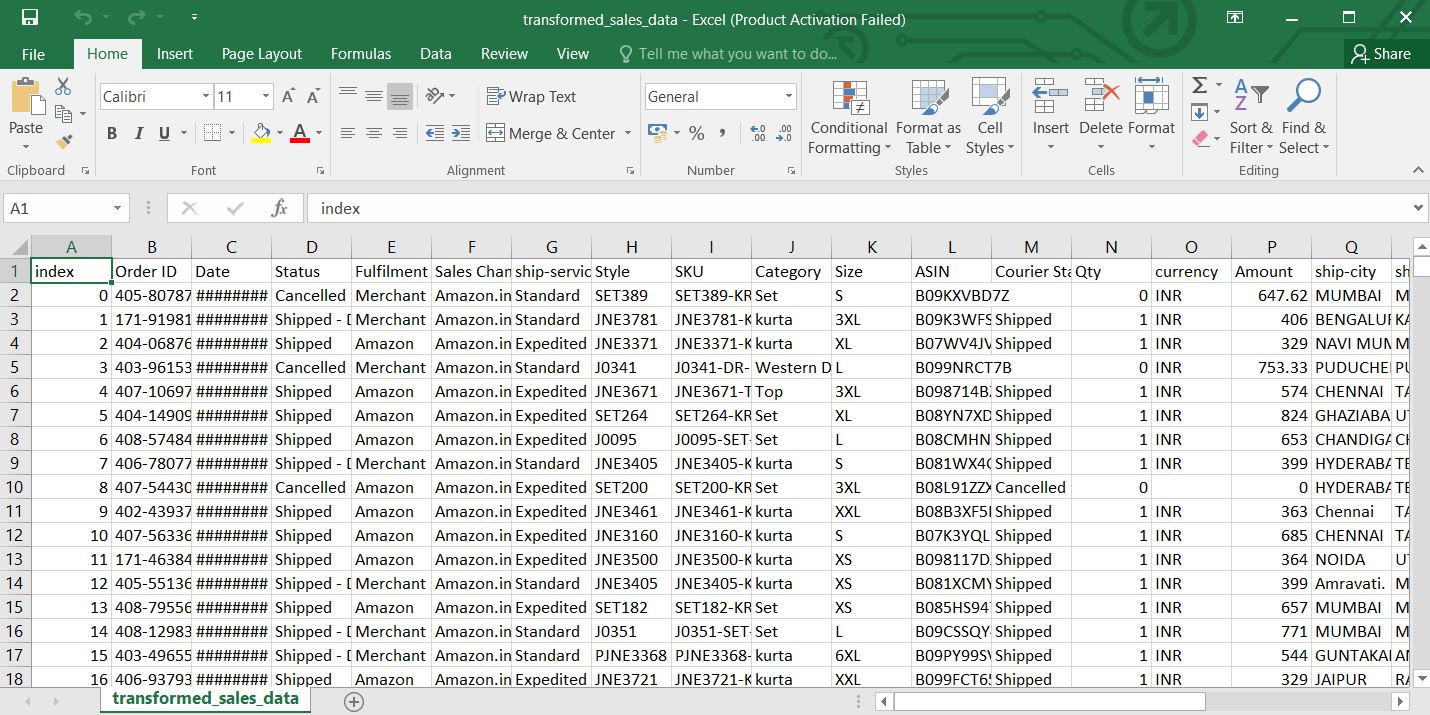
# Step 5: Save the Transformed Data

transformed\_file\_path = r'C:\Users\LENOVO\Downloads\Ecommerce\transformed\_sales\_data.csv'

df.to\_csv(transformed\_file\_path, index=False)

print("Transformed data saved to 'transformed\_sales\_data.csv'")



****

**Milestone 3: Database Loading and Reporting**

We'll load the transformed data into a PostgreSQL database and create reports for business analysis.

