National Telecommunication Institute

Trading Company Database Analysis and Insights

Exploring Key Business Questions Through SQL Queries & Visualizations

Introduction

Project Overview

This project focuses on answering several critical business questions by utilizing a combination of SQL queries, Python programming, and data visualization techniques.

The goal is to demonstrate how various analytical tools can work together to provide actionable insights from raw data.

Key components of the project include:

- **SQL Queries**: Extracting, manipulating, and aggregating data to answer specific questions.
- **Python Programming**: Using Python libraries like Pandas and NumPy to handle and process the data effectively.
- **Data Visualization**: Leveraging popular Python visualization libraries such as Matplotlib, Seaborn, and Plotly to create meaningful charts and graphs that help interpret the data.

Vision and Goals

Project Objectives

The primary objectives of this project are:

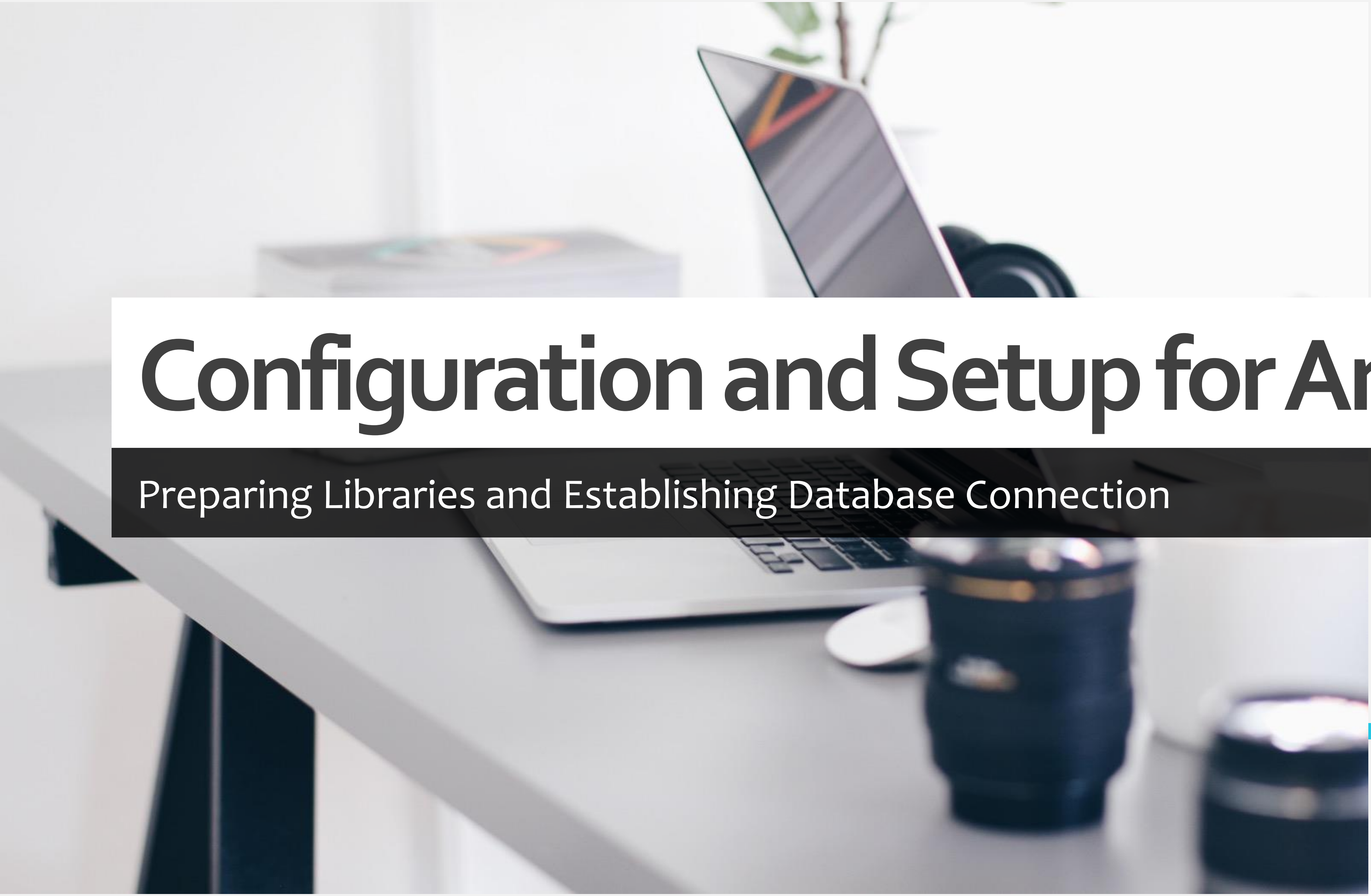
- To write and execute SQL queries that efficiently extract relevant data from large datasets.
- To process the extracted data using Python to answer business-related questions.
- To visualize the findings through compelling and clear charts, enabling stakeholders to make informed decisions based on the analysis.

Strategy and Execution

Approach and Methodology

This project follows a systematic approach to answering the business questions:

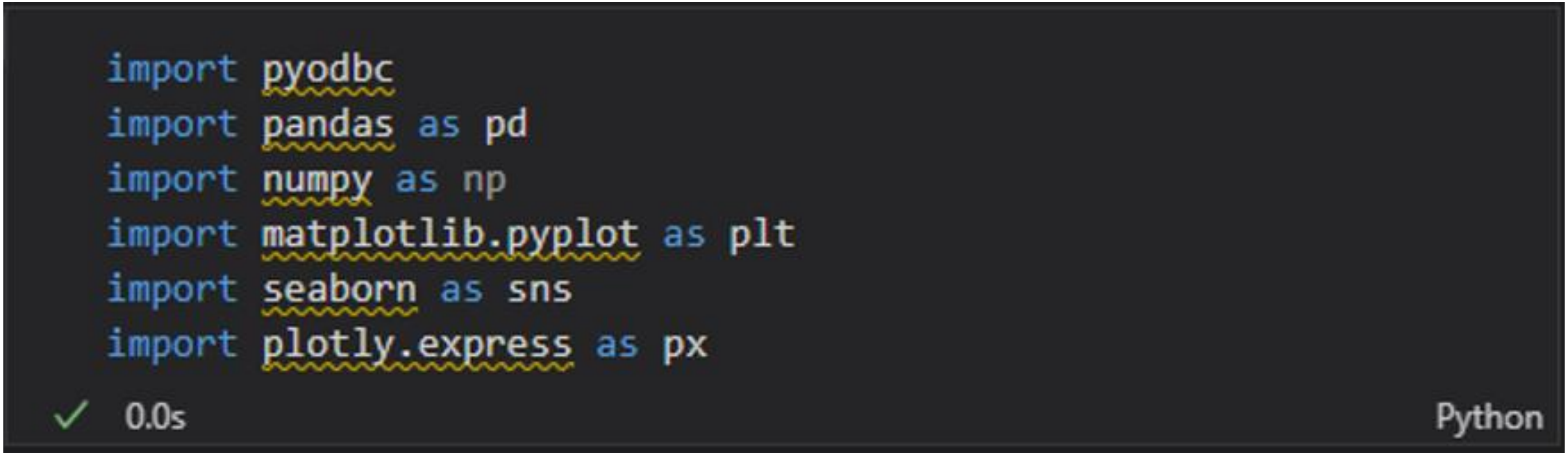
1. Data Extraction with SQL:
   SQL queries were designed to extract relevant data from the database based on the given questions. These queries involve filtering, joining, and aggregating data across multiple tables.
2. Data Processing in Python:
   Python was used to clean, transform, and analyze the SQL-extracted data. Libraries like Pandas were used to manipulate data structures, and NumPy for numerical operations.
3. Data Visualization:
   The results of the analysis were then visualized using Python libraries such as Matplotlib, Seaborn, and Plotly. The goal was to create charts and graphs that clearly communicate the results in a visually appealing manner.



Configuration and Setup for Analysis

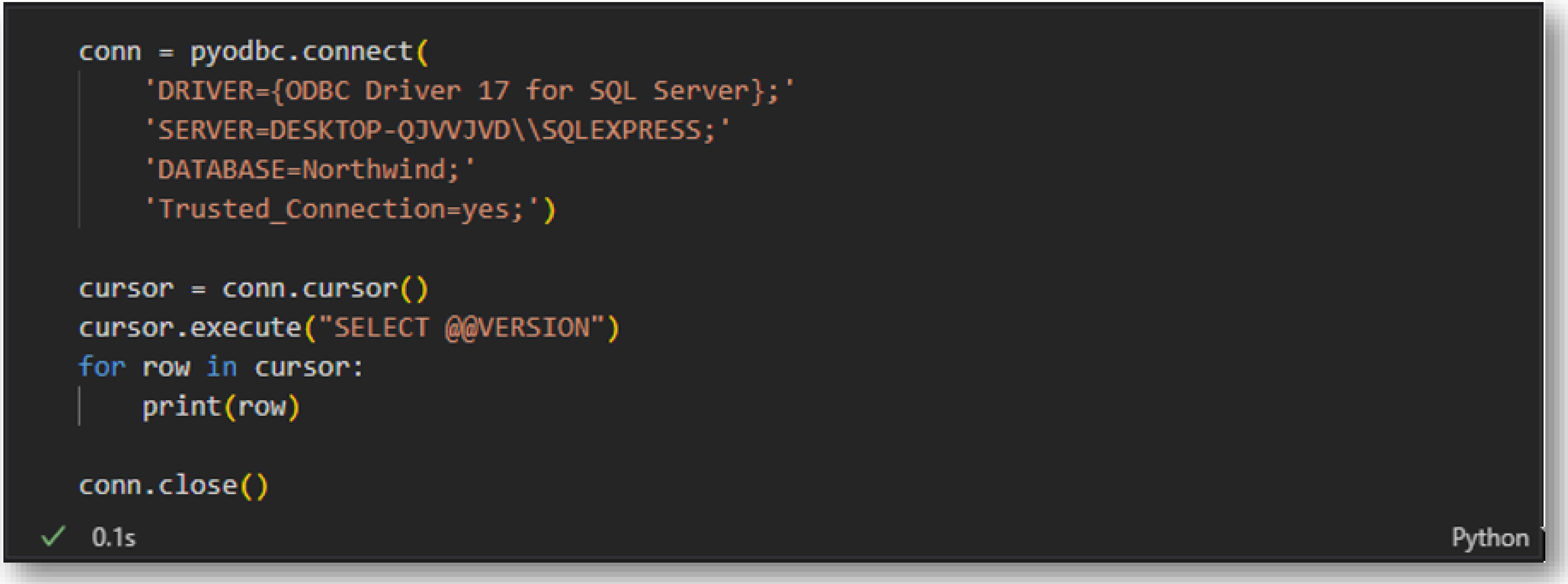
Preparing Libraries and Establishing Database Connection

Importing Necessary Python Libraries



```python
import pyodbc
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

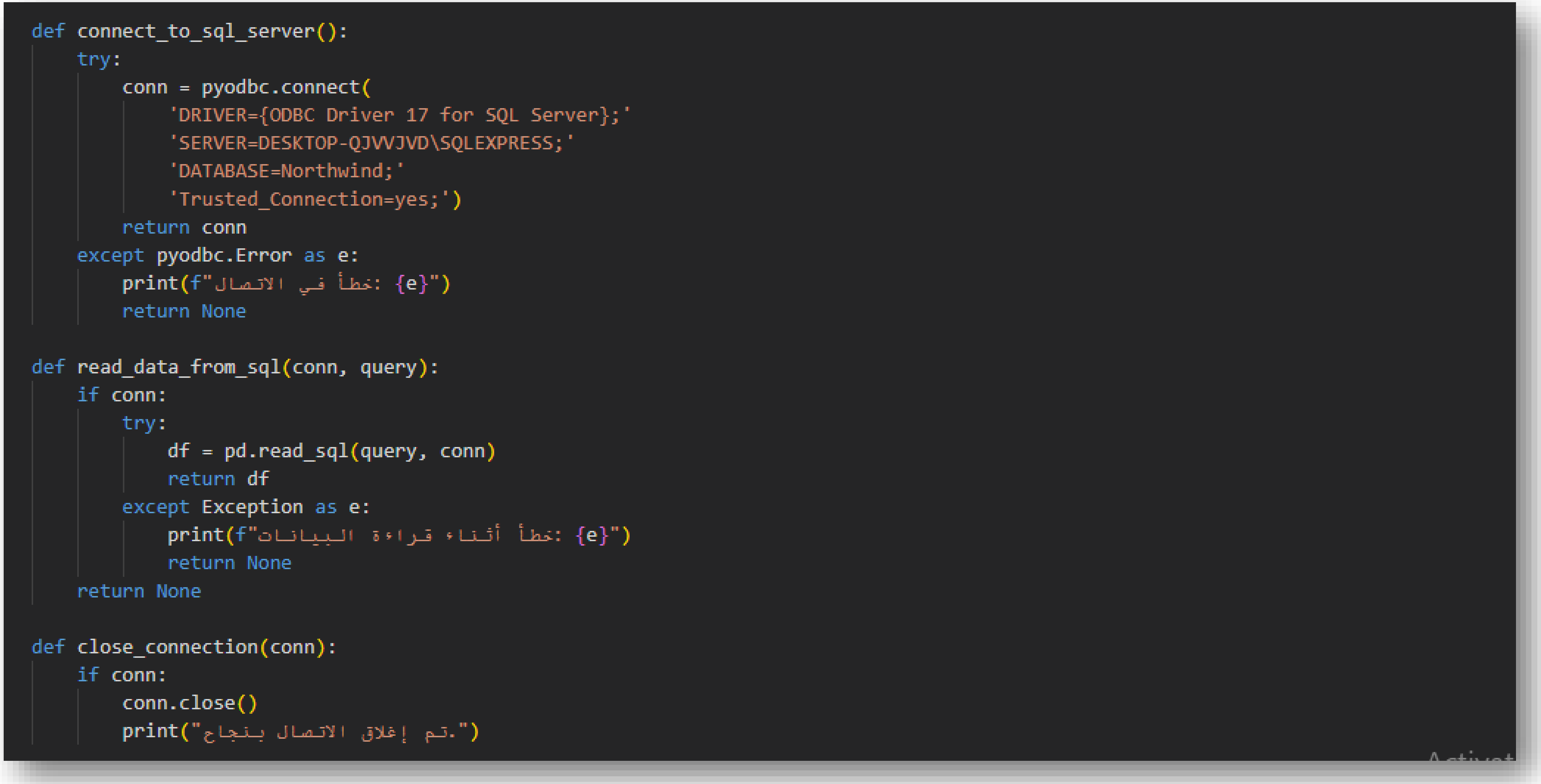
Connecting SQL Server with Python



```python
conn = pyodbc.connect(
        'DRIVER={ODBC Driver 17 for SQL Server};'
        'SERVER=DESKTOP-QJVVJVD\\SQLEXPRESS;'
        'DATABASE=Northwind;'
        'Trusted_Connection=yes;')

cursor = conn.cursor()
cursor.execute("SELECT @@VERSION")
for row in cursor:
    print(row)

conn.close()
```

✓ 0.1s Python

Connecting SQL Server with Python (Cont’d)



```python
def connect_to_sql_server():
    try:
        conn = pyodbc.connect(
            'DRIVER={ODBC Driver 17 for SQL Server};'
            'SERVER=DESKTOP-QJVVJVD\SQLEXPRESS;'
            'DATABASE=Northwind;'
            'Trusted_Connection=yes;')
        return conn
    except pyodbc.Error as e:
        print(f"خطأ في الاتصال: {e}")
        return None

def read_data_from_sql(conn, query):
    if conn:
        try:
            df = pd.read_sql(query, conn)
            return df
        except Exception as e:
            print(f"خطأ أثناء قراءة البيانات: {e}")
            return None
    return None

def close_connection(conn):
    if conn:
        conn.close()
        print("تم إغلاق الاتصال بنجاح.")
```

Business Questions

Key Business Questions Addressed

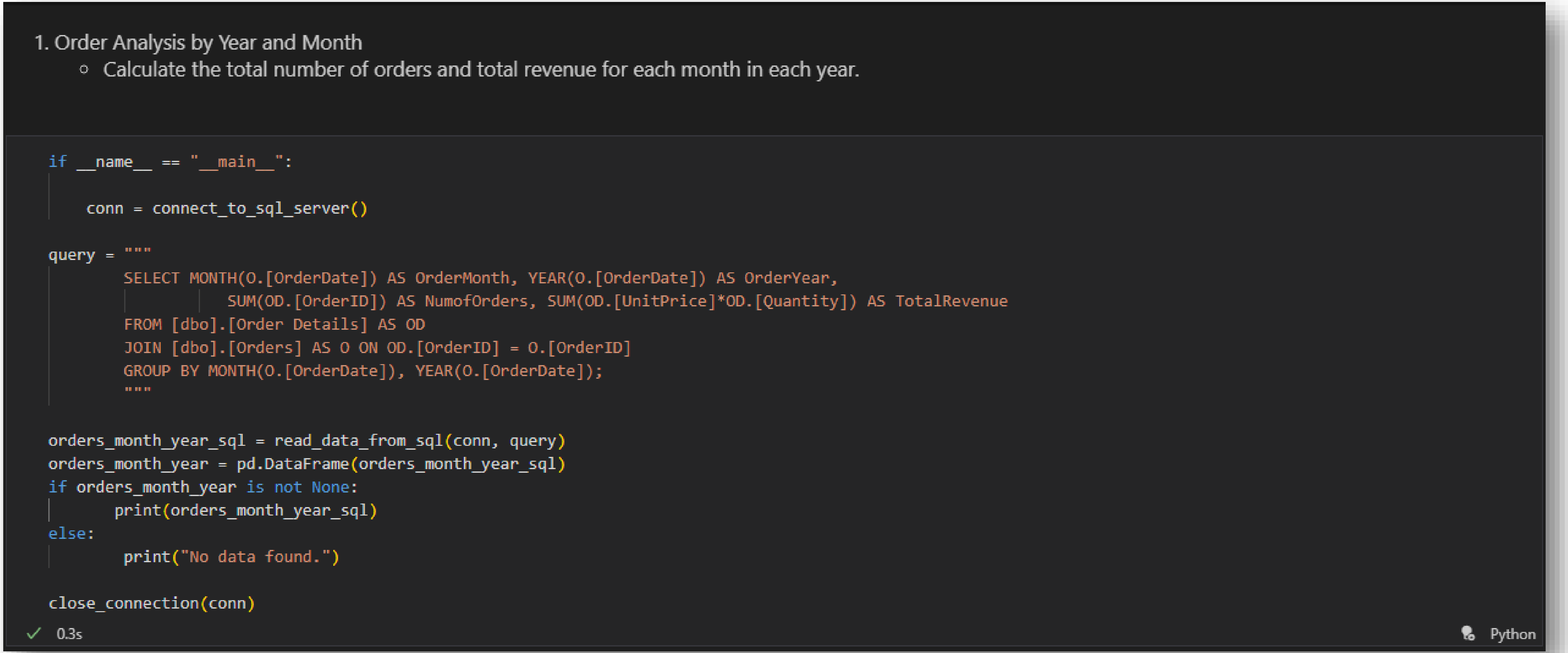
Question 1

Order Analysis by Year and Month

Calculate the total number of orders and total revenue for each month in each year.

Order Analysis by Year and Month

SQL Query & Python Code



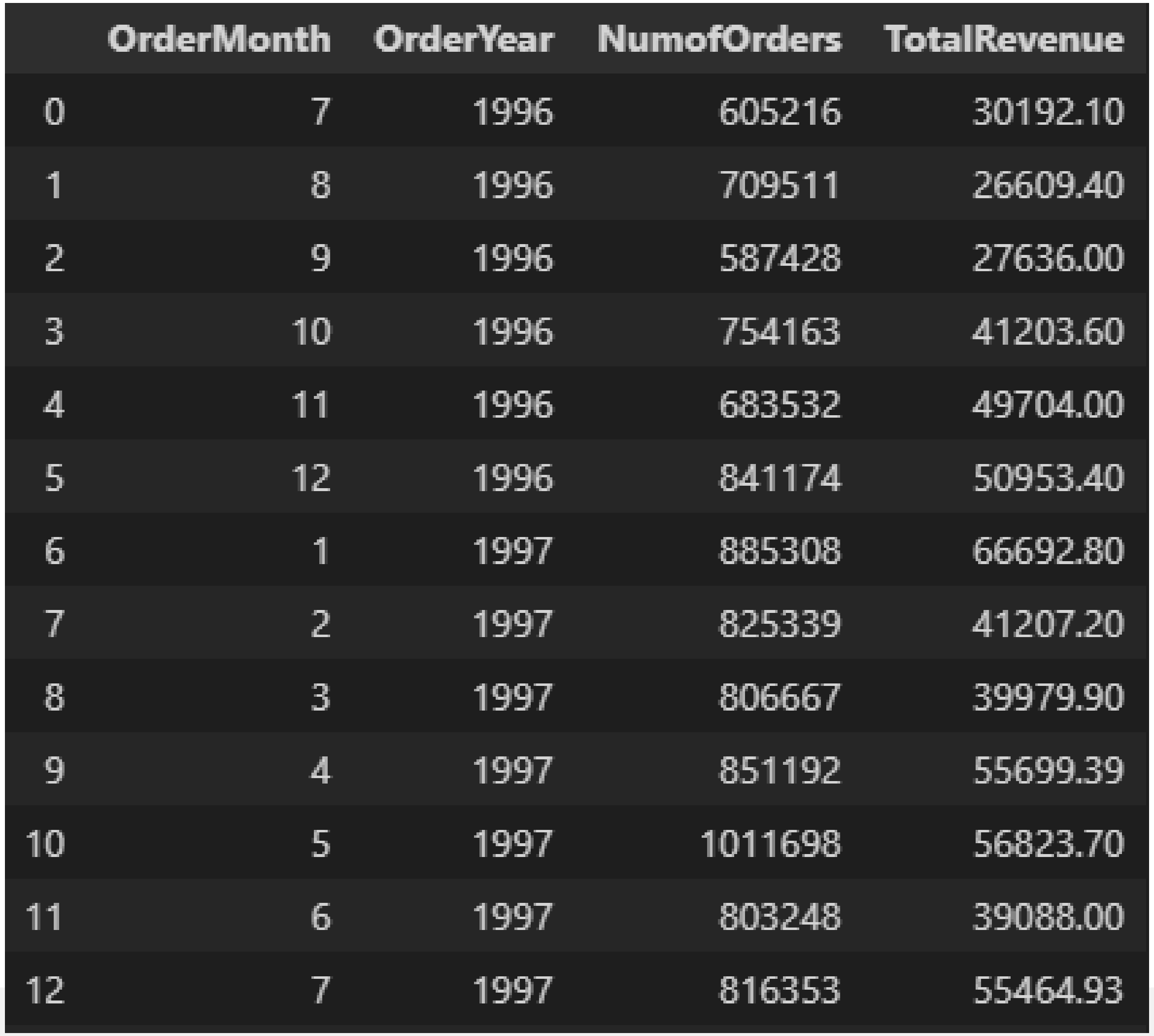
1. Order Analysis by Year and Month
   - Calculate the total number of orders and total revenue for each month in each year.

