**Step 1: Data Extraction Documentation**

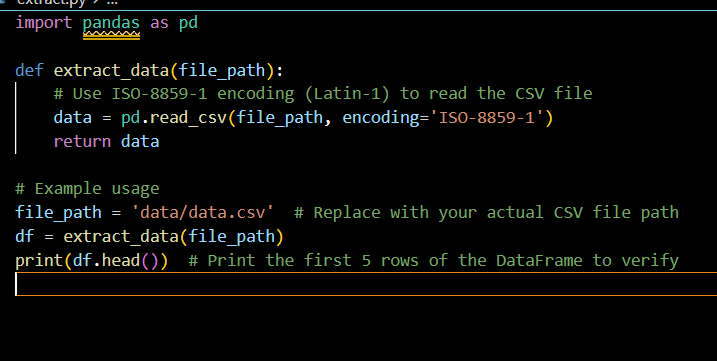
**Objective**

To extract data from a CSV file using **Pandas** in Python.

**Code Implementation**

The following script reads a CSV file using Pandas and handles special character encodings (ISO-8859-1).

**Code sample**



**Output Example (First 5 Rows of CSV Data)**

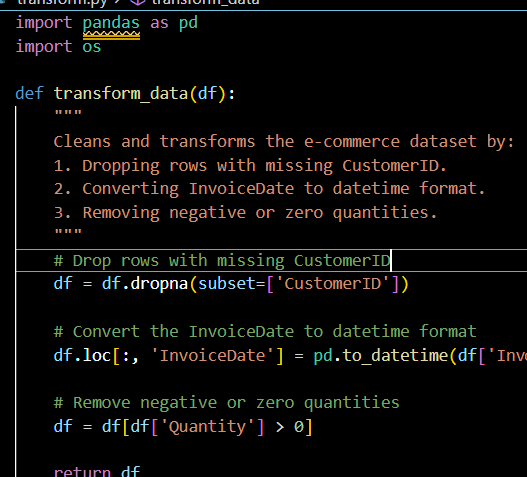
| **InvoiceNo** | **StockCode** | **Description** | **Quantity** | **InvoiceDate** | **UnitPrice** | **CustomerID** | **Country** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 536365 | 85123A | WHITE METAL LANTERN | 6 | 01-12-2010 08:26 | 2.55 | 17850 | United Kingdom |
| 536365 | 71053 | HAND WARMER UNION JACK | 6 | 01-12-2010 08:26 | 3.39 | 17850 | United Kingdom |
| 536365 | 84406B | CREAM CUPID HEARTS | 8 | 01-12-2010 08:26 | 2.75 | 17850 | United Kingdom |

### Step 2: Data Transformation Documentation

**Objective**  
To clean and transform the e-commerce dataset by removing missing values, converting date formats, and filtering invalid data.

#### Code Implementation sample

The following script processes an e-commerce dataset, performs necessary transformations like removing rows with missing CustomerID, converting InvoiceDate to a datetime format, and removing invalid or non-positive Quantity values.



#### Output Example (First 5 Rows of Transformed Data)

InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country

536365 85123A WHITE METAL LANTERN 6 2010-12-01 08:26:00 2.55 17850 United Kingdom

536365 71053 HAND WARMER UNION JACK 6 2010-12-01 08:26:00 3.39 17850 United Kingdom

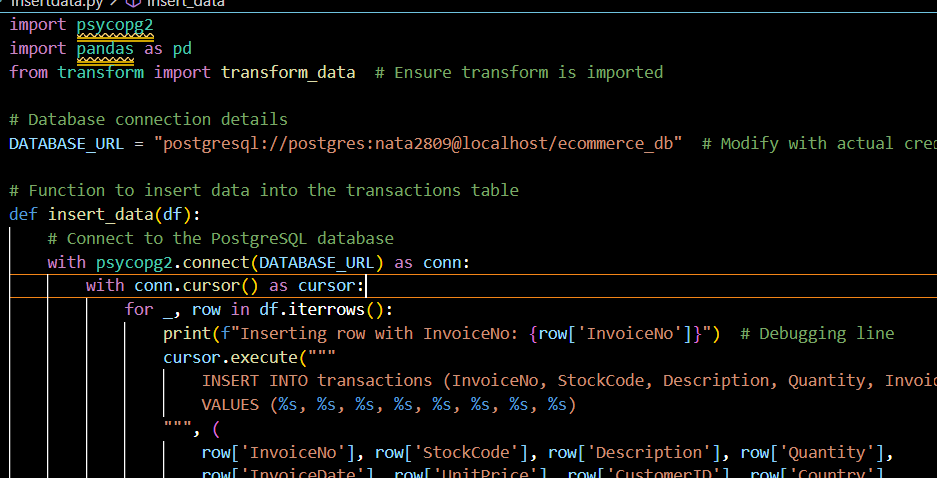
536365 84406B CREAM CUPID HEARTS COAT HANGER 8 2010-12-01 08:26:00 2.75 17850 United Kingdom