**Software and Tools:**

* PostgreSQL (Database)
* pgAdmin (Database Management)
* SQLAlchemy (Python SQL Toolkit)
* psycopg2 (PostgreSQL adapter for Python)

**Steps:**

1. **Install PostgreSQL and pgAdmin:**
   * Download and install PostgreSQL from [PostgreSQL Official Site](https://www.postgresql.org/download/).
   * During installation, set a username and password for the PostgreSQL superuser.
2. **Install Required Python Libraries:**

sh

pip install pandas sqlalchemy psycopg2

1. **Create Database Schema in PostgreSQL:**
   * Open pgAdmin and create a new database named ecommerce.
2. **Database Schema Design:**

sql

CREATE TABLE sales (

order\_id VARCHAR PRIMARY KEY,

date DATE,

price FLOAT,

quantity INT,

currency VARCHAR,

revenue FLOAT,

cost FLOAT,

profit FLOAT

);

**Code to Load Data into PostgreSQL:**

python

from sqlalchemy import create\_engine

# Database connection string

db\_url = 'postgresql+psycopg2://username:password@localhost:5432/ecommerce'

# Create SQLAlchemy engine

engine = create\_engine(db\_url)

# Load transformed data into the PostgreSQL database

df.to\_sql('sales', engine, if\_exists='replace', index=False)

print("Data loaded into PostgreSQL database successfully.")

**Detailed Reporting:**

You can now create SQL queries in pgAdmin or use a business intelligence tool (e.g., Tableau, Power BI) connected to your PostgreSQL database for reporting.

**Example SQL Query for Reporting:**

sql

-- Total Revenue and Profit by Month

SELECT

DATE\_TRUNC('month', date) AS month,

SUM(revenue) AS total\_revenue,

SUM(profit) AS total\_profit

FROM sales

GROUP BY month

ORDER BY month;

**Complete Project Directory Structure:**

makefile

Copy code

C:\Users\LENOVO\Downloads\Ecommerce

│ transform.py # Script for data transformation

│ load\_data.py # Script to load data into PostgreSQL

│ sales\_data.csv # Original sales data

│ transformed\_sales\_data.csv# Transformed sales data

└───\_\_init\_\_.py # Initialization file for Python package

1. **Download and Install PostgreSQL:**
   * Go to the [PostgreSQL website](https://www.postgresql.org/download/) and download the installer for your operating system.
   * Follow the installation instructions and make sure to remember the username, password, and port number you set during the installation.
2. **Install the Required Python Libraries:**
   * Install psycopg2 for PostgreSQL interaction and sqlalchemy for ORM.

sh

pip install psycopg2-binary sqlalchemy

### Step 2: Create a Database and Table

1. **Open PostgreSQL:**
   * Open pgAdmin (a GUI tool for PostgreSQL) or connect to your PostgreSQL server using the command line.
   * Create a new database, e.g., sales\_db.
2. **Create a Table:**
   * Define a table schema for storing the sales data. Here’s an example SQL command to create a table:

sql

CREATE TABLE sales\_data (

index SERIAL PRIMARY KEY,

order\_id VARCHAR(50),

date DATE,

status VARCHAR(50),

fulfilment VARCHAR(50),

sales\_channel VARCHAR(50),

ship\_service\_level VARCHAR(50),

style VARCHAR(50),

sku VARCHAR(50),

category VARCHAR(50),

size VARCHAR(50),

asin VARCHAR(50),

courier\_status VARCHAR(50),

qty INT,

currency VARCHAR(10),

amount DECIMAL,

ship\_city VARCHAR(50),

ship\_state VARCHAR(50),

ship\_postal\_code VARCHAR(20),

ship\_country VARCHAR(50),

promotion\_ids VARCHAR(100),

b2b BOOLEAN,

fulfilled\_by VARCHAR(50),

revenue DECIMAL,

cost DECIMAL,

profit DECIMAL

);

### Step 3: Load Data into the Database

1. **Modify the Data Transformation Script to Include Database Loading:**
   * Update the trans.py script to load the transformed data into PostgreSQL.

python

import pandas as pd

from sqlalchemy import create\_engine

# Define the path to the CSV file

file\_path = r'C:\Users\LENOVO\Downloads\Ecommerce\sales\_data.csv'

# Step 1: Load the Data

df = pd.read\_csv(file\_path, low\_memory=False)

print("Original Data:")

print(df.head())

# Step 2: Clean the Data

# Remove duplicates

df = df.drop\_duplicates()

# Normalize Date Format to YYYY-MM-DD

df['Date'] = pd.to\_datetime(df['Date'], format='%m-%d-%y', errors='coerce').dt.strftime('%Y-%m-%d')