```
if __name__ == "__main__":

    conn = connect_to_sql_server()

query = """
        SELECT MONTH(O.[OrderDate]) AS OrderMonth, YEAR(O.[OrderDate]) AS OrderYear,
                   SUM(OD.[OrderID]) AS NumofOrders, SUM(OD.[UnitPrice]*OD.[Quantity]) AS TotalRevenue
        FROM [dbo].[Order Details] AS OD
        JOIN [dbo].[Orders] AS O ON OD.[OrderID] = O.[OrderID]
        GROUP BY MONTH(O.[OrderDate]), YEAR(O.[OrderDate]);
        """

orders_month_year_sql = read_data_from_sql(conn, query)
orders_month_year = pd.DataFrame(orders_month_year_sql)
if orders_month_year is not None:
       print(orders_month_year_sql)
else:
        print("No data found.")

close_connection(conn)
```

✓ 0.3s

Order Analysis by Year and Month

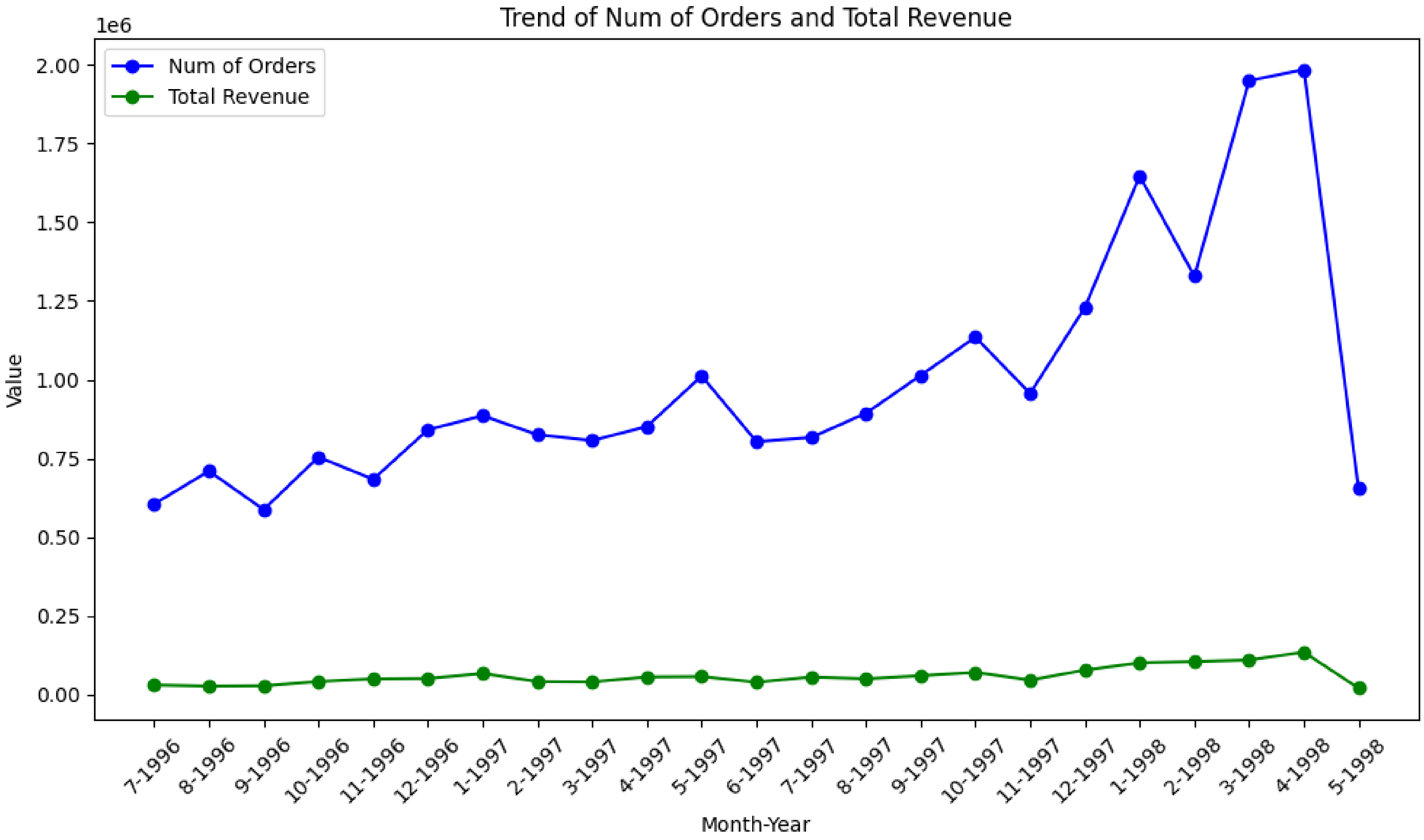
Subset of the Data Generated



	OrderMonth	OrderYear	NumofOrders	TotalRevenue
0	7	1996	605216	30192.10
1	8	1996	709511	26609.40
2	9	1996	587428	27636.00
3	10	1996	754163	41203.60
4	11	1996	683532	49704.00
5	12	1996	841174	50953.40
6	1	1997	885308	66692.80
7	2	1997	825339	41207.20
8	3	1997	806667	39979.90
9	4	1997	851192	55699.39
10	5	1997	1011698	56823.70
11	6	1997	803248	39088.00
12	7	1997	816353	55464.93

Order Analysis by Year and Month

Data Visualization



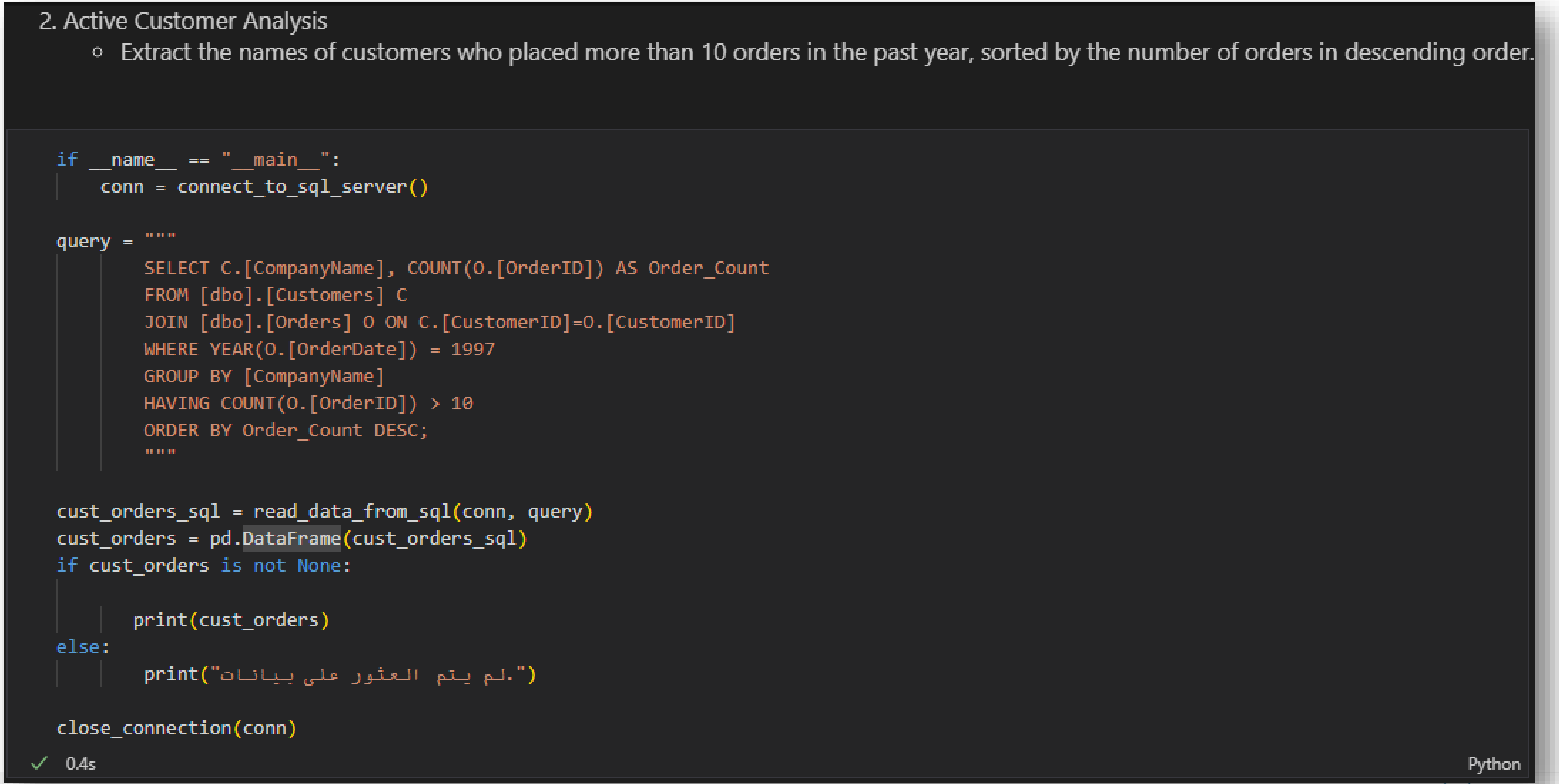
Question 2

Active Customer Analysis

Extract the names of customers who placed more than 10 orders in the past year, sorted by the number of orders in descending order.

Active Customer Analysis

SQL Query & Python Code



2. Active Customer Analysis

   - Extract the names of customers who placed more than 10 orders in the past year, sorted by the number of orders in descending order.

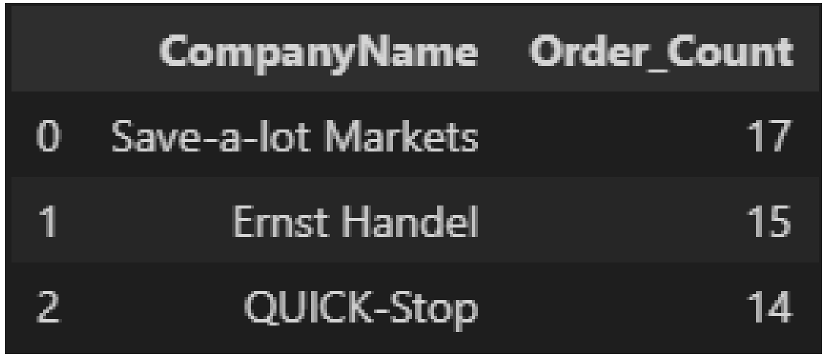
```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT C.[CompanyName], COUNT(O.[OrderID]) AS Order_Count
        FROM [dbo].[Customers] C
        JOIN [dbo].[Orders] O ON C.[CustomerID]=O.[CustomerID]
        WHERE YEAR(O.[OrderDate]) = 1997
        GROUP BY [CompanyName]
        HAVING COUNT(O.[OrderID]) > 10
        ORDER BY Order_Count DESC;
        """

cust_orders_sql = read_data_from_sql(conn, query)
cust_orders = pd.DataFrame(cust_orders_sql)
if cust_orders is not None:

       print(cust_orders)
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.4s

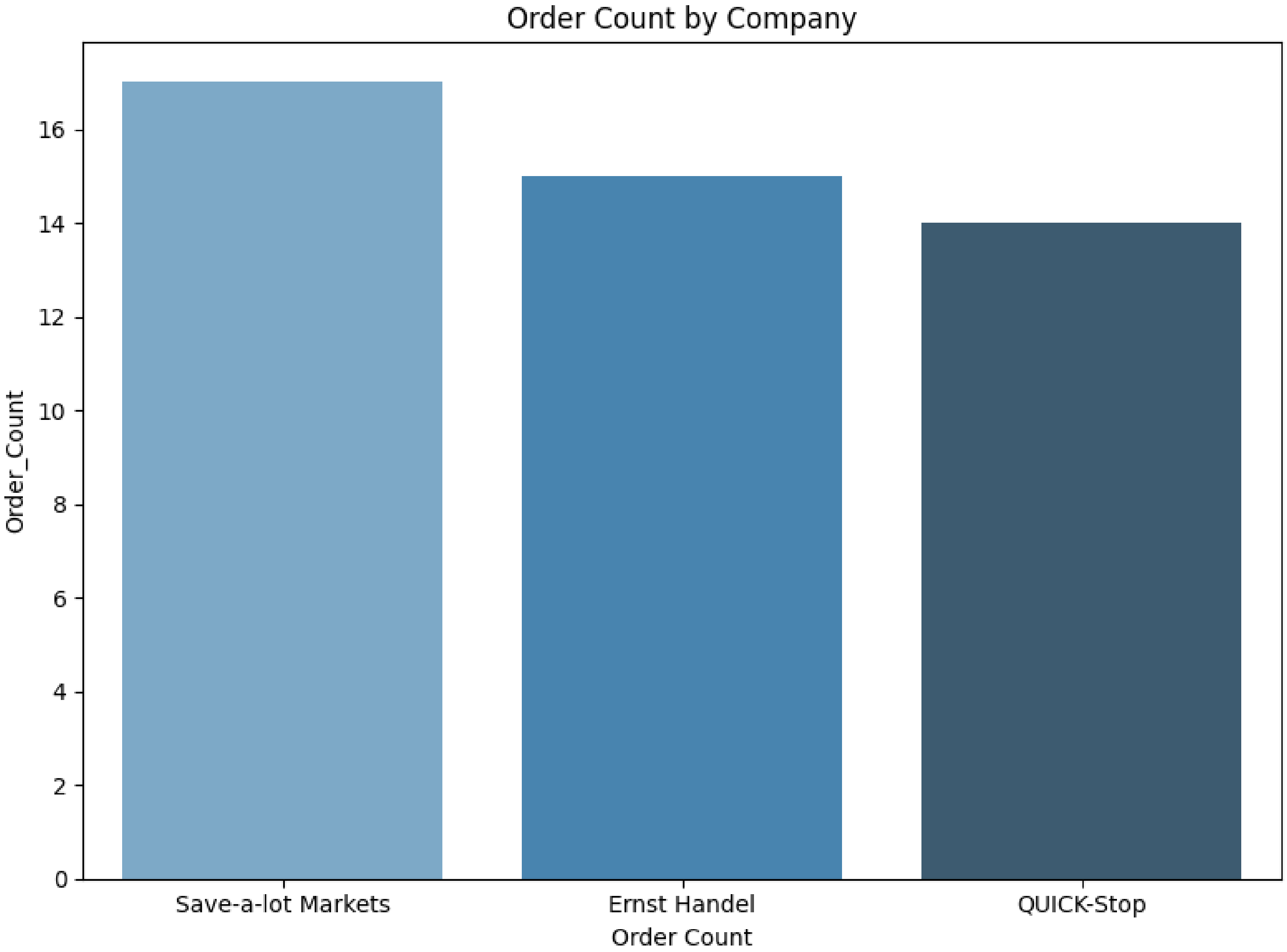
Python

Active Customer Analysis

Data Generated



	CompanyName	Order_Count
0	Save-a-lot Markets	17
1	Ernst Handel	15
2	QUICK-Stop	14



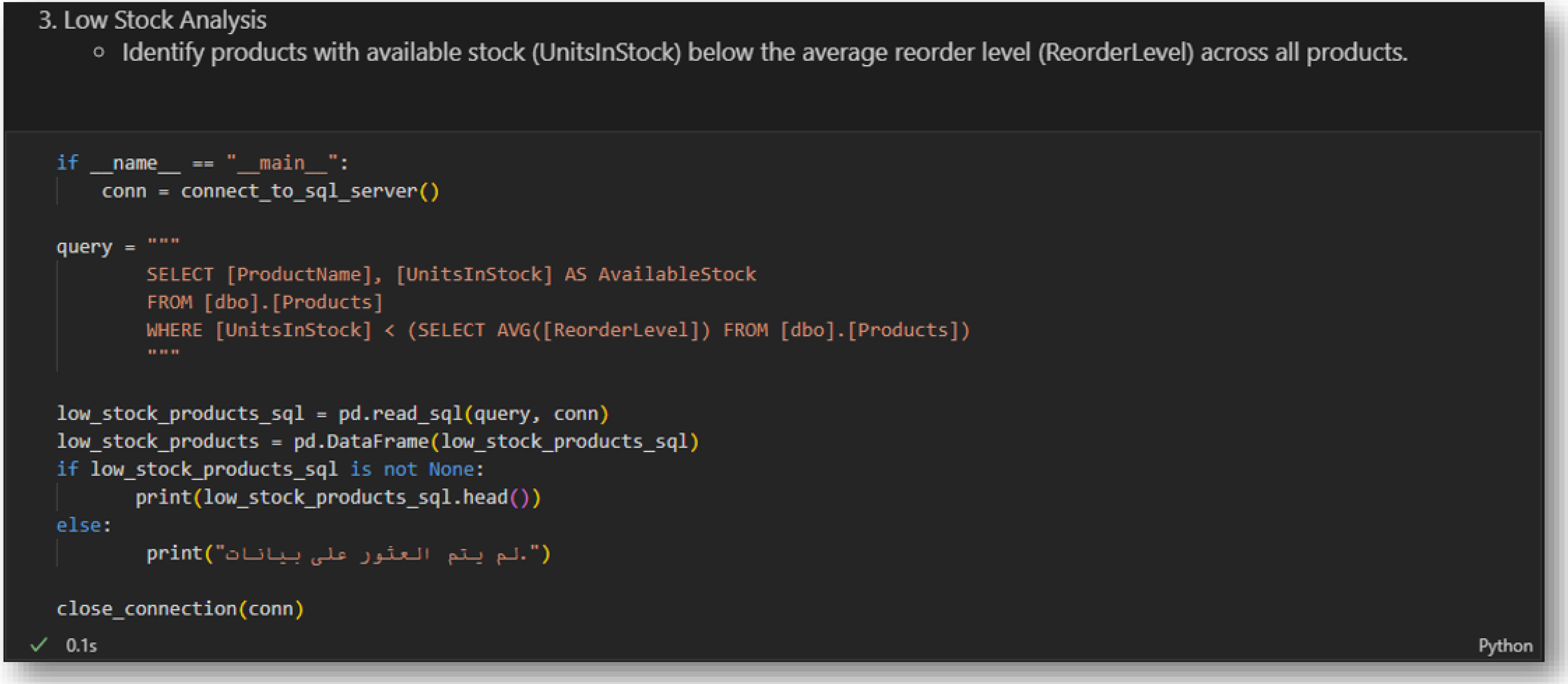
Question 3

Low Stock Analysis

Identify products with available stock (UnitsInStock) below the average reorder level (ReorderLevel) across all products.

Low Stock Analysis

SQL Query & Python Code



3. Low Stock Analysis
    - Identify products with available stock (UnitsInStock) below the average reorder level (ReorderLevel) across all products.