### Step 3: Data Loading Documentation

**Objective**  
To load the transformed data into a PostgreSQL database, specifically inserting records into a transactions table.

#### Code Implementation sample

The following script reads the transformed data from a CSV file, establishes a connection to the PostgreSQL database, and inserts the data into the transactions table.



#### Output Example (Console Logs)

Inserting row with InvoiceNo: 536365

Inserting row with InvoiceNo: 536366

Inserting row with InvoiceNo: 536367

...

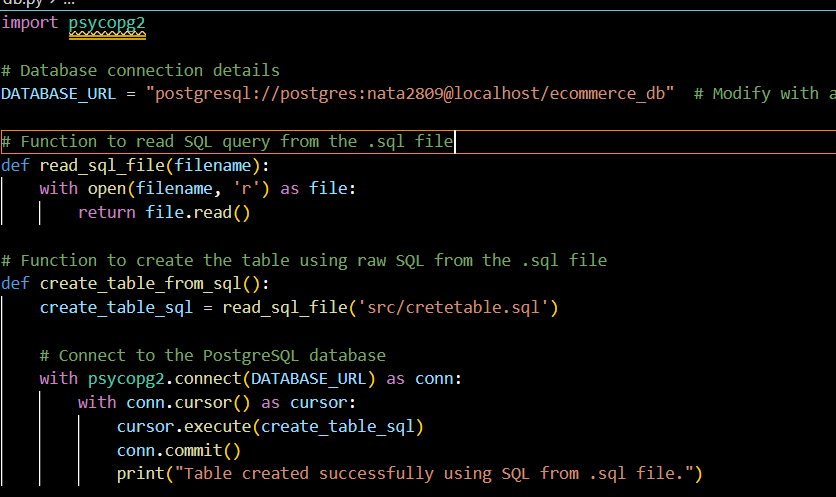
Data inserted successfully into the transactions tabl

### Step 4: Database Setup Documentation

**Objective**  
To set up the PostgreSQL database table by reading the SQL query from an external .sql file and executing it using psycopg2.

#### Code Implementation

This script reads a SQL query from an external file, connects to the PostgreSQL database, and executes the query to create the necessary table in the database.



#### Code Explanation

1. **read\_sql\_file(filename)**:
   * This function reads the content of an SQL file (filename) and returns it as a string. The file should contain the SQL query to create the database table.
2. **create\_table\_from\_sql()**:
   * This function reads the SQL query from the specified .sql file and uses psycopg2.connect() to establish a connection to the PostgreSQL database.
   * It then executes the SQL query using cursor.execute(), which creates the necessary table (e.g., transactions table) in the database.
   * Finally, it commits the transaction using conn.commit() to save the changes to the database.
3. **Database Connection**:
   * The connection string DATABASE\_URL should be updated with actual database credentials (username, password, and database name).
4. **SQL File**:
   * The SQL file (src/cretetable.sql) should contain the SQL query to create the necessary table structure. This script assumes the SQL file is in the src directory.