# Handle missing values in 'Amount' and 'Qty'

df['Amount'] = pd.to\_numeric(df['Amount'], errors='coerce').fillna(0)

df['Qty'] = pd.to\_numeric(df['Qty'], errors='coerce').fillna(0)

# Step 3: Transform the Data

# Calculate Revenue

df['Revenue'] = df['Amount'] \* df['Qty']

# Assuming a fixed cost percentage of 70% of the sale price

df['Cost'] = df['Amount'] \* df['Qty'] \* 0.7

df['Profit'] = df['Revenue'] - df['Cost']

print("Transformed Data:")

print(df.head())

# Step 4: Calculate Key Metrics

total\_revenue = df['Revenue'].sum()

total\_profit = df['Profit'].sum()

print(f"Total Revenue: {total\_revenue}")

print(f"Total Profit: {total\_profit}")

# Step 5: Save the Transformed Data

transformed\_file\_path = r'C:\Users\LENOVO\Downloads\Ecommerce\transformed\_sales\_data.csv'

df.to\_csv(transformed\_file\_path, index=False)

print("Transformed data saved to 'transformed\_sales\_data.csv'")

# Step 6: Load Data into PostgreSQL

# Database connection details

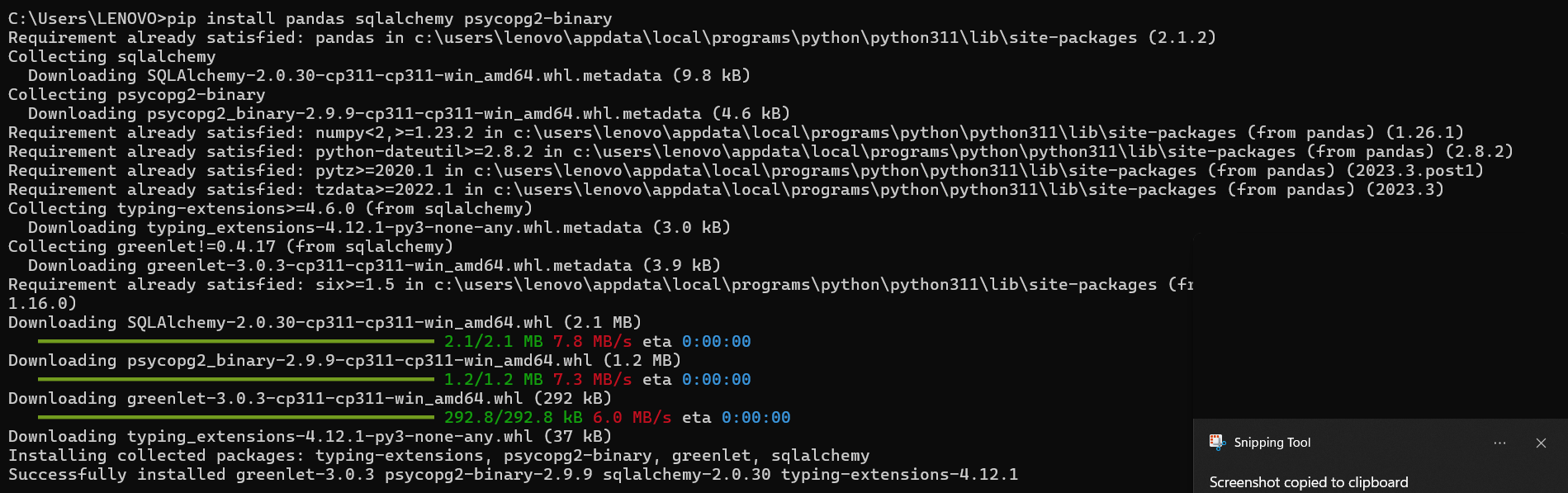
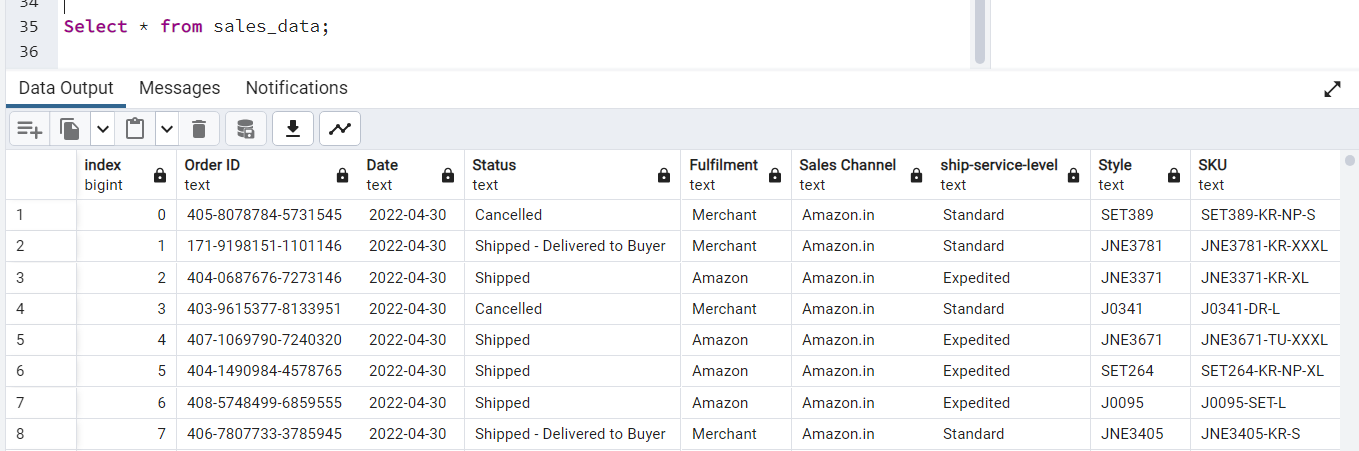
db\_url = 'postgresql+psycopg2://**username:password**@localhost:5432/sales\_db'  
#username: postgres, password :1234