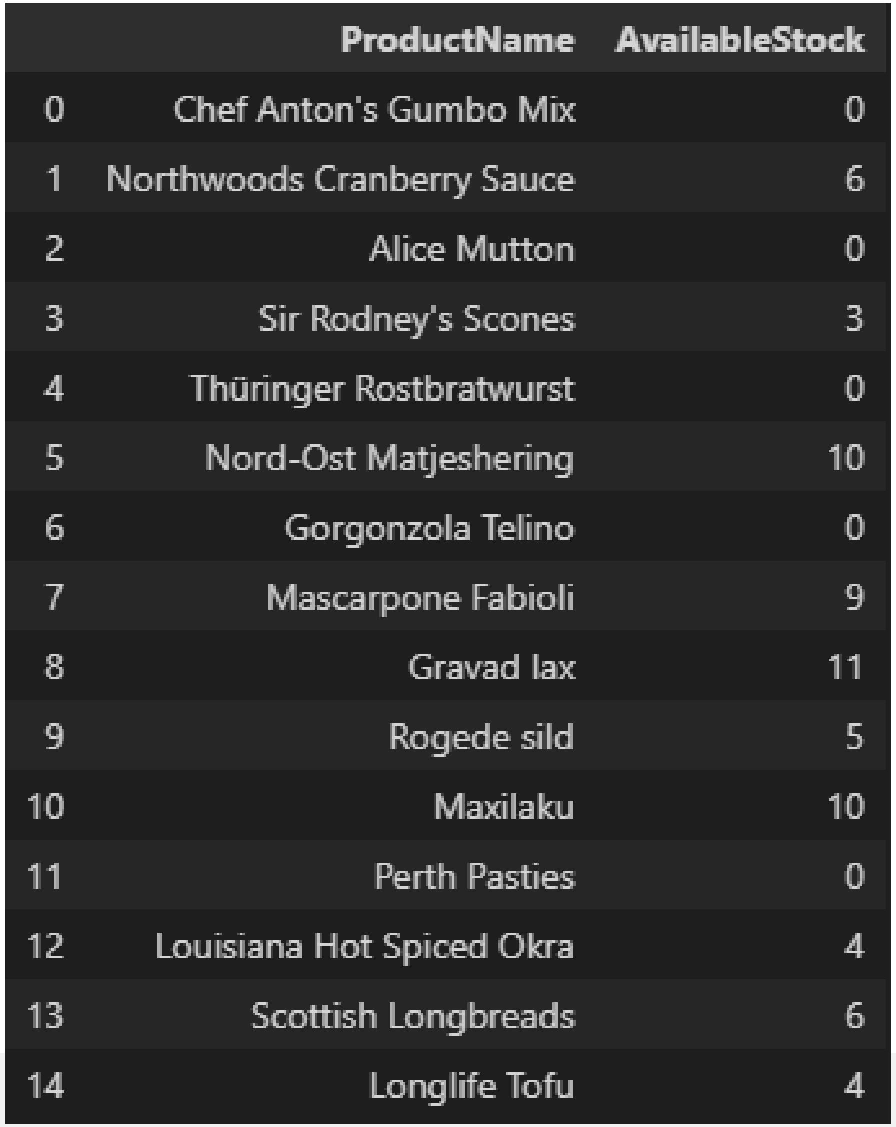
```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT [ProductName], [UnitsInStock] AS AvailableStock
        FROM [dbo].[Products]
        WHERE [UnitsInStock] < (SELECT AVG([ReorderLevel]) FROM [dbo].[Products])
        """

low_stock_products_sql = pd.read_sql(query, conn)
low_stock_products = pd.DataFrame(low_stock_products_sql)
if low_stock_products_sql is not None:
       print(low_stock_products_sql.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.1s

Python

Low Stock Analysis

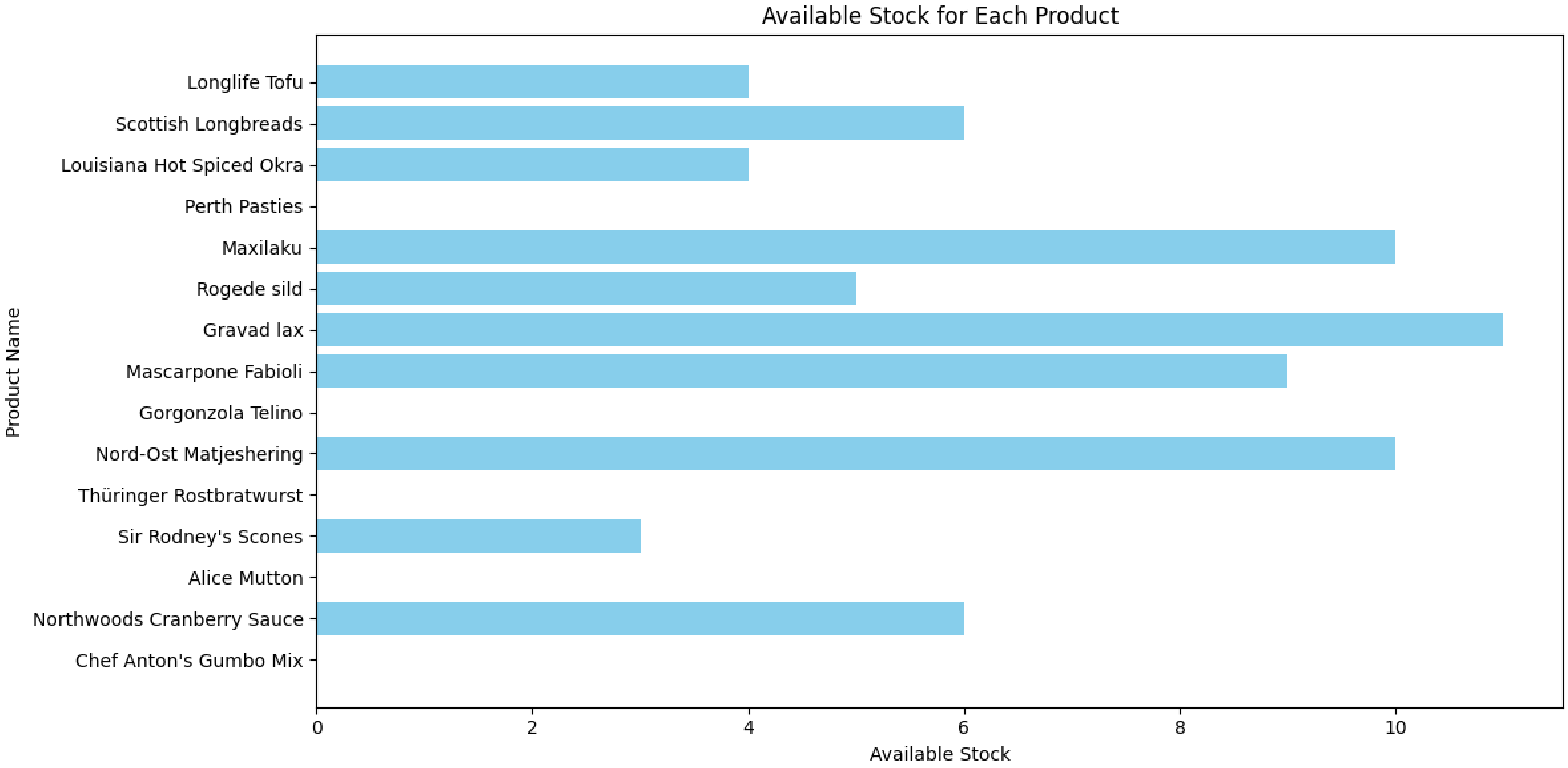
Data Generated



	ProductName	AvailableStock
0	Chef Anton's Gumbo Mix	0
1	Northwoods Cranberry Sauce	6
2	Alice Mutton	0
3	Sir Rodney's Scones	3
4	Thüringer Rostbratwurst	0
5	Nord-Ost Matjeshering	10
6	Gorgonzola Telino	0
7	Mascarpone Fabioli	9
8	Gravad lax	11
9	Rogede sild	5
10	Maxilaku	10
11	Perth Pasties	0
12	Louisiana Hot Spiced Okra	4
13	Scottish Longbreads	6
14	Longlife Tofu	4

Low Stock Analysis

Data Visualization



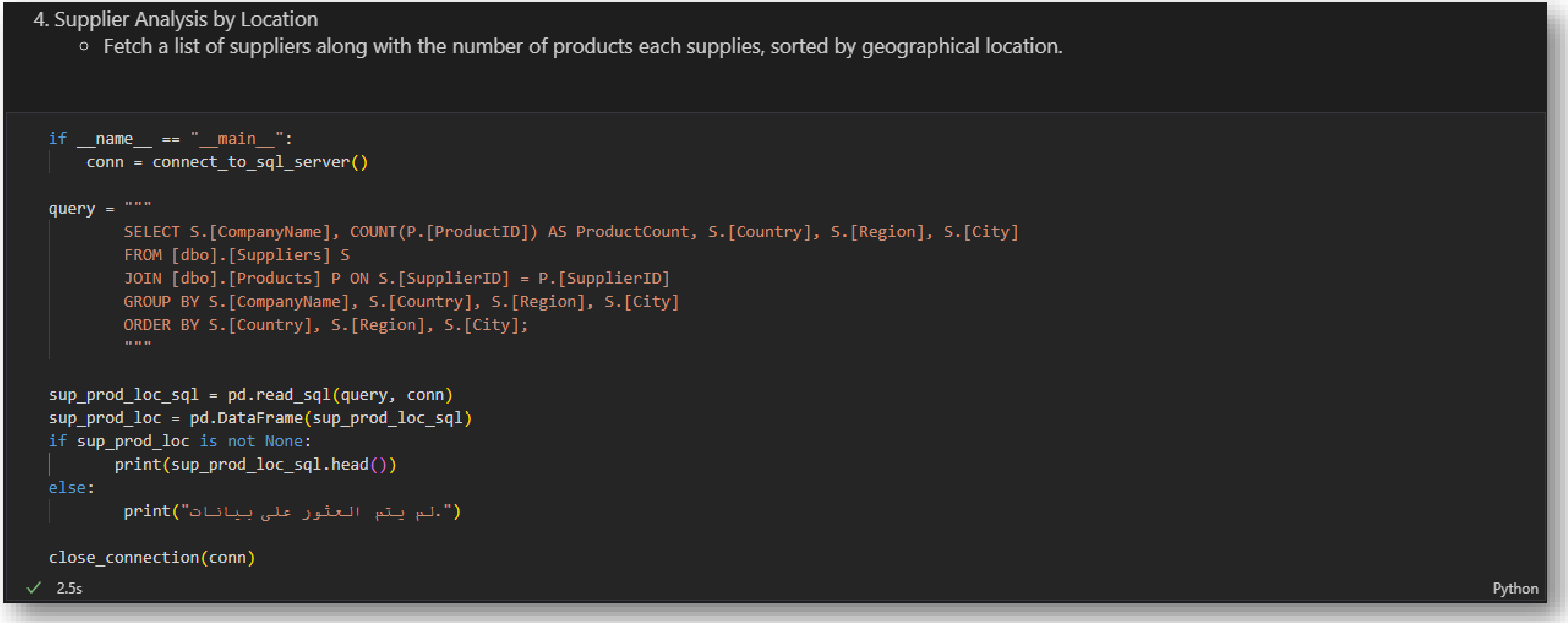
Question 4

Supplier Analysis by Location

Fetch a list of suppliers along with the number of products each supplies, sorted by geographical Location.

Supplier Analysis by Location

SQL Query & Python Code

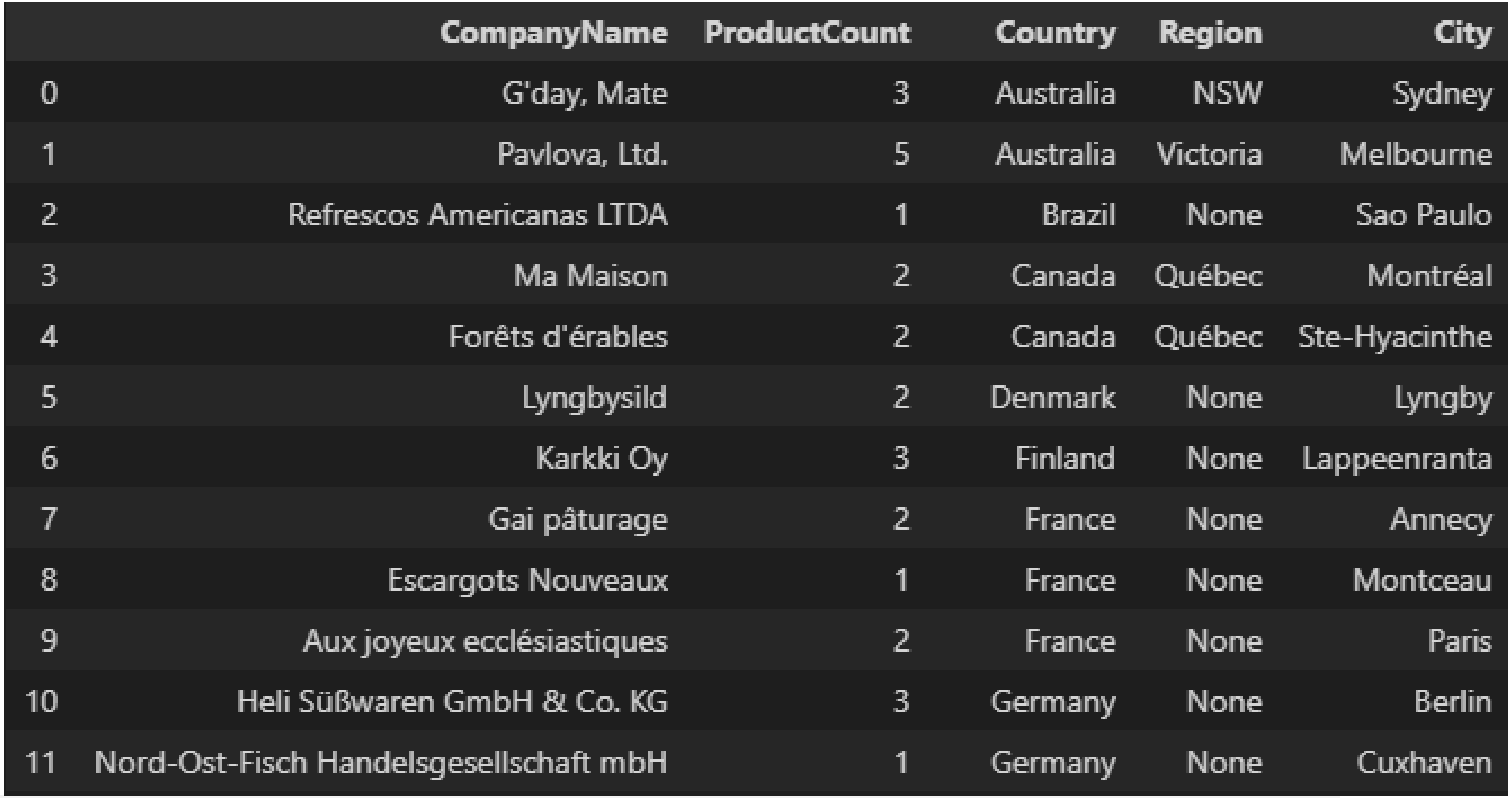


4. Supplier Analysis by Location
    - Fetch a list of suppliers along with the number of products each supplies, sorted by geographical location.

```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT S.[CompanyName], COUNT(P.[ProductID]) AS ProductCount, S.[Country], S.[Region], S.[City]
        FROM [dbo].[Suppliers] S
        JOIN [dbo].[Products] P ON S.[SupplierID] = P.[SupplierID]
        GROUP BY S.[CompanyName], S.[Country], S.[Region], S.[City]
        ORDER BY S.[Country], S.[Region], S.[City];
        """

sup_prod_loc_sql = pd.read_sql(query, conn)
sup_prod_loc = pd.DataFrame(sup_prod_loc_sql)
if sup_prod_loc is not None:
       print(sup_prod_loc_sql.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Supplier Analysis by Location

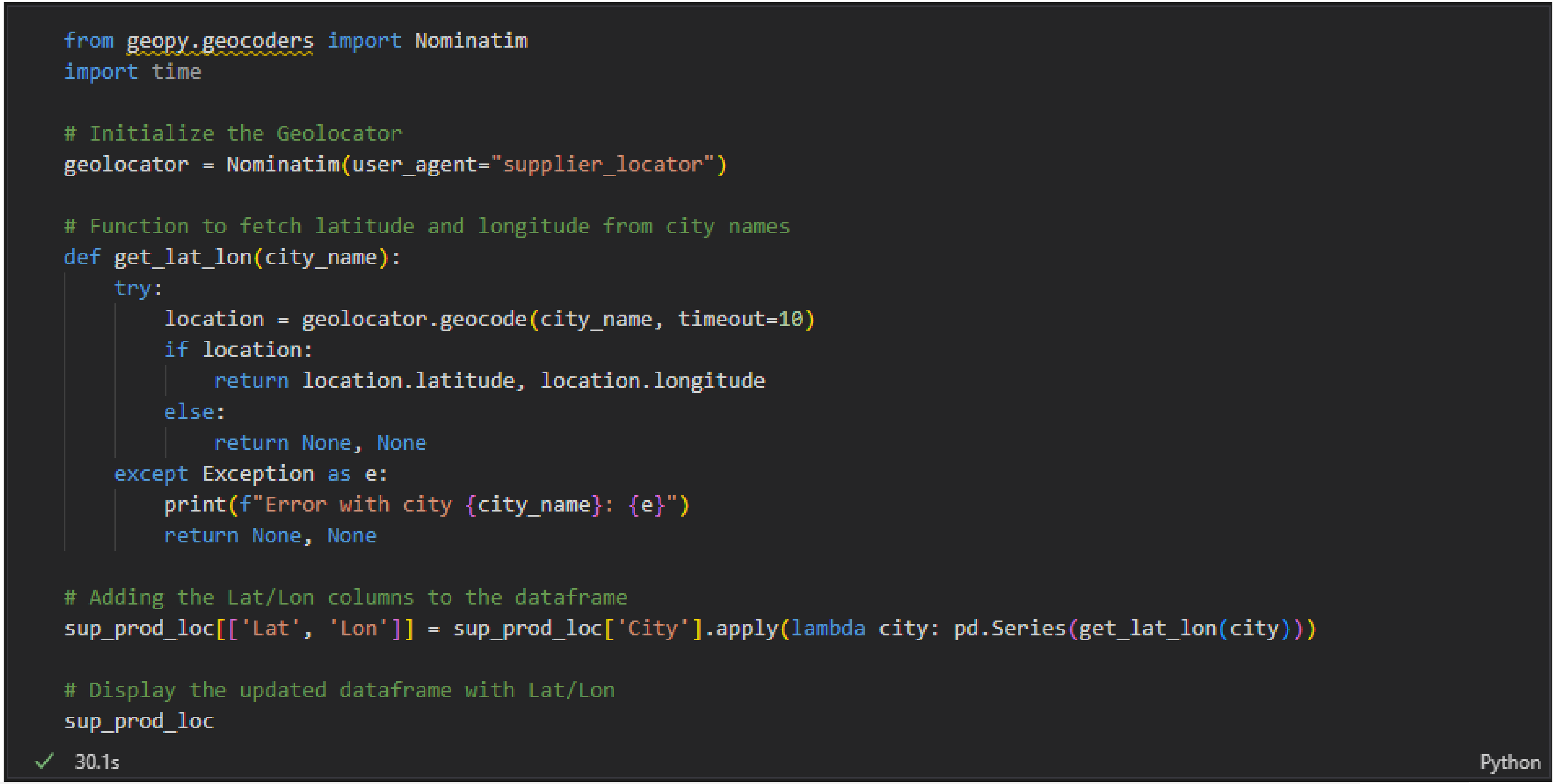
Subset of the Data Generated



	CompanyName	ProductCount	Country	Region	City
0	G'day, Mate	3	Australia	NSW	Sydney
1	Pavlova, Ltd.	5	Australia	Victoria	Melbourne
2	Refrescos Americanas LTDA	1	Brazil	None	Sao Paulo
3	Ma Maison	2	Canada	Québec	Montréal
4	Forêts d'érables	2	Canada	Québec	Ste-Hyacinthe
5	Lyngbysild	2	Denmark	None	Lyngby
6	Karkki Oy	3	Finland	None	Lappeenranta
7	Gai pâturage	2	France	None	Annecy
8	Escargots Nouveaux	1	France	None	Montceau
9	Aux joyeux ecclésiastiques	2	France	None	Paris
10	Heli Süßwaren GmbH & Co. KG	3	Germany	None	Berlin
11	Nord-Ost-Fisch Handelsgesellschaft mbH	1	Germany	None	Cuxhaven