#### Output Example (Console Logs)

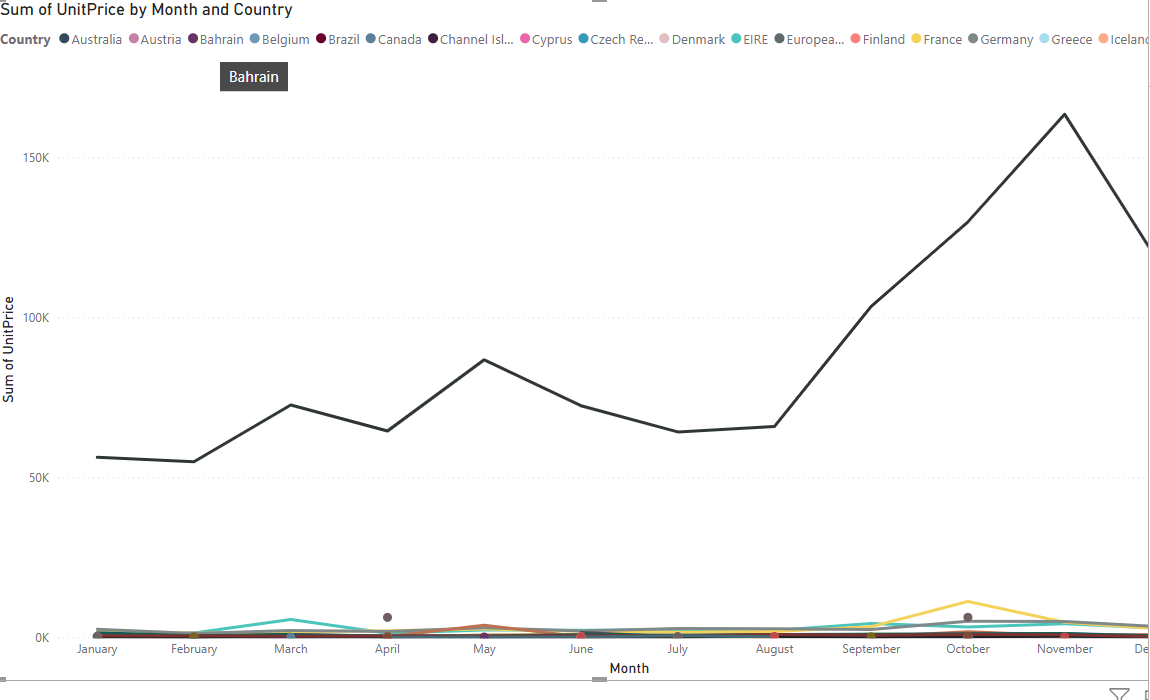
Table created successfully using SQL from .sql file.

Step 5 visualization

1. Sales Trend Over Time (Line Chart)

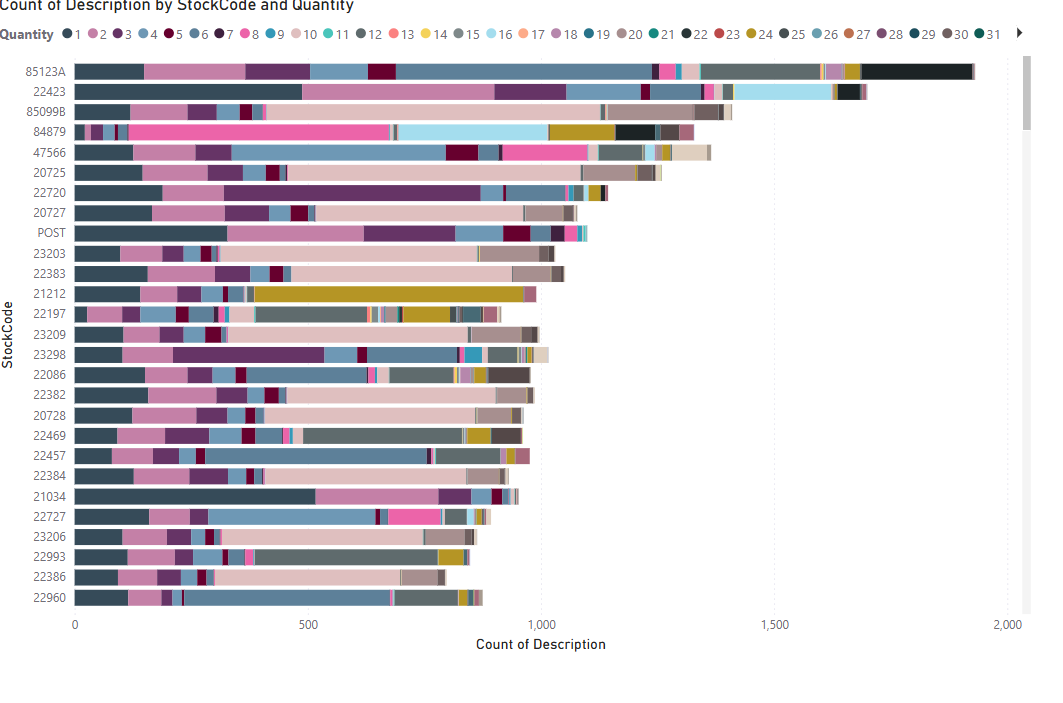
## ****1. Sales Trend Over Time (Line Chart)****