# Create SQLAlchemy engine

engine = create\_engine(db\_url)

# Load DataFrame into PostgreSQL

df.to\_sql('sales\_data', engine, if\_exists='replace', index=False)

print("Data loaded into PostgreSQL database")  
  
  
  
  


### Step 4: Creating Reports

1. **Using SQL Queries for Business Analysis:**
   * You can run SQL queries to analyze the data directly in PostgreSQL.

sql

-- Example: Total Revenue and Profit

SELECT SUM(revenue) AS total\_revenue, SUM(profit) AS total\_profit FROM sales\_data;

-- Example: Revenue by Date

SELECT date, SUM(revenue) AS daily\_revenue FROM sales\_data GROUP BY date ORDER BY date;

-- Example: Top Selling Products

SELECT sku, SUM(qty) AS total\_quantity, SUM(revenue) AS total\_revenue FROM sales\_data GROUP BY sku ORDER BY total\_revenue DESC LIMIT 10;

1. **Using a BI Tool:**
   * Connect your PostgreSQL database to a BI tool like Tableau, Power BI, or Google Data Studio to create interactive dashboards and reports.

### Final Checklist

* Ensure PostgreSQL is installed and running.
* Ensure the required Python libraries are installed (psycopg2, sqlalchemy, pandas).
* Create the PostgreSQL database and table.
* Modify and run the trans.py script to load the data into the database.
* Use SQL queries or a BI tool to create reports and perform business analysis.

By following these steps, you should be able to complete Milestone 3 and have your data loaded into PostgreSQL, ready for analysis and reporting. If you encounter any errors or need further assistance, please let me know.

You can now connect to sales\_db in PostgreSQL using a BI tool or run SQL queries to analyze your data directly.

### Example SQL Queries:

* **Total Revenue and Profit:**

sql

SELECT SUM(revenue) AS total\_revenue, SUM(profit) AS total\_profit FROM sales\_data;

* **Revenue by Date:**

sql

SELECT date, SUM(revenue) AS daily\_revenue FROM sales\_data GROUP BY date ORDER BY date;

* **Top Selling Products:**

sql

SELECT sku, SUM(qty) AS total\_quantity, SUM(revenue) AS total\_revenue FROM sales\_data GROUP BY sku ORDER BY total\_revenue DESC LIMIT 10;

This completes Milestone 3. If you encounter any issues, please provide the error messages for further assistance.