Supplier Analysis by Location

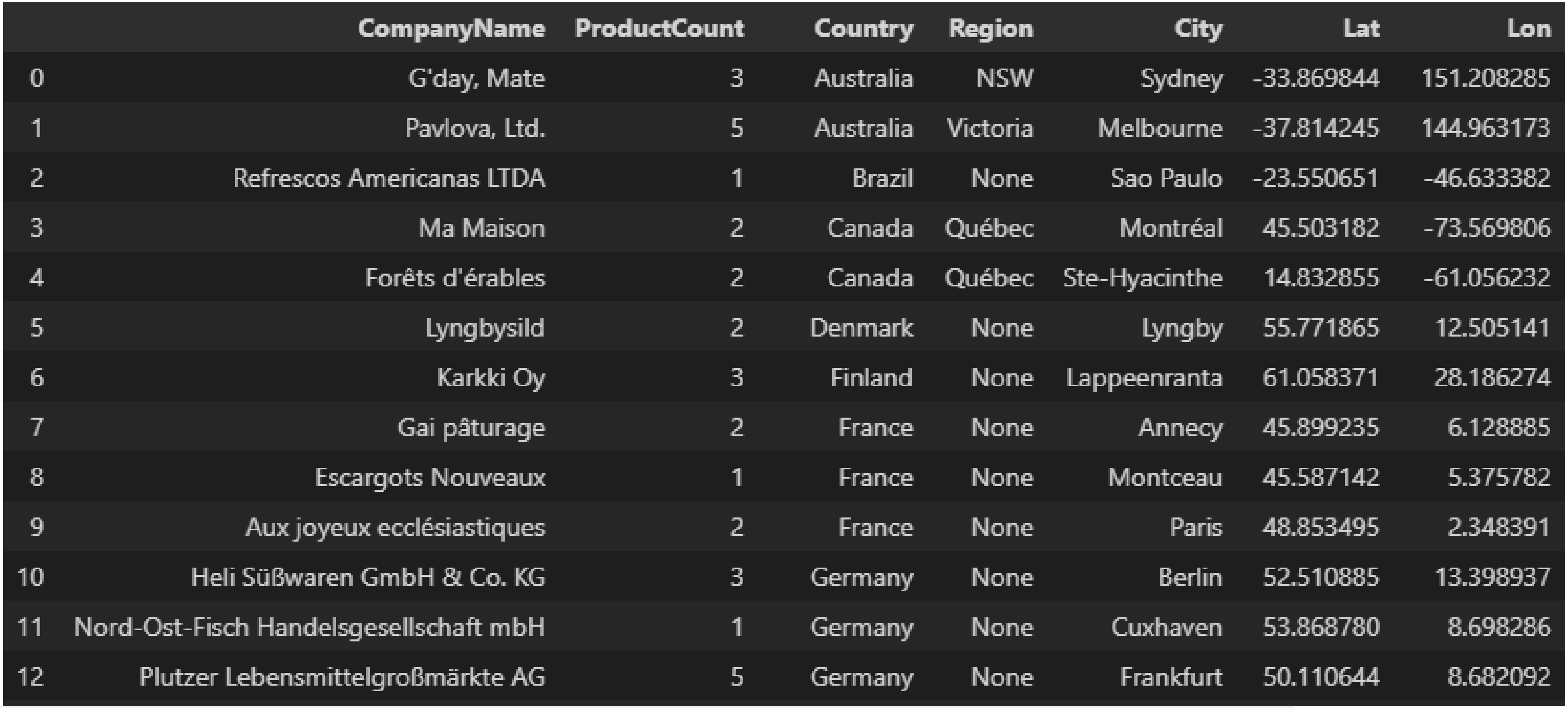
Add Location Coordinates for ALL Cities



```python
from geopy.geocoders import Nominatim
import time

# Initialize the Geolocator
geolocator = Nominatim(user_agent="supplier_locator")

# Function to fetch latitude and longitude from city names
def get_lat_lon(city_name):
    try:
        location = geolocator.geocode(city_name, timeout=10)
        if location:
            return location.latitude, location.longitude
        else:
            return None, None
    except Exception as e:
        print(f"Error with city {city_name}: {e}")
        return None, None

# Adding the Lat/Lon columns to the dataframe
sup_prod_loc[['Lat', 'Lon']] = sup_prod_loc['City'].apply(lambda city: pd.Series(get_lat_lon(city)))

# Display the updated dataframe with Lat/Lon
sup_prod_loc
```

Supplier Analysis by Location

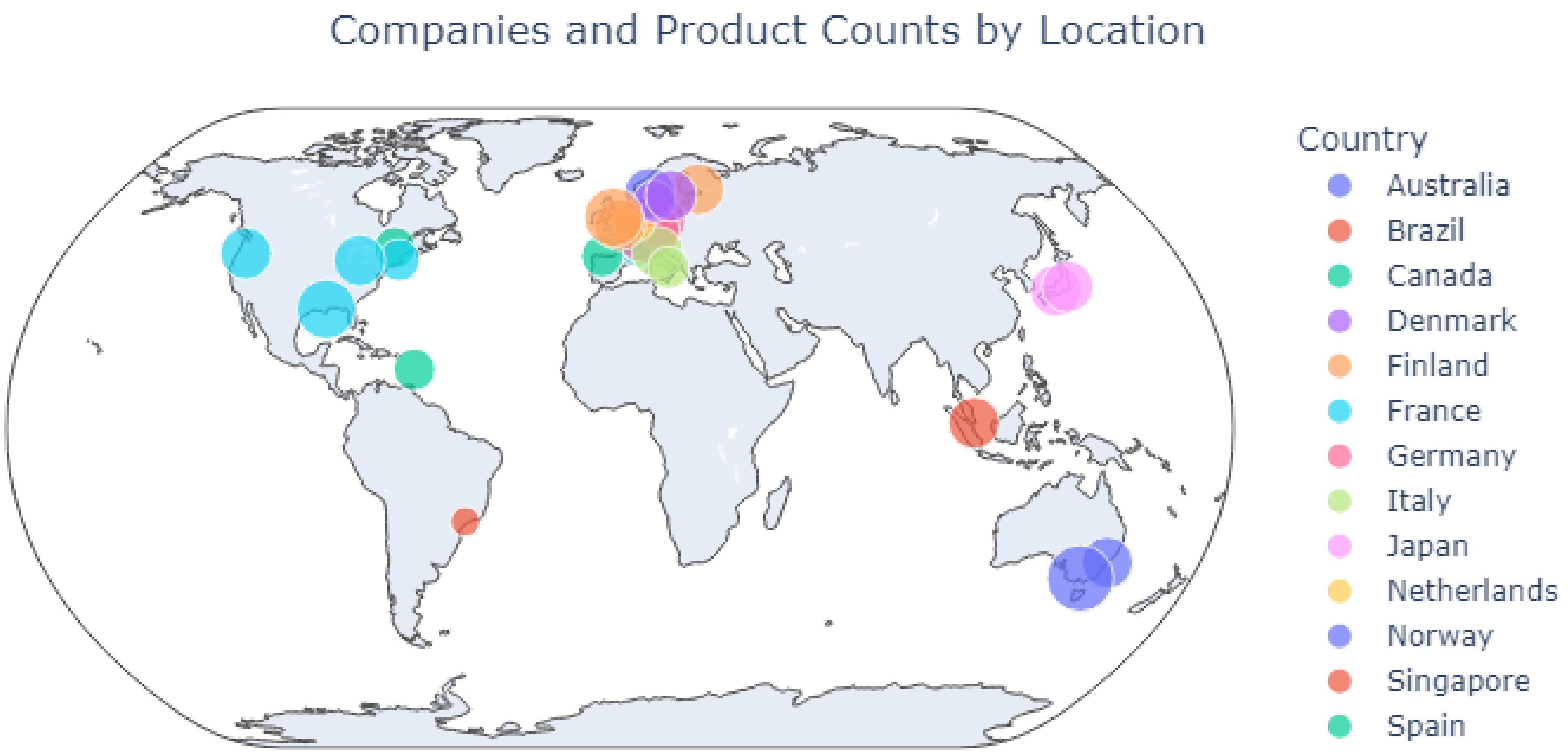
Portion of the Updated Data



	CompanyName	ProductCount	Country	Region	City	Lat	Lon
0	G'day, Mate	3	Australia	NSW	Sydney	-33.869844	151.208285
1	Pavlova, Ltd.	5	Australia	Victoria	Melbourne	-37.814245	144.963173
2	Refrescos Americanas LTDA	1	Brazil	None	Sao Paulo	-23.550651	-46.633382
3	Ma Maison	2	Canada	Québec	Montréal	45.503182	-73.569806
4	Forêts d'érables	2	Canada	Québec	Ste-Hyacinthe	14.832855	-61.056232
5	Lyngbysild	2	Denmark	None	Lyngby	55.771865	12.505141
6	Karkki Oy	3	Finland	None	Lappeenranta	61.058371	28.186274
7	Gai pâturage	2	France	None	Annecy	45.899235	6.128885
8	Escargots Nouveaux	1	France	None	Montceau	45.587142	5.375782
9	Aux joyeux ecclésiastiques	2	France	None	Paris	48.853495	2.348391
10	Heli Süßwaren GmbH & Co. KG	3	Germany	None	Berlin	52.510885	13.398937
11	Nord-Ost-Fisch Handelsgesellschaft mbH	1	Germany	None	Cuxhaven	53.868780	8.698286
12	Plutzer Lebensmittelgroßmärkte AG	5	Germany	None	Frankfurt	50.110644	8.682092

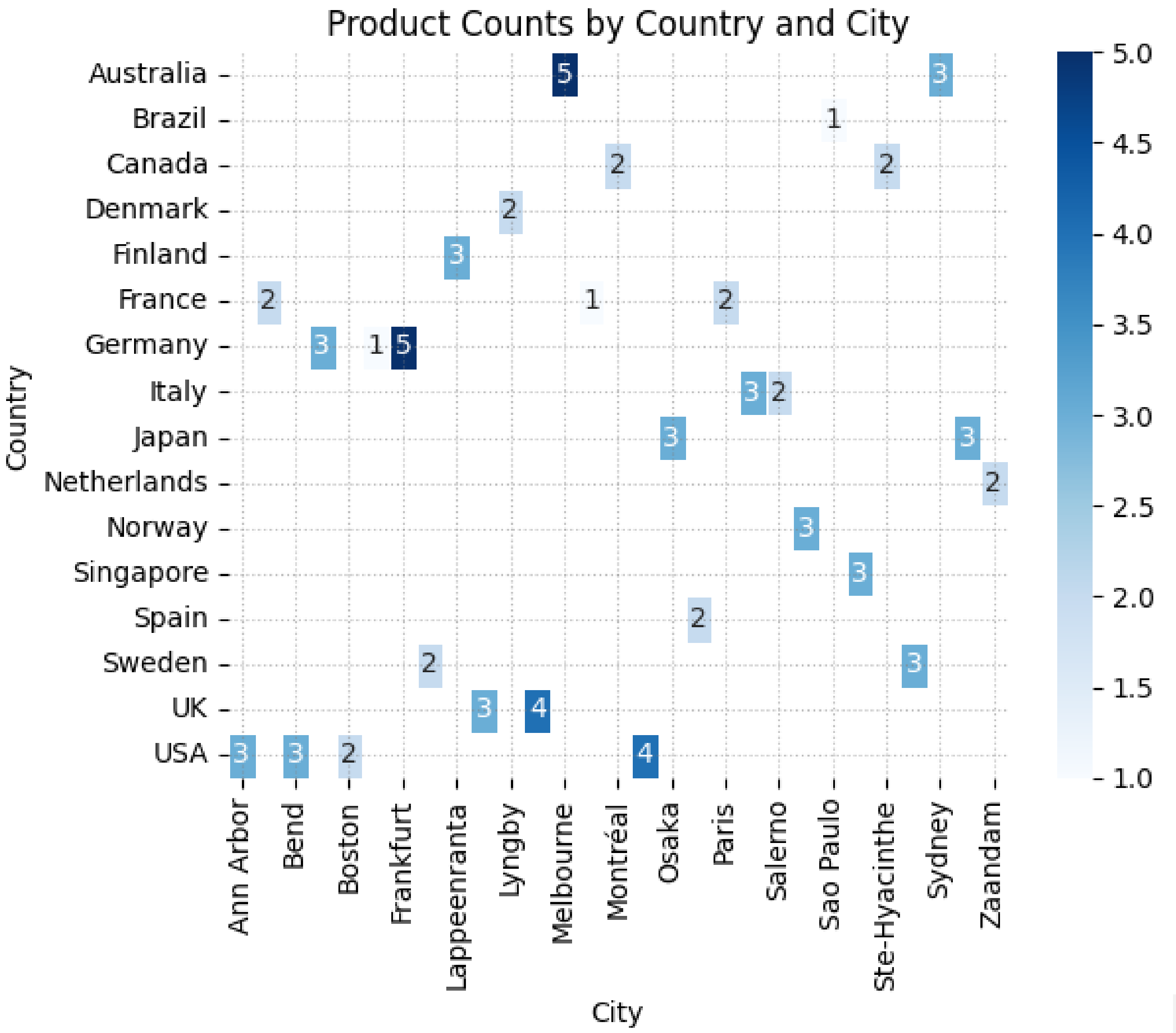
Supplier Analysis by Location

Data Visualization



Supplier Analysis by Location

Data Visualization



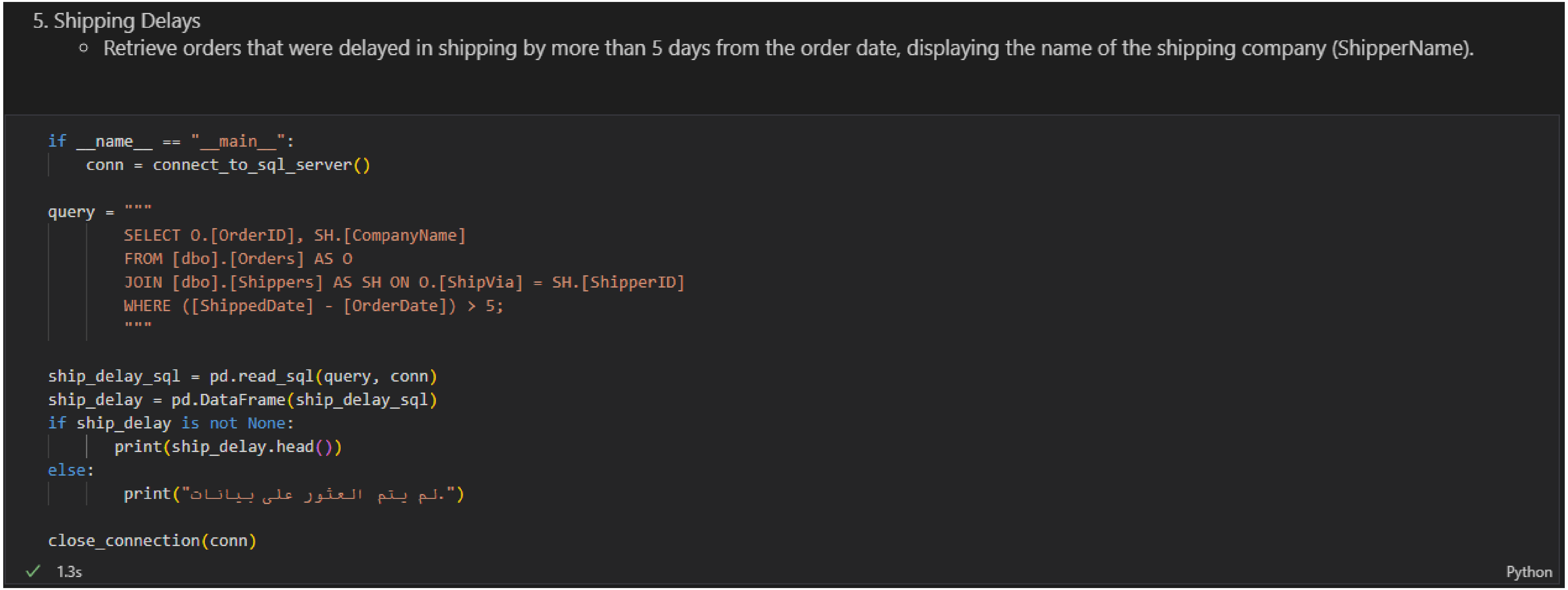
Question 5

Shipping Delays

Retrieve orders that were delayed in shipping by more than 5 days from the order date, displaying the name of the shipping company (ShipperName).

Shipping Delays

SQL Query & Python Code

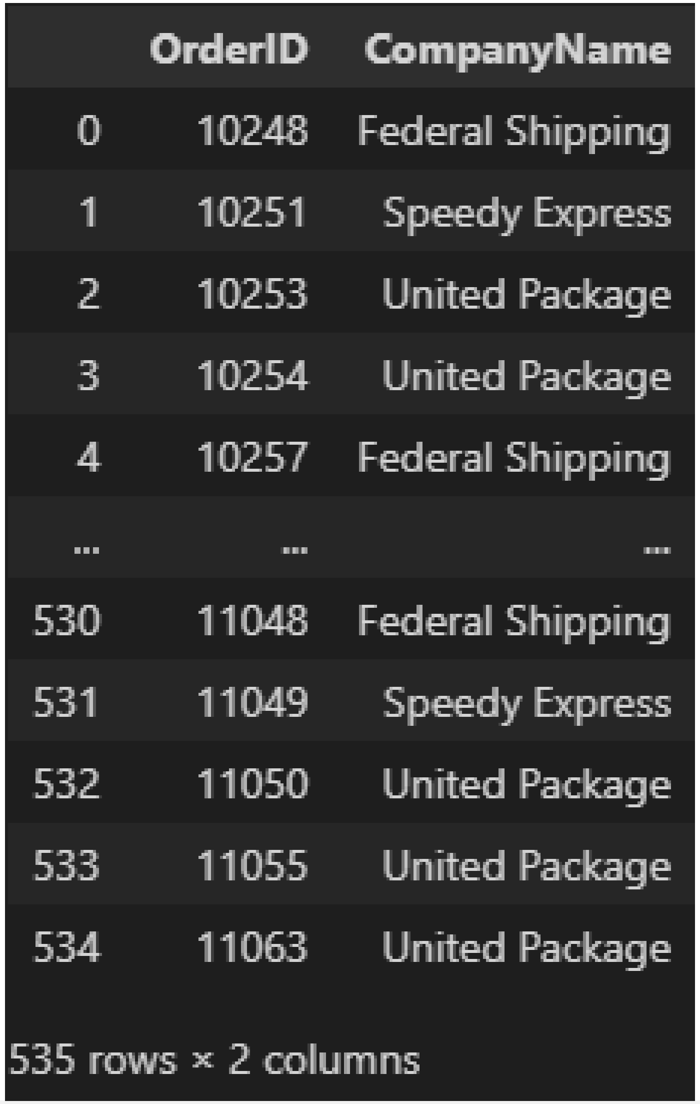


5. Shipping Delays
    - Retrieve orders that were delayed in shipping by more than 5 days from the order date, displaying the name of the shipping company (ShipperName).