**Goal:** Show how sales are changing over time.  
1️**Visual Type:** Line Chart  
2️**X-Axis:** InvoiceDate (Convert to **Date Hierarchy** for Year/Month/Day view)  
3️**Y-Axis:** Quantity or UnitPrice \* Quantity (Total Sales)



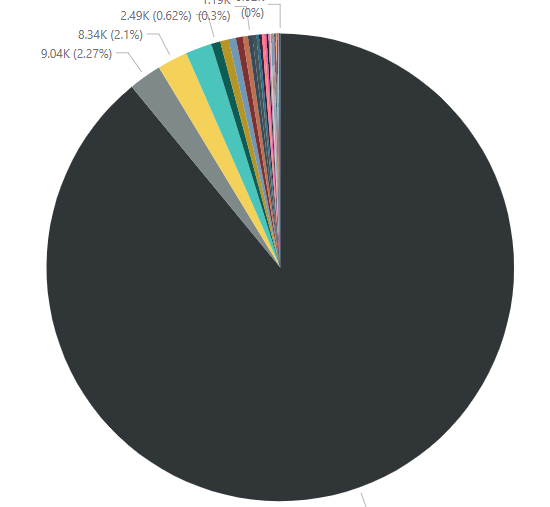
## ****2. Top Selling Products (Bar Chart)****

**Goal:** Find the most sold products.  
1️**Visual Type:** Bar Chart  
2️**X-Axis:** Description (Product Name)  
3️**Y-Axis:** SUM(Quantity) (Total Quantity Sold)



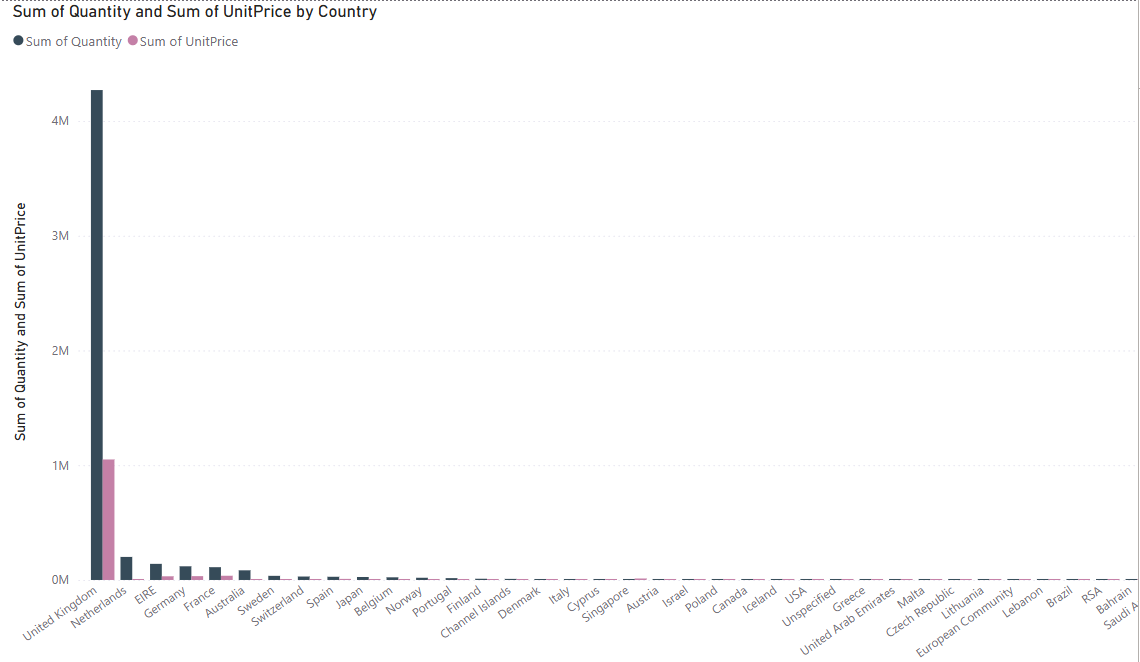
## ****3. Customer Segmentation by Country (Pie Chart)****