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT O.[OrderID], SH.[CompanyName]
        FROM [dbo].[Orders] AS O
        JOIN [dbo].[Shippers] AS SH ON O.[ShipVia] = SH.[ShipperID]
        WHERE ([ShippedDate] - [OrderDate]) > 5;
        """

ship_delay_sql = pd.read_sql(query, conn)
ship_delay = pd.DataFrame(ship_delay_sql)
if ship_delay is not None:
       print(ship_delay.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Shipping Delays

Data Generated

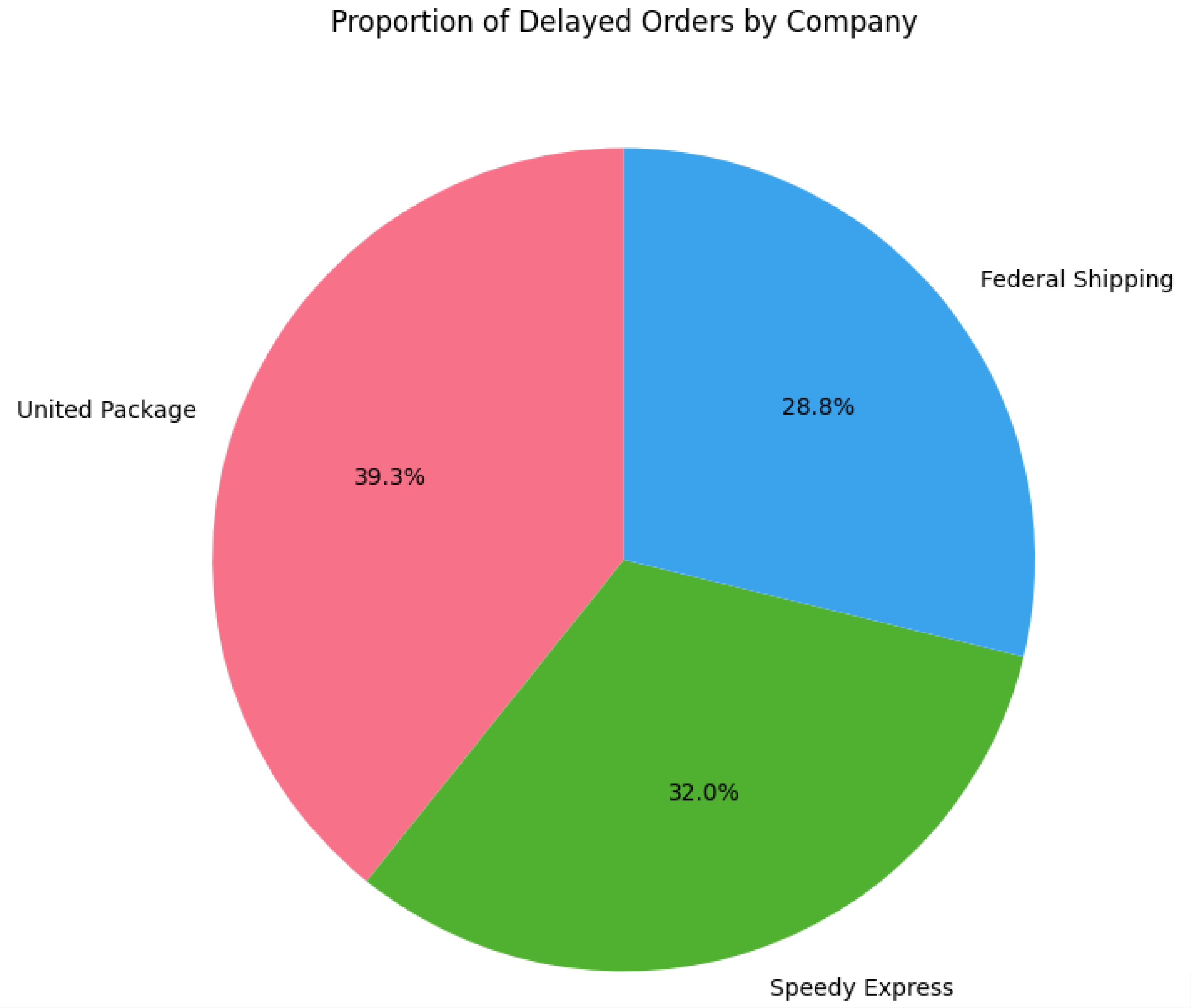


	OrderID	CompanyName
0	10248	Federal Shipping
1	10251	Speedy Express
2	10253	United Package
3	10254	United Package
4	10257	Federal Shipping
...	...	...
530	11048	Federal Shipping
531	11049	Speedy Express
532	11050	United Package
533	11055	United Package
534	11063	United Package

535 rows × 2 columns

Shipping Delays

Data Visualization of Order Proportions by Company



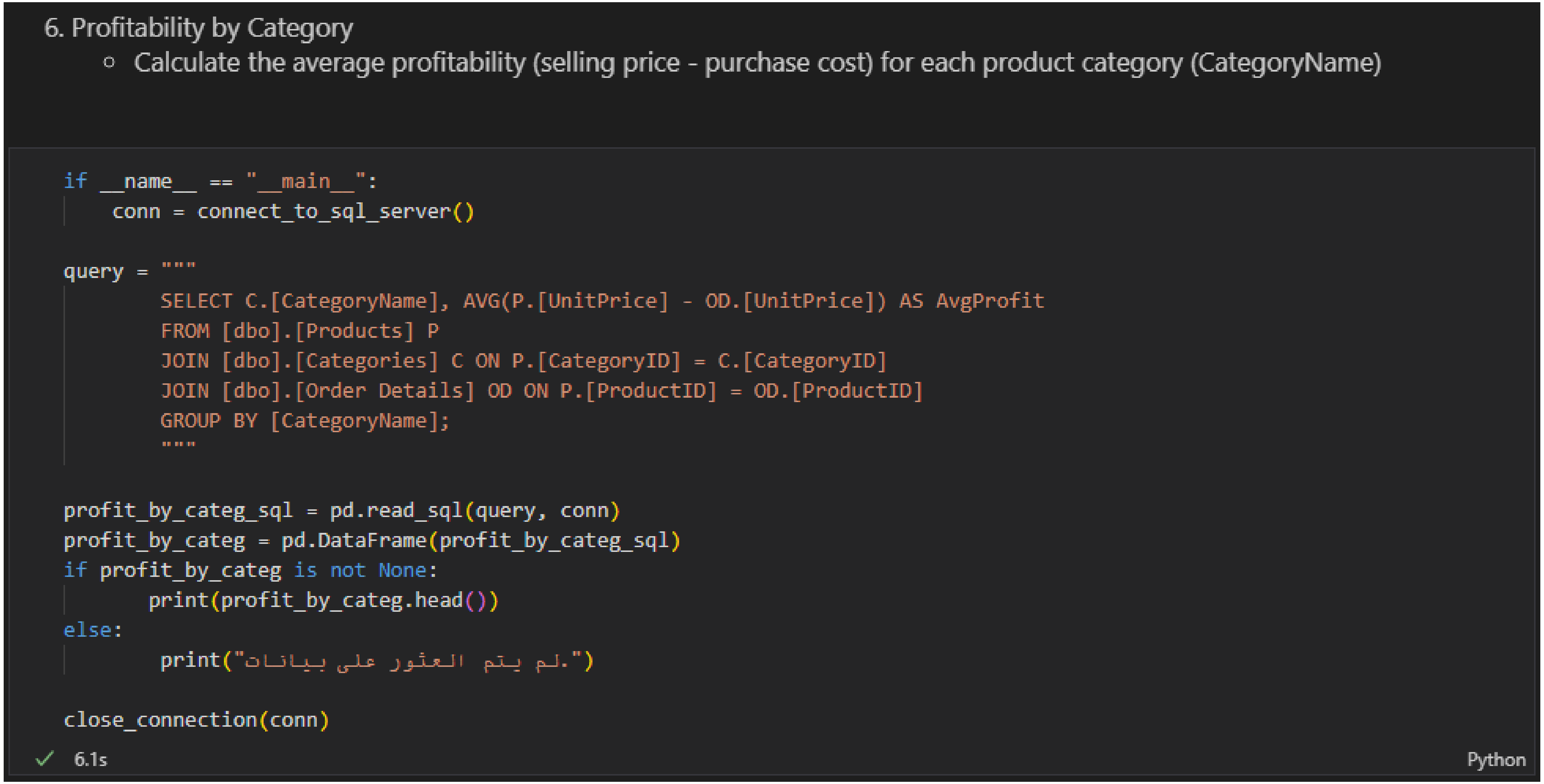
Question 6

Profitability by Category

Calculate the average profitability (selling price - purchase cost) for each product category (CategoryName).

Profitability by Category

SQL Query & Python Code



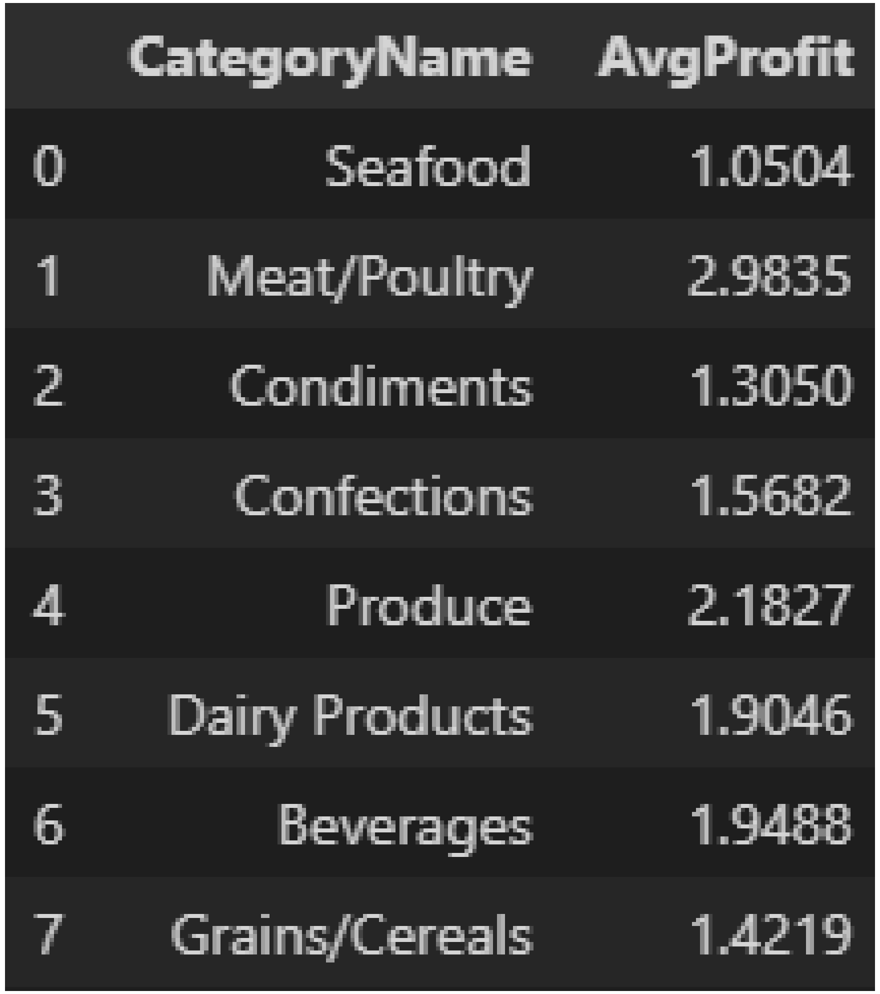
6. Profitability by Category
    - Calculate the average profitability (selling price - purchase cost) for each product category (CategoryName)

```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT C.[CategoryName], AVG(P.[UnitPrice] - OD.[UnitPrice]) AS AvgProfit
        FROM [dbo].[Products] P
        JOIN [dbo].[Categories] C ON P.[CategoryID] = C.[CategoryID]
        JOIN [dbo].[Order Details] OD ON P.[ProductID] = OD.[ProductID]
        GROUP BY [CategoryName];
        """

profit_by_categ_sql = pd.read_sql(query, conn)
profit_by_categ = pd.DataFrame(profit_by_categ_sql)
if profit_by_categ is not None:
       print(profit_by_categ.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 6.1s

Profitability by Category

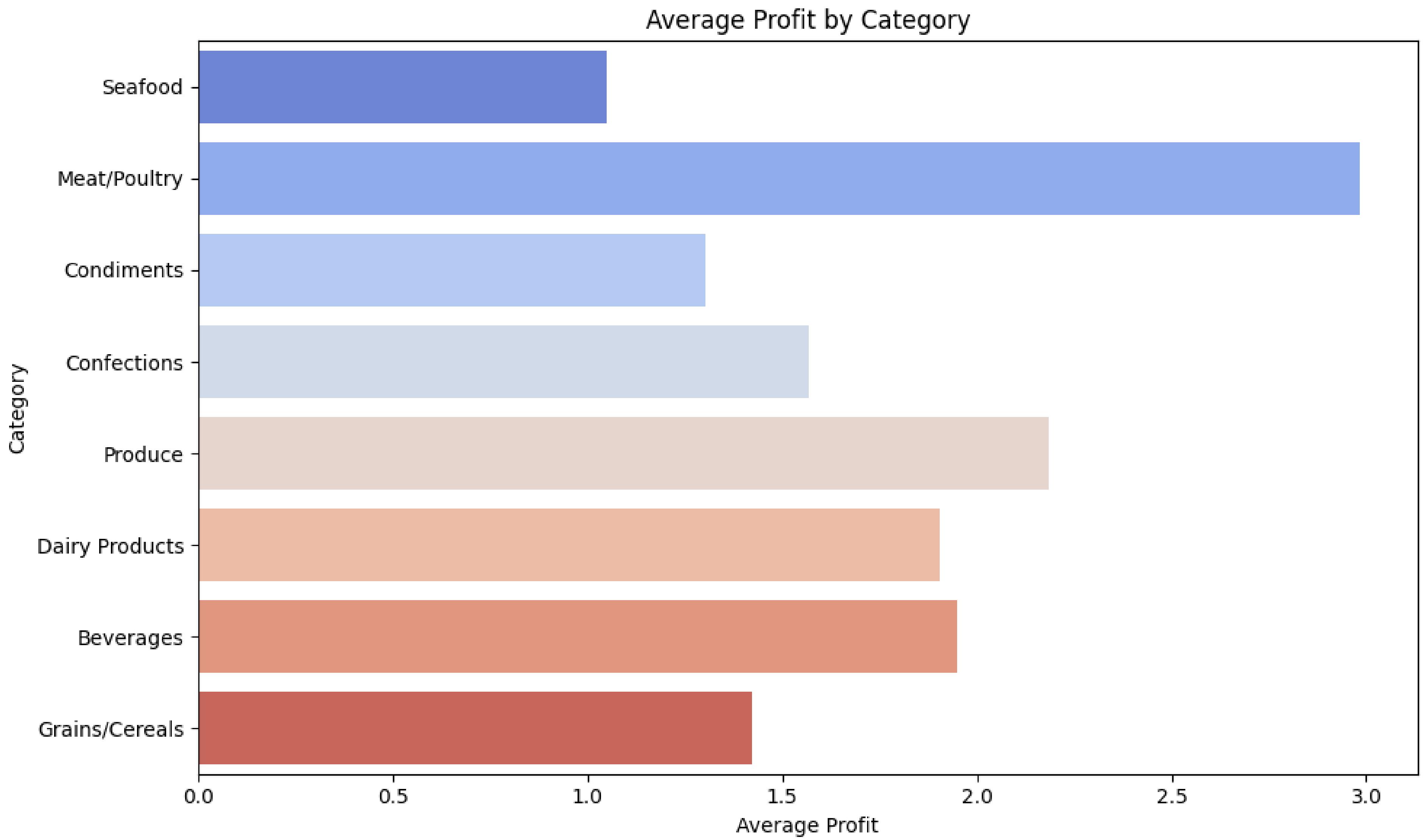
Data Generated



	CategoryName	AvgProfit
0	Seafood	1.0504
1	Meat/Poultry	2.9835
2	Condiments	1.3050
3	Confections	1.5682
4	Produce	2.1827
5	Dairy Products	1.9046
6	Beverages	1.9488
7	Grains/Cereals	1.4219

Profitability by Category

Data Visualization



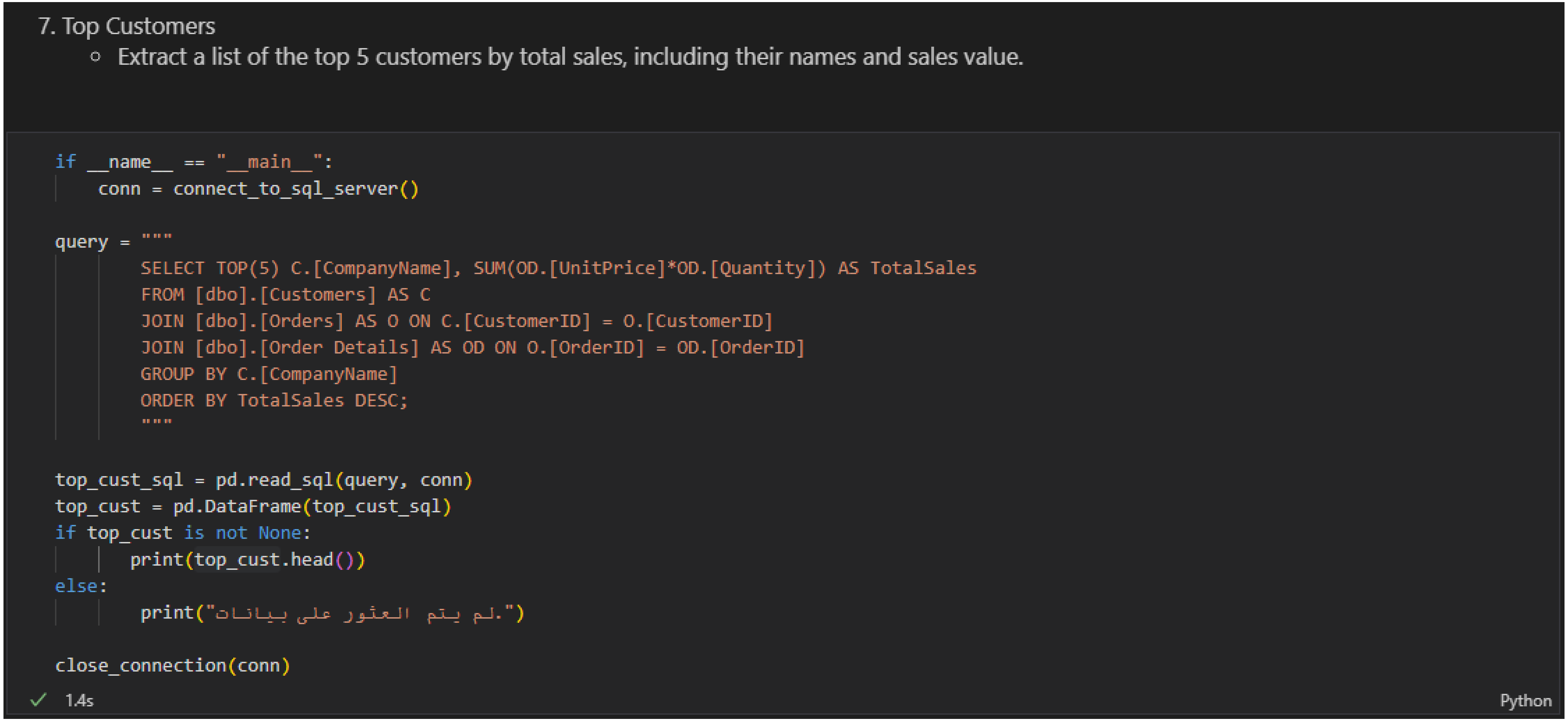
Question 7

Top Customers

Extract a list of the top 5 customers by total sales, including their names and sales value.

Top Customers

SQL Query & Python Code



7. Top Customers
    - Extract a list of the top 5 customers by total sales, including their names and sales value.

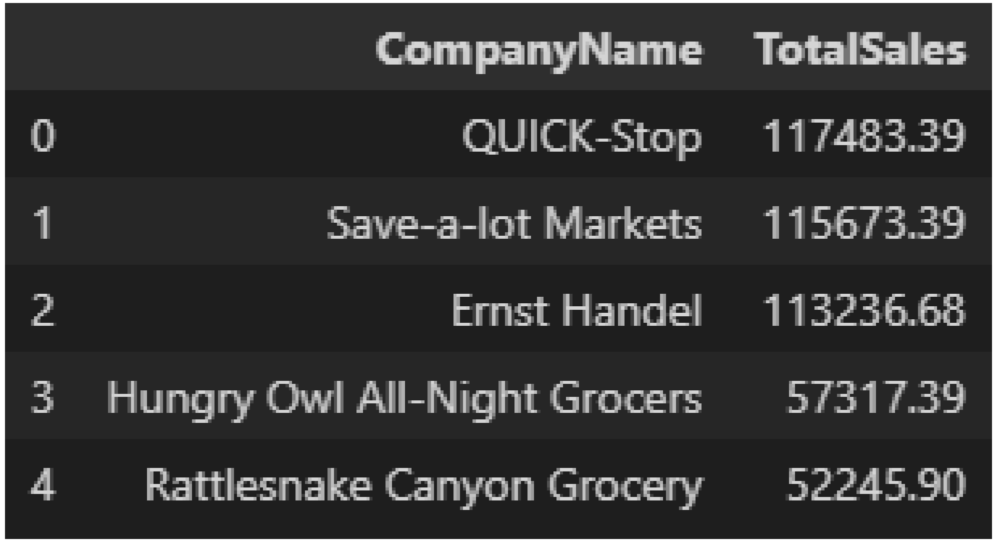
```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT TOP(5) C.[CompanyName], SUM(OD.[UnitPrice]*OD.[Quantity]) AS TotalSales
        FROM [dbo].[Customers] AS C
        JOIN [dbo].[Orders] AS O ON C.[CustomerID] = O.[CustomerID]
        JOIN [dbo].[Order Details] AS OD ON O.[OrderID] = OD.[OrderID]
        GROUP BY C.[CompanyName]
        ORDER BY TotalSales DESC;
        """

top_cust_sql = pd.read_sql(query, conn)
top_cust = pd.DataFrame(top_cust_sql)
if top_cust is not None:
       print(top_cust.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 1.4s

Python

Top Customers

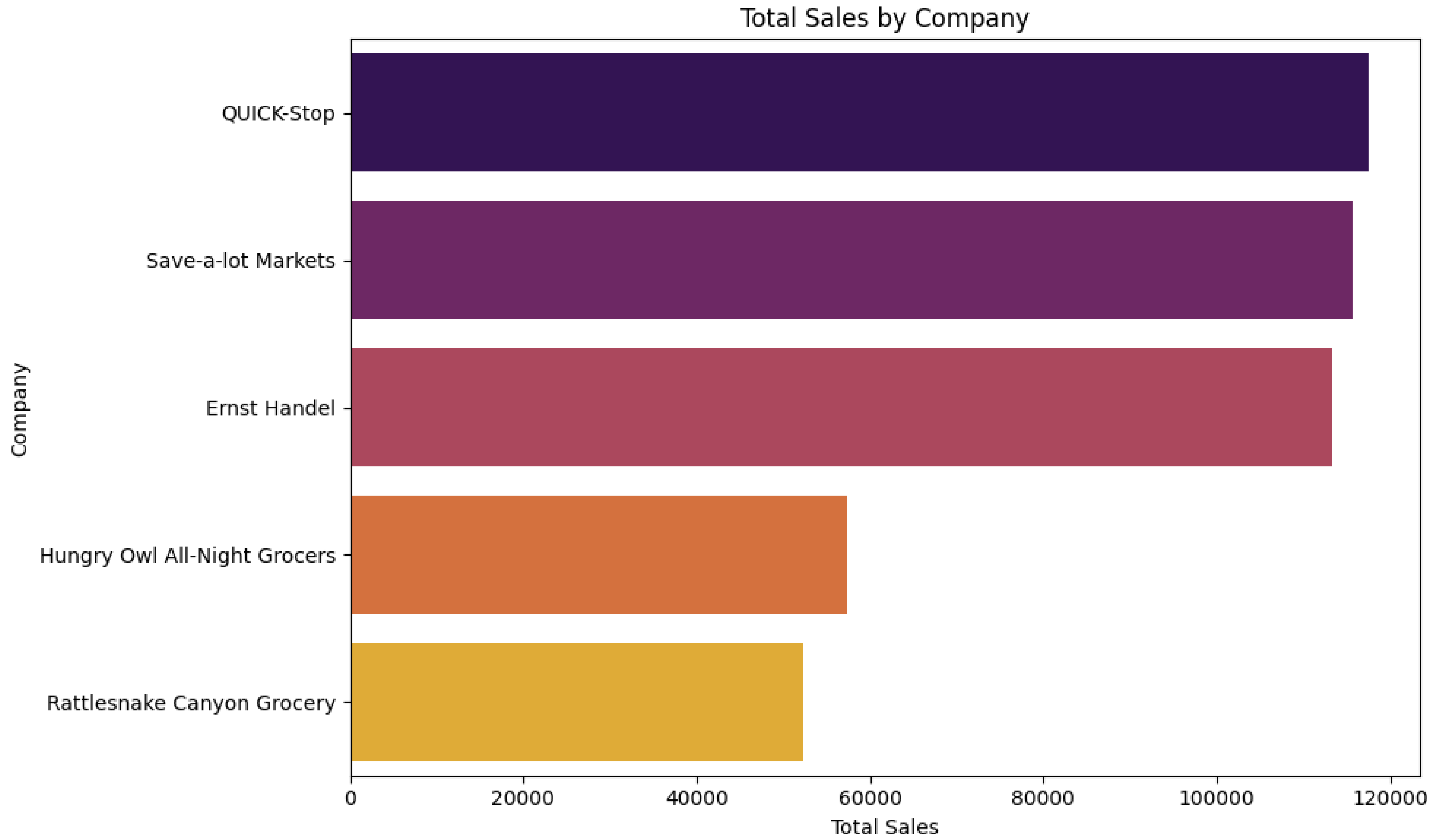
Data Generated



	CompanyName	TotalSales
0	QUICK-Stop	117483.39
1	Save-a-lot Markets	115673.39
2	Ernst Handel	113236.68
3	Hungry Owl All-Night Grocers	57317.39
4	Rattlesnake Canyon Grocery	52245.90

Top Customers

Data Visualization



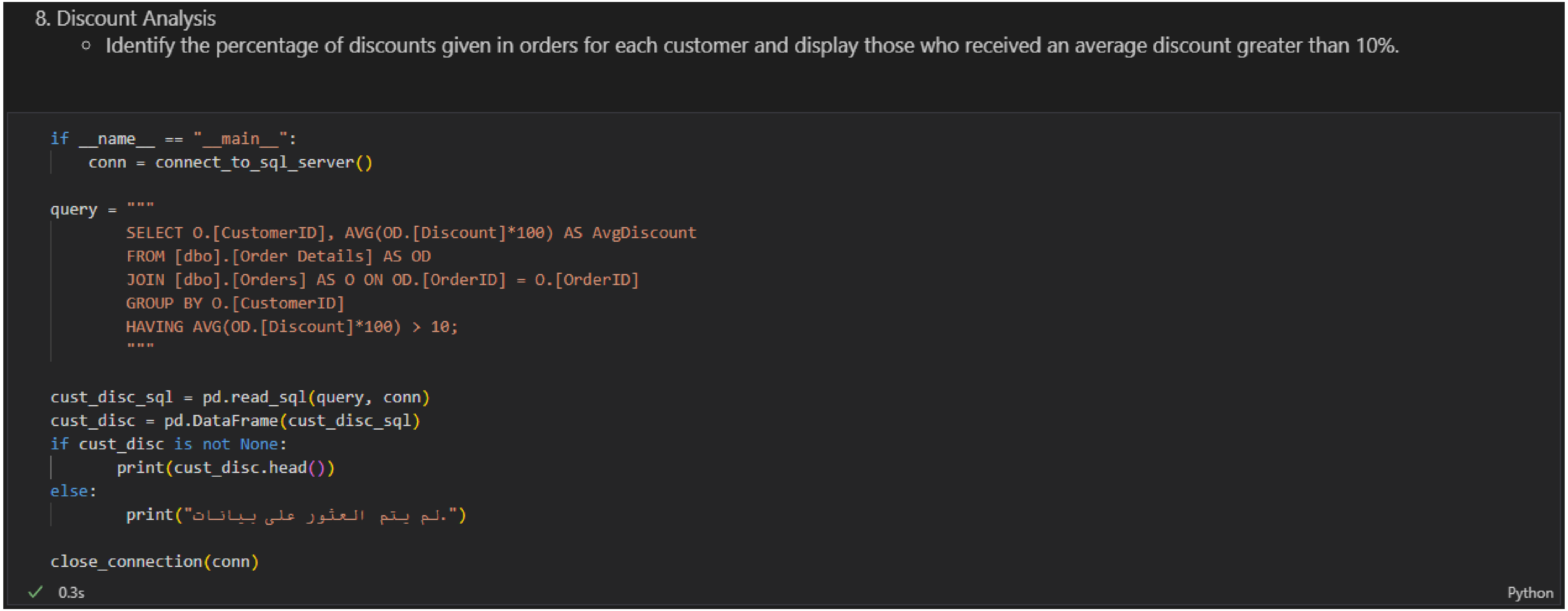
Question 8

Discount Analysis

Identify the percentage of discounts given in orders for each customer and display those who received an average discount greater than 10%.

Discount Analysis

SQL Query & Python Code



8. Discount Analysis

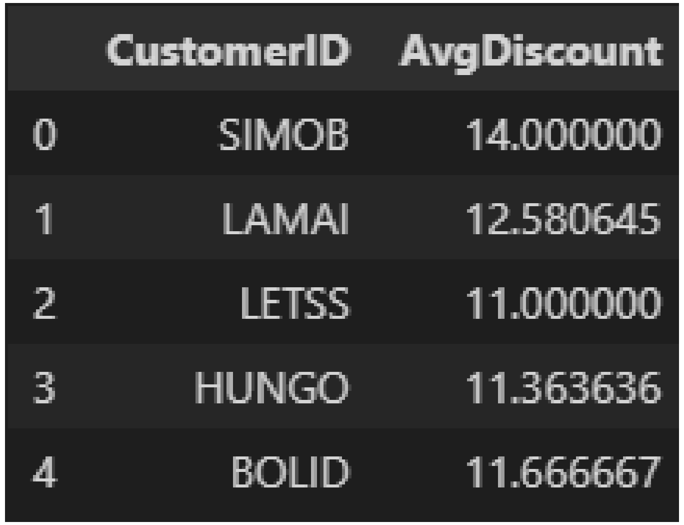
- ○ Identify the percentage of discounts given in orders for each customer and display those who received an average discount greater than 10%.

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT O.[CustomerID], AVG(OD.[Discount]*100) AS AvgDiscount
        FROM [dbo].[Order Details] AS OD
        JOIN [dbo].[Orders] AS O ON OD.[OrderID] = O.[OrderID]
        GROUP BY O.[CustomerID]
        HAVING AVG(OD.[Discount]*100) > 10;
        """

cust_disc_sql = pd.read_sql(query, conn)
cust_disc = pd.DataFrame(cust_disc_sql)
if cust_disc is not None:
       print(cust_disc.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.3s

Discount Analysis

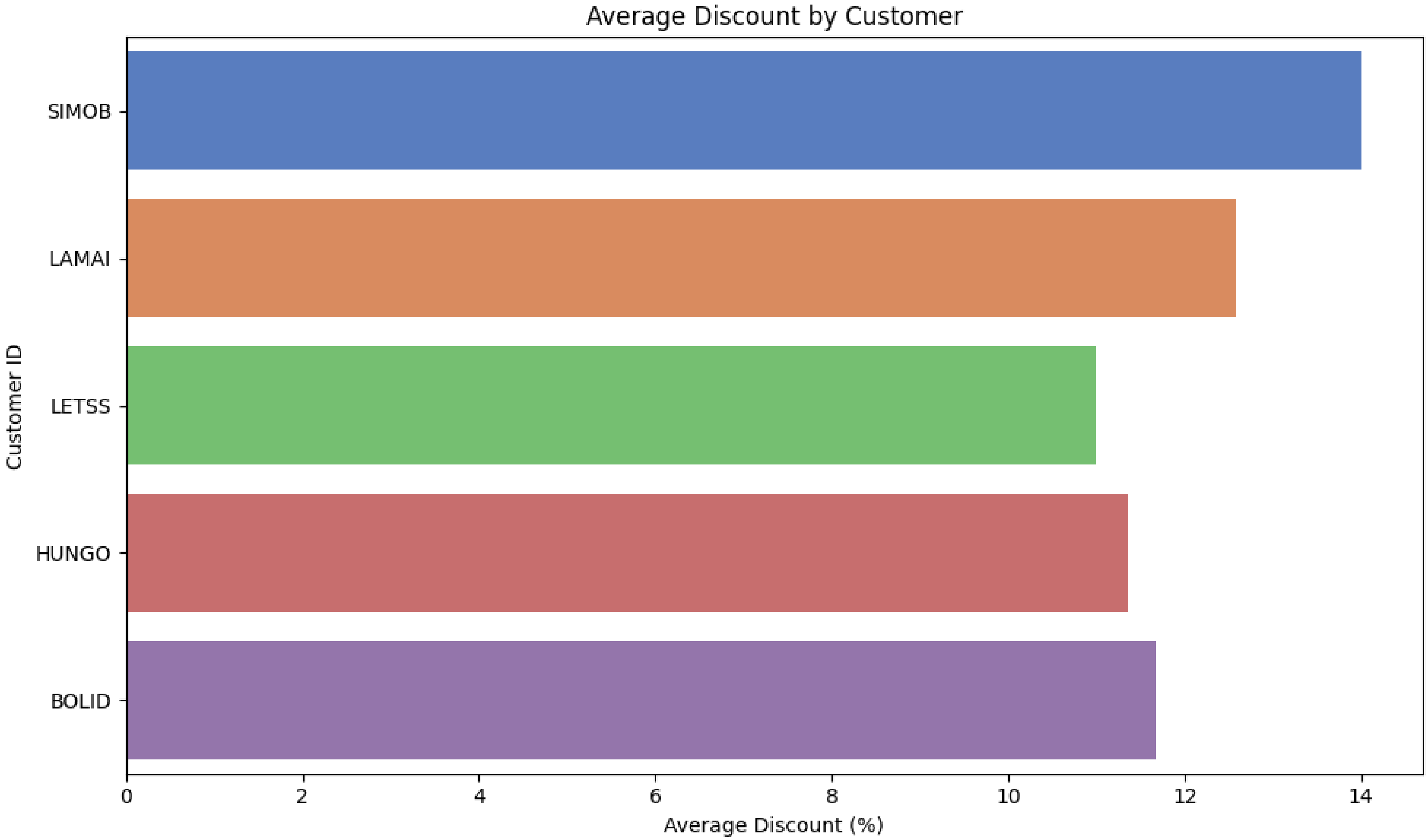
Data Generated



	CustomerID	AvgDiscount
0	SIMOB	14.000000
1	LAMAI	12.580645
2	LETSS	11.000000
3	HUNGO	11.363636
4	BOLID	11.666667

Discount Analysis

Data Visualization



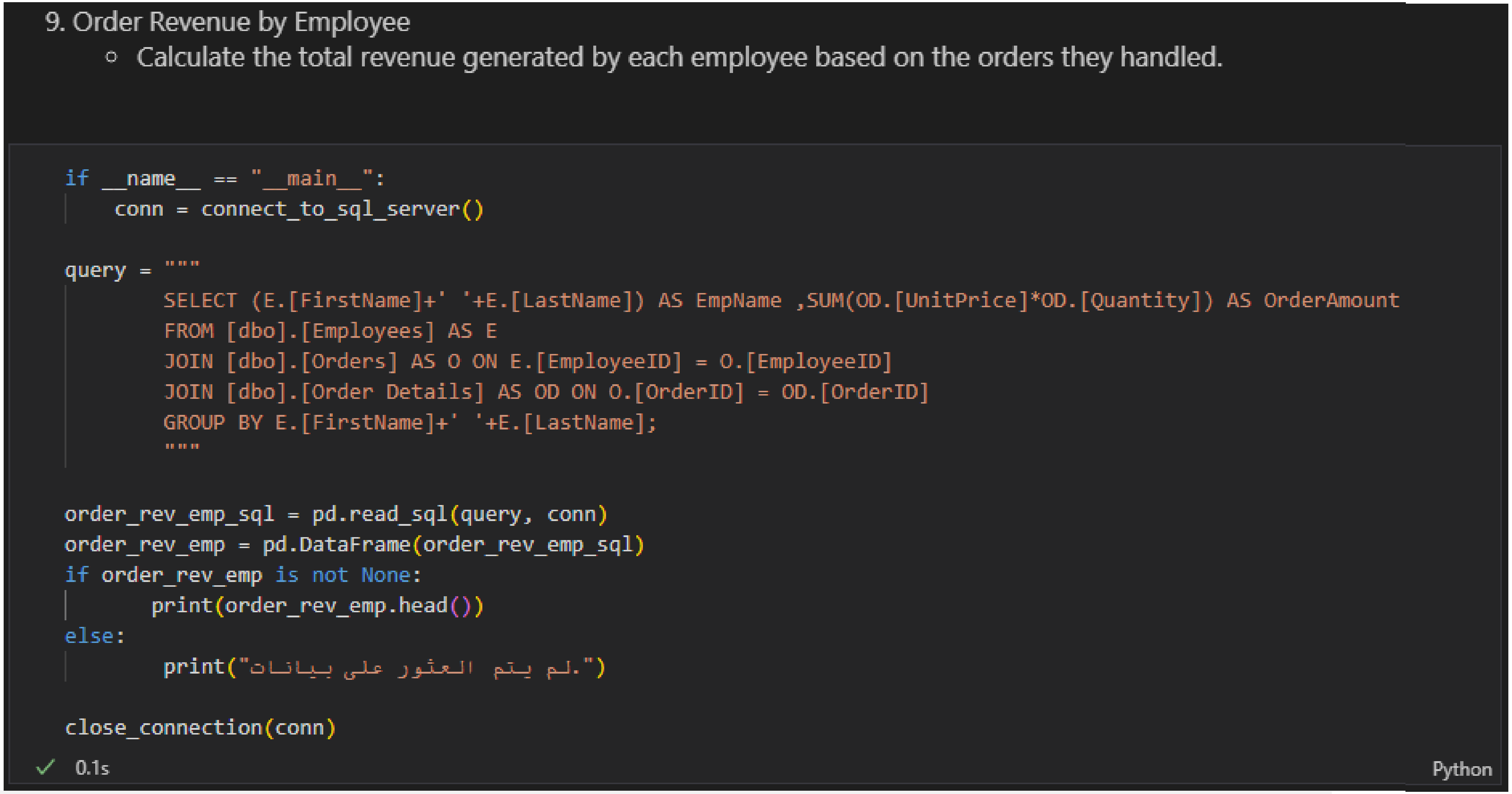
Question 9

Order Revenue by Employee

Calculate the total revenue generated by each employee based on the orders they handled.

Order Revenue by Employee

SQL Query & Python Code

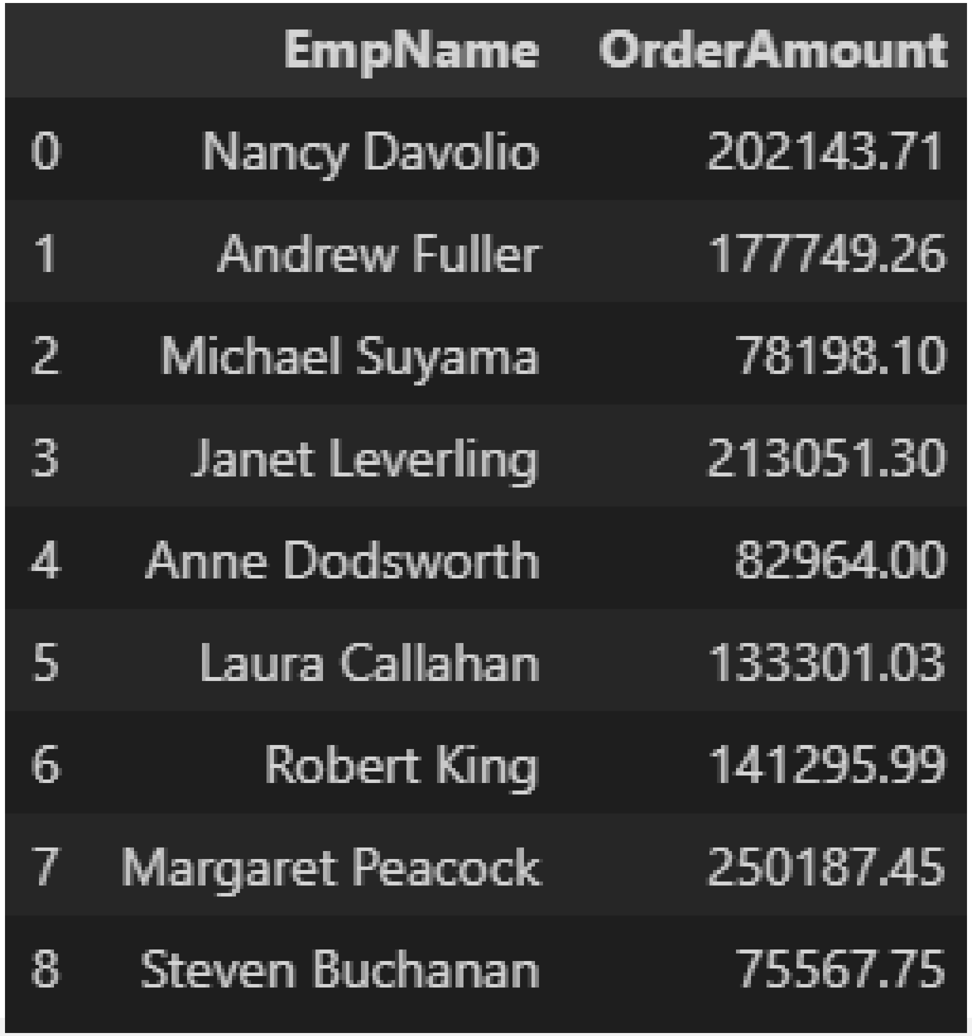


9. Order Revenue by Employee
    - Calculate the total revenue generated by each employee based on the orders they handled.