**Goal:** See customer distribution by country.  
1️**Visual Type:** Pie Chart  
2️**Values:** COUNT(CustomerID) (Number of Customers)  
3️**Legend:** Country



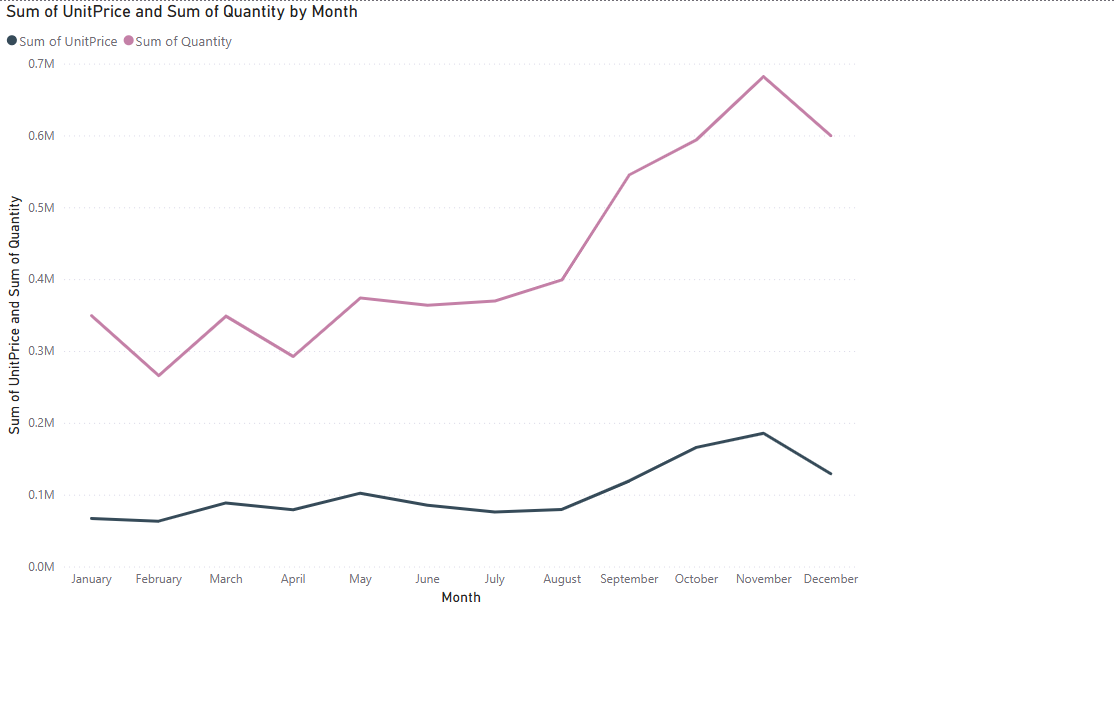
## ****4. Revenue by Country (Column Chart)****

**Goal:** Compare total sales in different countries.  
1️**Visual Type:** Clustered Column Chart  
2️**X-Axis:** Country  
3️**Y-Axis:** SUM(UnitPrice \* Quantity) (Total Revenue)



## ****5. Average Order Value Over Time (Line Chart)****

**Goal:** Track how the average order value changes over time.  
1️**Visual Type:** Line Chart  
2️**X-Axis:** InvoiceDate (Convert to **Date Hierarchy**)  
3️**Y-Axis:** AVERAGE(UnitPrice \* Quantity) (Average Order Value)



Tools for this project

Python => programing tool

Pandas =>python liberariye to reade .csv file

Power BI => for visualization data

Postgres => for database

Kaggle => for dataset

Other odbs ,pycopg2 etc