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT (E.[FirstName]+' '+E.[LastName]) AS EmpName ,SUM(OD.[UnitPrice]*OD.[Quantity]) AS OrderAmount
        FROM [dbo].[Employees] AS E
        JOIN [dbo].[Orders] AS O ON E.[EmployeeID] = O.[EmployeeID]
        JOIN [dbo].[Order Details] AS OD ON O.[OrderID] = OD.[OrderID]
        GROUP BY E.[FirstName]+' '+E.[LastName];
        """

order_rev_emp_sql = pd.read_sql(query, conn)
order_rev_emp = pd.DataFrame(order_rev_emp_sql)
if order_rev_emp is not None:
       print(order_rev_emp.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Order Revenue by Employee

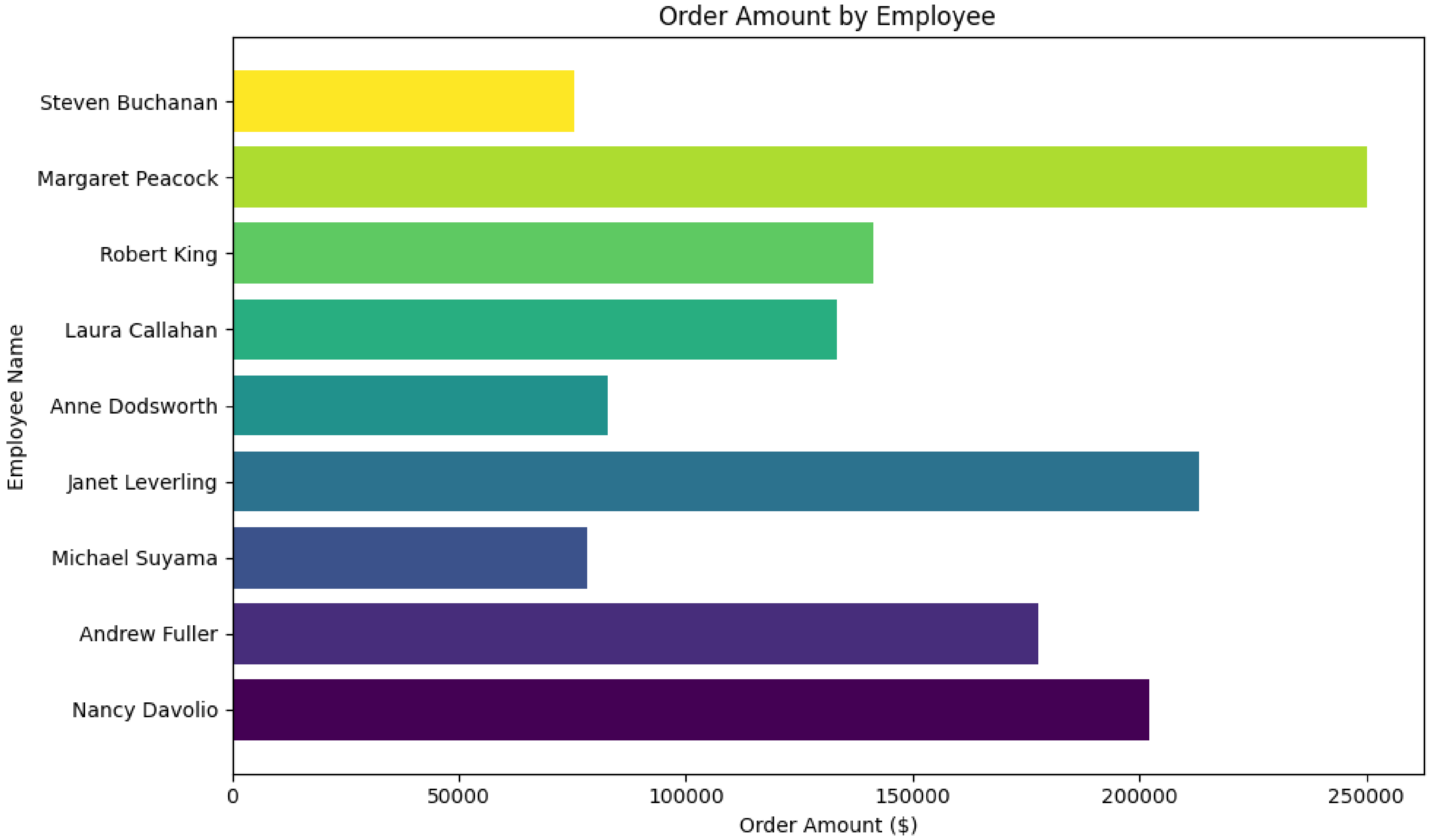
Data Generated



	EmpName	OrderAmount
0	Nancy Davolio	202143.71
1	Andrew Fuller	177749.26
2	Michael Suyama	78198.10
3	Janet Leverling	213051.30
4	Anne Dodsworth	82964.00
5	Laura Callahan	133301.03
6	Robert King	141295.99
7	Margaret Peacock	250187.45
8	Steven Buchanan	75567.75

Order Revenue by Employee

Data Visualization



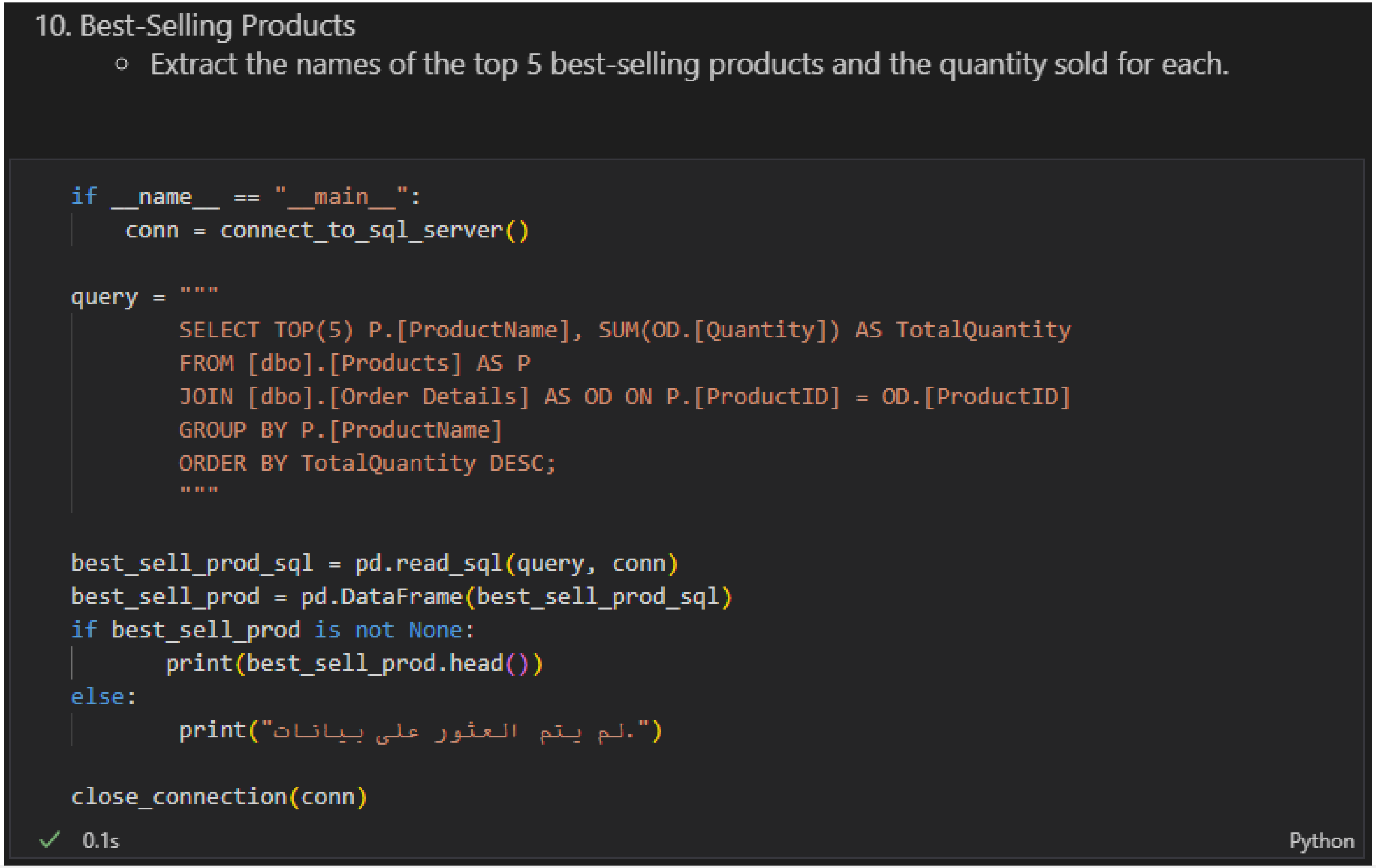
Question 10

Best-Selling Products

Extract the names of the top 5 best-selling products and the quantity sold for each.

Best-Selling Products

SQL Query & Python Code



10. Best-Selling Products

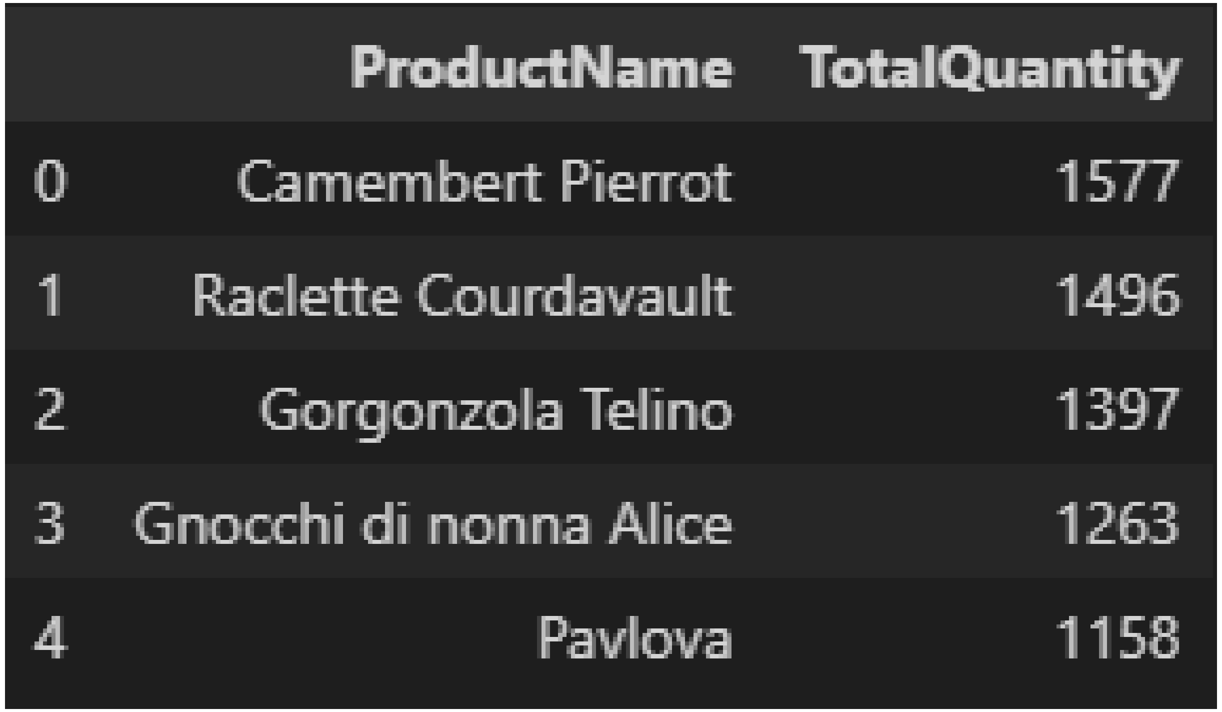
- Extract the names of the top 5 best-selling products and the quantity sold for each.

```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT TOP(5) P.[ProductName], SUM(OD.[Quantity]) AS TotalQuantity
        FROM [dbo].[Products] AS P
        JOIN [dbo].[Order Details] AS OD ON P.[ProductID] = OD.[ProductID]
        GROUP BY P.[ProductName]
        ORDER BY TotalQuantity DESC;
        """

best_sell_prod_sql = pd.read_sql(query, conn)
best_sell_prod = pd.DataFrame(best_sell_prod_sql)
if best_sell_prod is not None:
       print(best_sell_prod.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.1s

Best-Selling Products

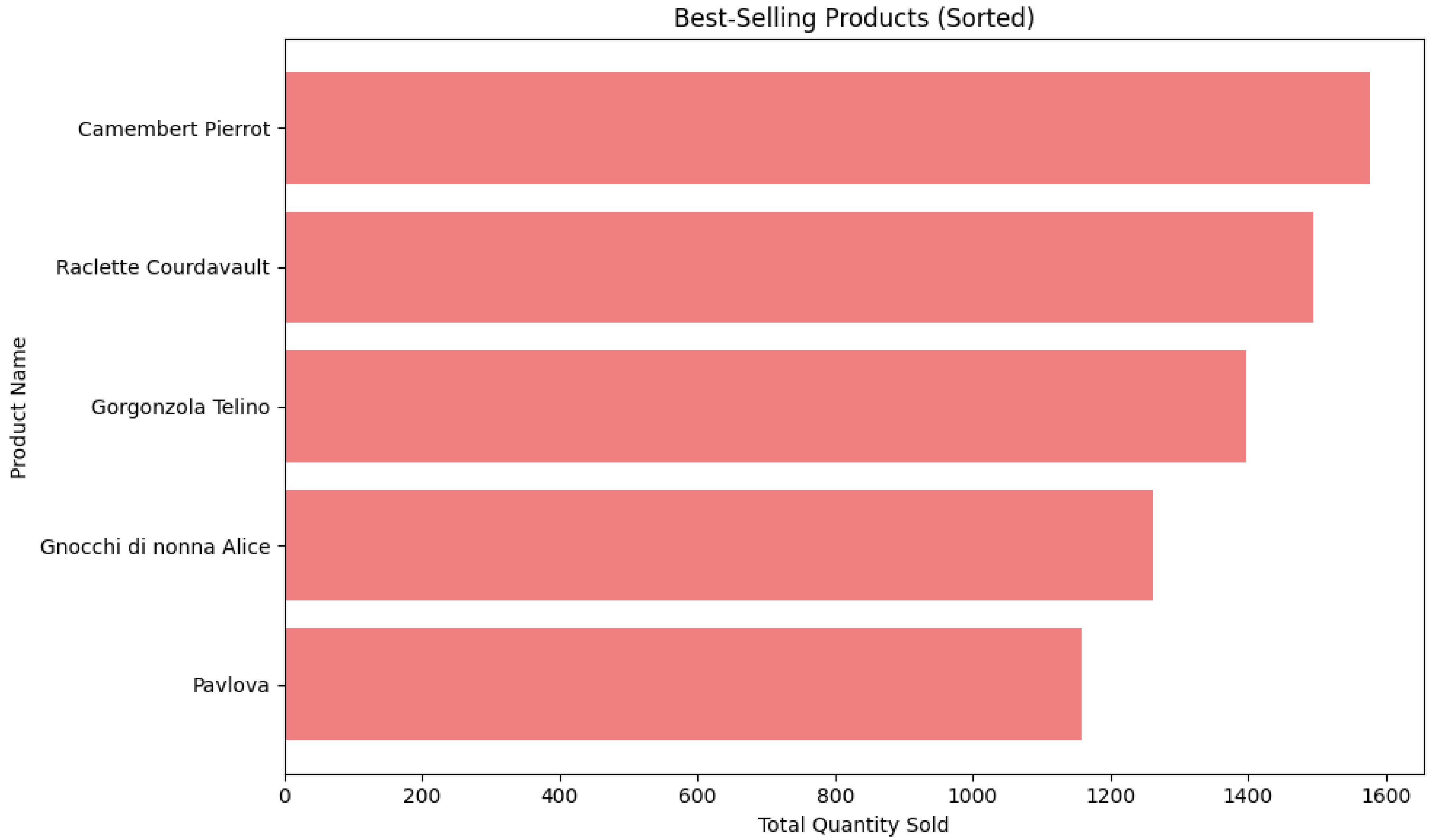
Data Generated



	ProductName	TotalQuantity
0	Camembert Pierrot	1577
1	Raclette Courdavault	1496
2	Gorgonzola Telino	1397
3	Gnocchi di nonna Alice	1263
4	Pavlova	1158

Best-Selling Products

Data Visualization



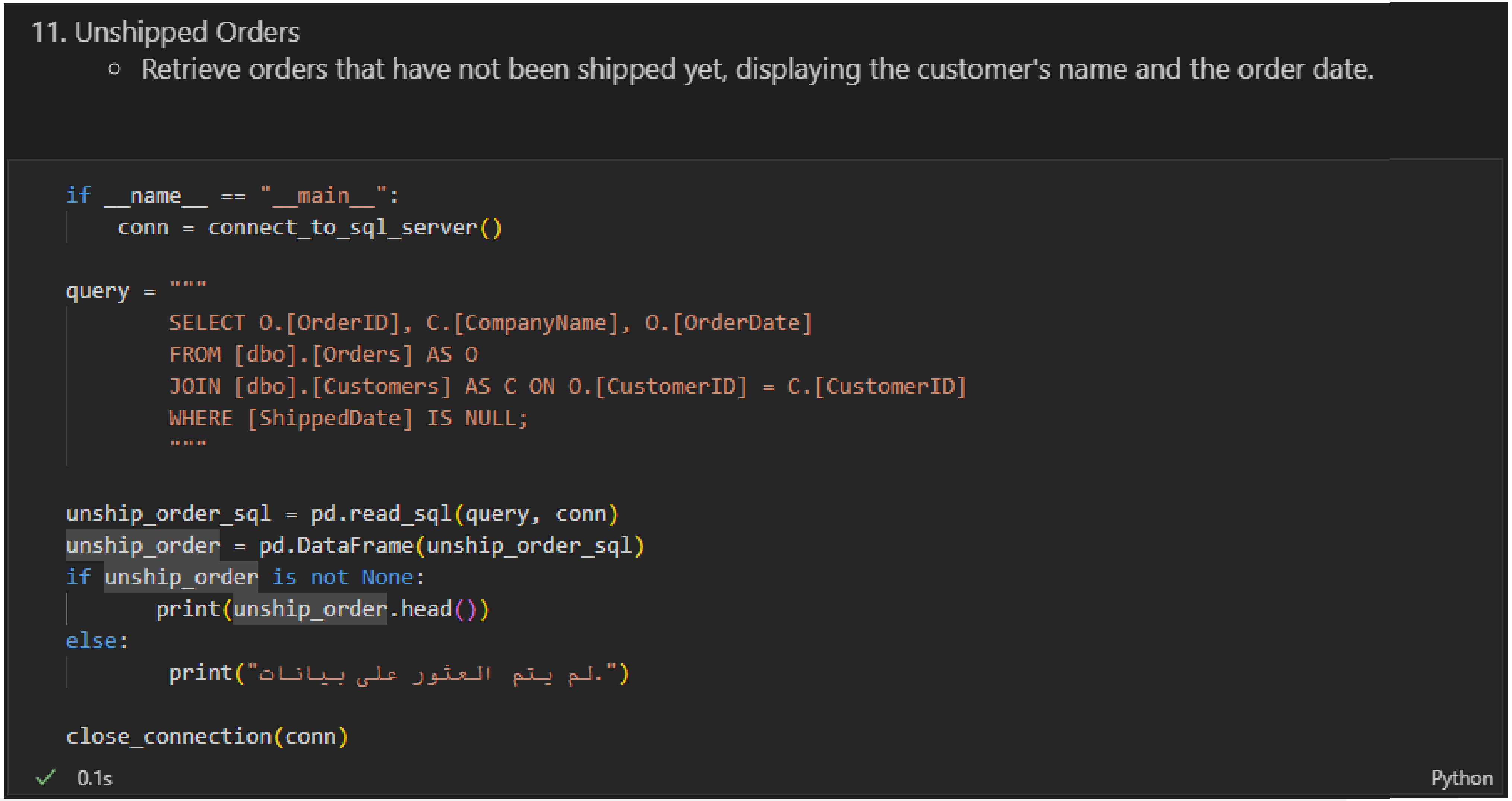
Question 11

Unshipped Orders

Retrieve orders that have not been shipped yet, displaying the customer's name and the order date.

Unshipped Orders

SQL Query & Python Code

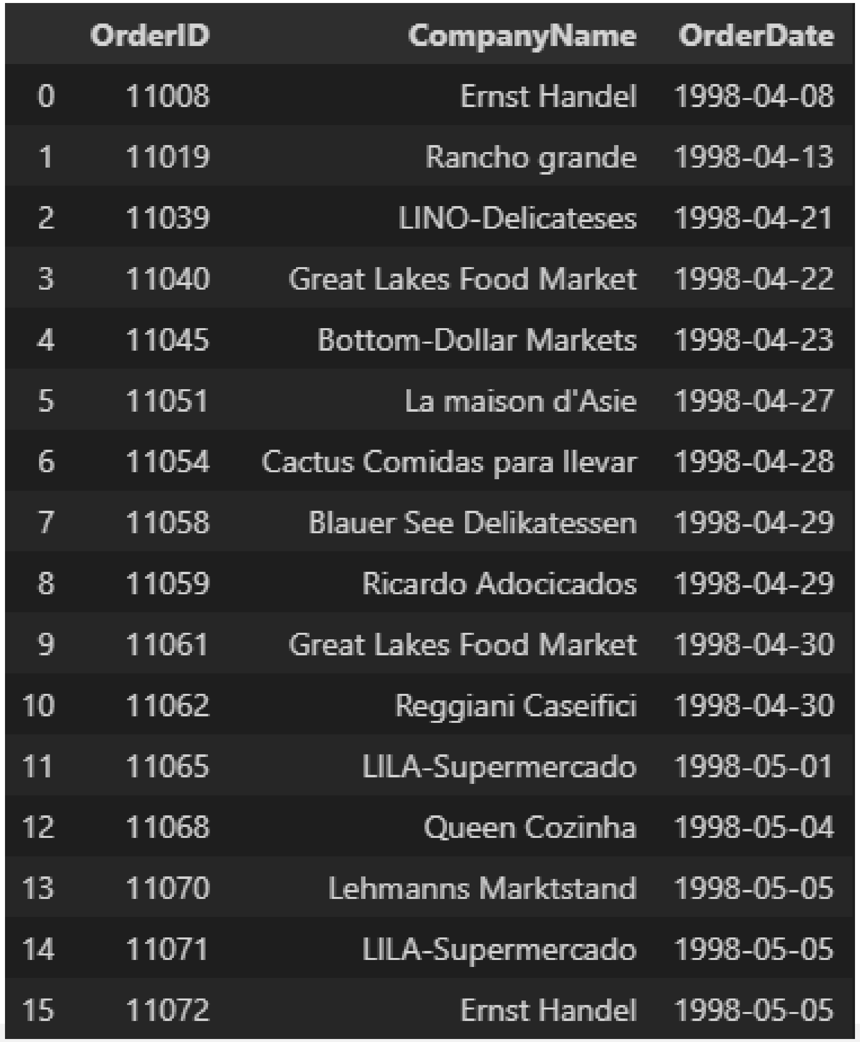


11. Unshipped Orders
    - Retrieve orders that have not been shipped yet, displaying the customer's name and the order date.

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT O.[OrderID], C.[CompanyName], O.[OrderDate]
        FROM [dbo].[Orders] AS O
        JOIN [dbo].[Customers] AS C ON O.[CustomerID] = C.[CustomerID]
        WHERE [ShippedDate] IS NULL;
        """

unship_order_sql = pd.read_sql(query, conn)
unship_order = pd.DataFrame(unship_order_sql)
if unship_order is not None:
       print(unship_order.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Unshipped Orders

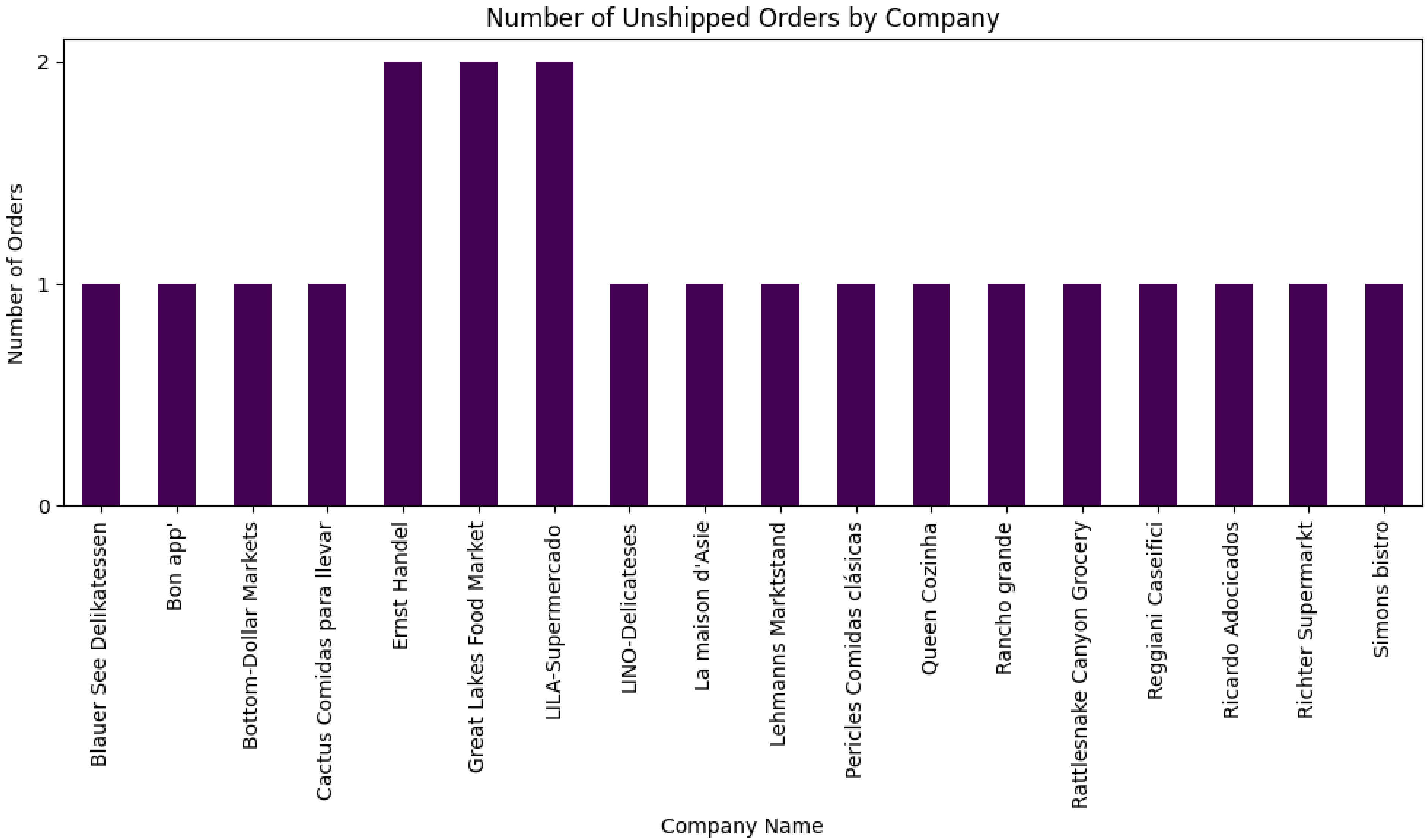
Portion of the Data Generated



	OrderID	CompanyName	OrderDate
0	11008	Ernst Handel	1998-04-08
1	11019	Rancho grande	1998-04-13
2	11039	LINO-Delicateses	1998-04-21
3	11040	Great Lakes Food Market	1998-04-22
4	11045	Bottom-Dollar Markets	1998-04-23
5	11051	La maison d'Asie	1998-04-27
6	11054	Cactus Comidas para llevar	1998-04-28
7	11058	Blauer See Delikatessen	1998-04-29
8	11059	Ricardo Adocicados	1998-04-29
9	11061	Great Lakes Food Market	1998-04-30
10	11062	Reggiani Caseifici	1998-04-30
11	11065	LILA-Supermercado	1998-05-01
12	11068	Queen Cozinha	1998-05-04
13	11070	Lehmanns Marktstand	1998-05-05
14	11071	LILA-Supermercado	1998-05-05
15	11072	Ernst Handel	1998-05-05

Unshipped Orders

Data Visualization



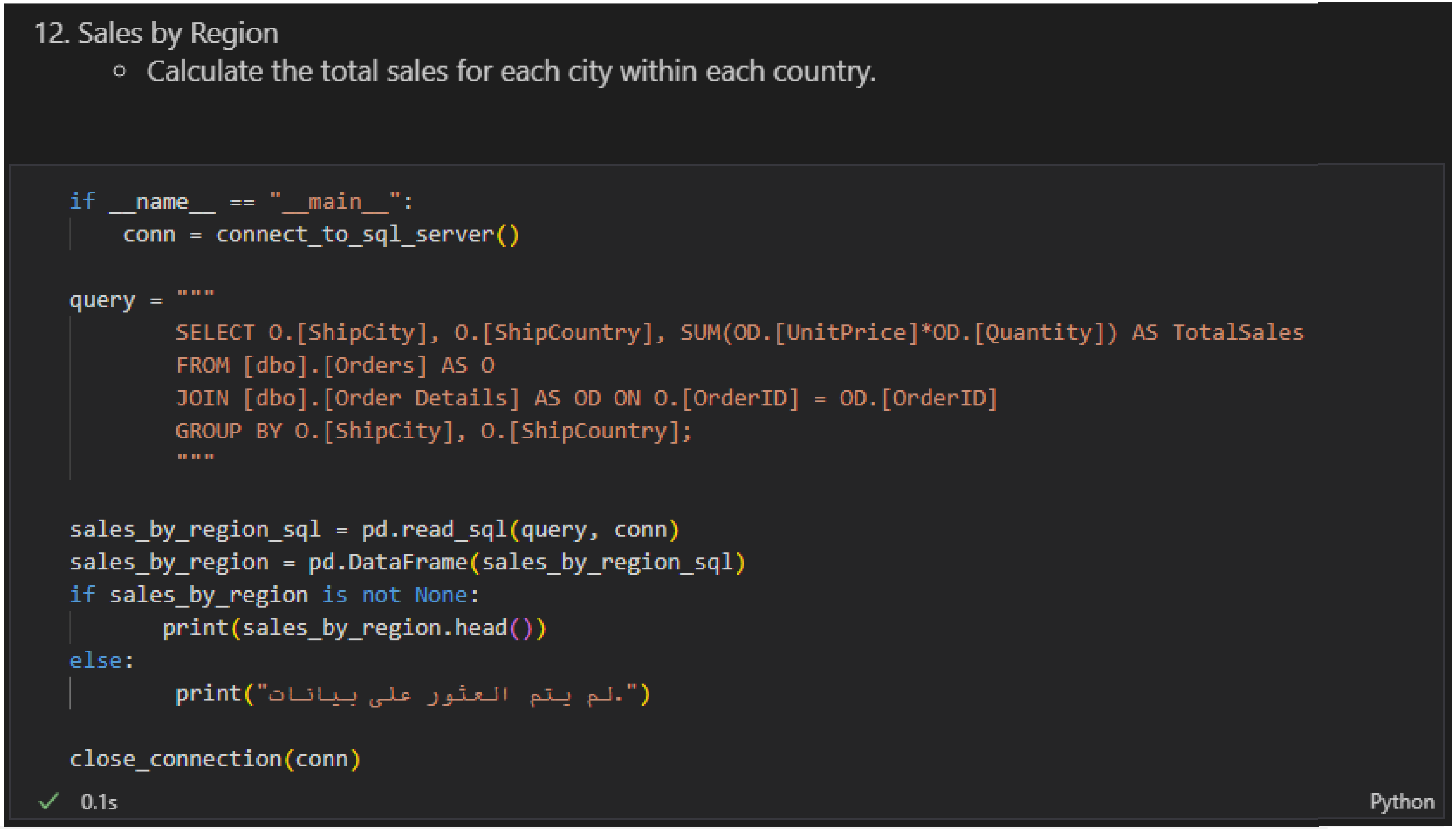
Question 12

Sales by Region

Calculate the total sales for each city within each country.

Sales by Region

SQL Query & Python Code



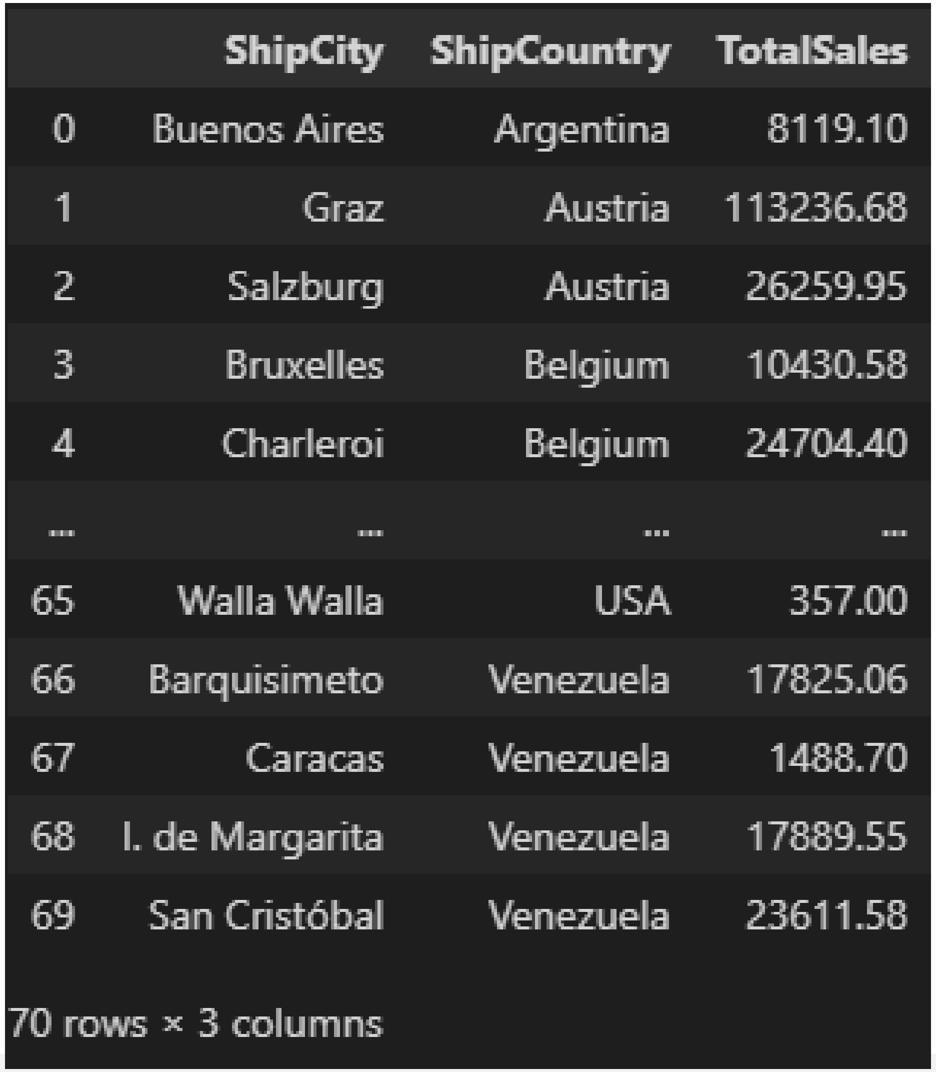
12. Sales by Region

  - Calculate the total sales for each city within each country.

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT O.[ShipCity], O.[ShipCountry], SUM(OD.[UnitPrice]*OD.[Quantity]) AS TotalSales
        FROM [dbo].[Orders] AS O
        JOIN [dbo].[Order Details] AS OD ON O.[OrderID] = OD.[OrderID]
        GROUP BY O.[ShipCity], O.[ShipCountry];
        """

sales_by_region_sql = pd.read_sql(query, conn)
sales_by_region = pd.DataFrame(sales_by_region_sql)
if sales_by_region is not None:
       print(sales_by_region.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Sales by Region

Data Generated

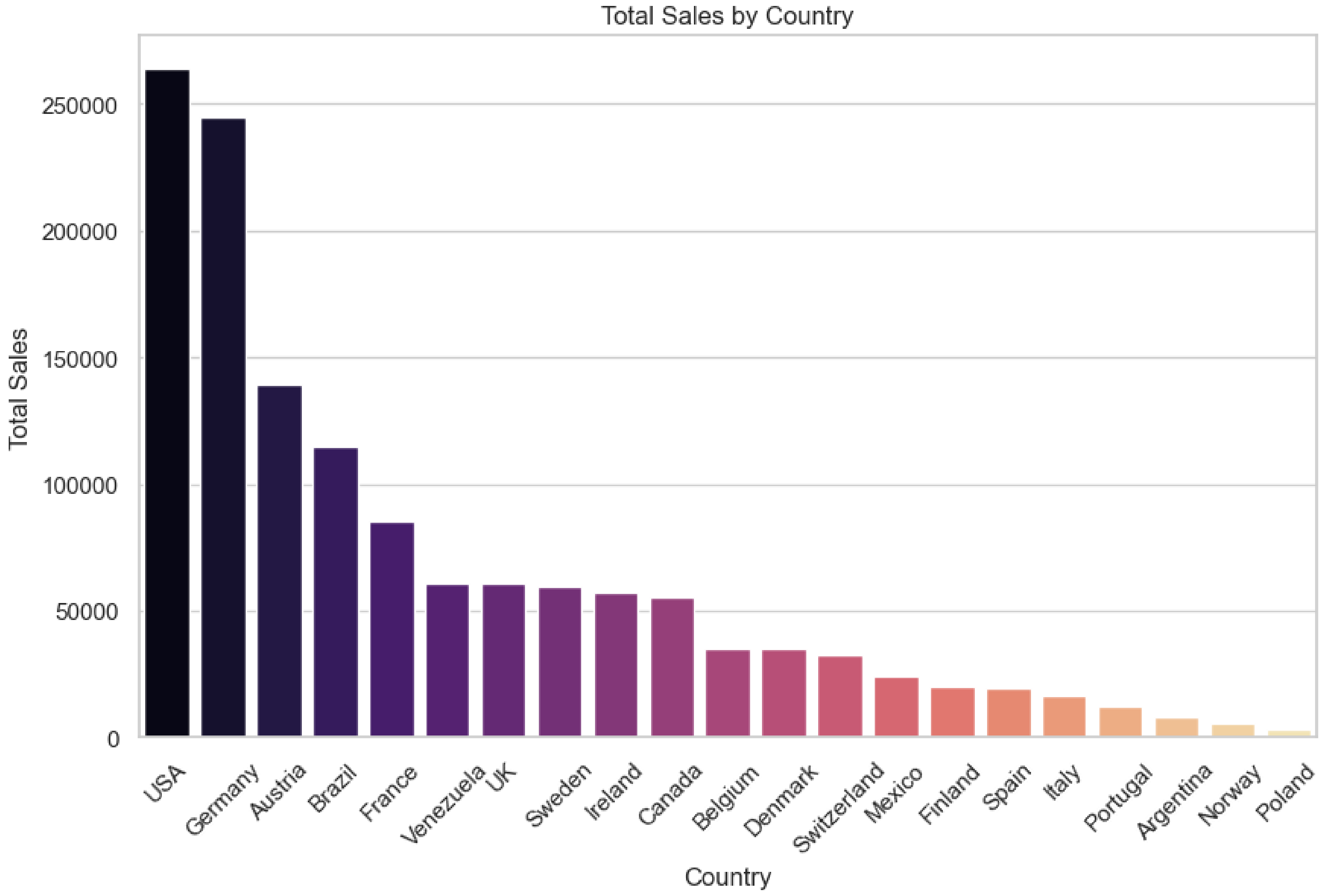


	ShipCity	ShipCountry	TotalSales
0	Buenos Aires	Argentina	8119.10
1	Graz	Austria	113236.68
2	Salzburg	Austria	26259.95
3	Bruxelles	Belgium	10430.58
4	Charleroi	Belgium	24704.40
...	...	...	...
65	Walla Walla	USA	357.00
66	Barquisimeto	Venezuela	17825.06
67	Caracas	Venezuela	1488.70
68	I. de Margarita	Venezuela	17889.55
69	San Cristóbal	Venezuela	23611.58

70 rows × 3 columns

Sales by Region

Data Visualization



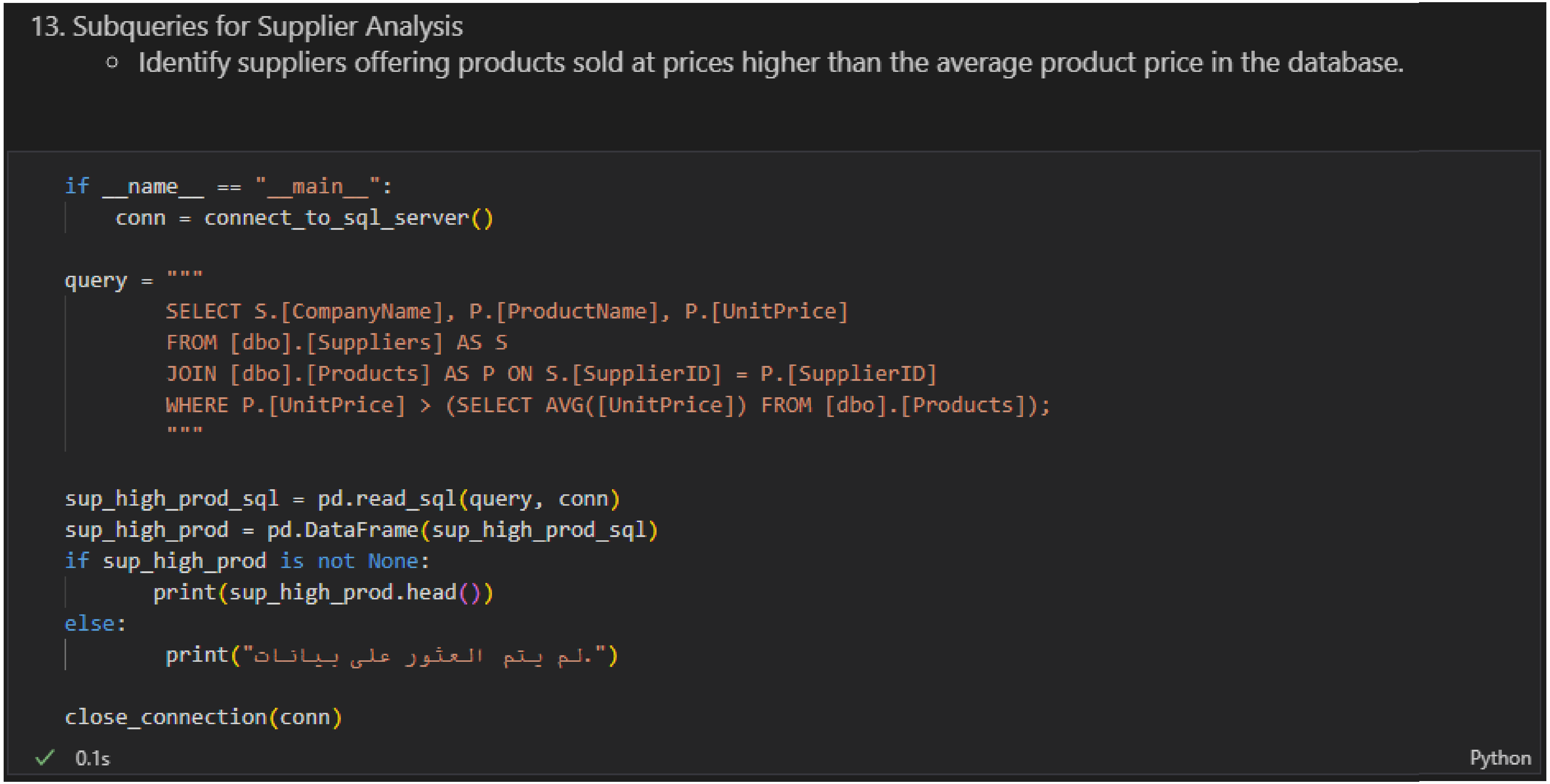
Question 13

Sub-queries for Supplier Analysis

Identify suppliers offering products sold at prices higher than the average product price in the database.

Sub-queries for Supplier Analysis

SQL Query & Python Code



13. Subqueries for Supplier Analysis
    - ○ Identify suppliers offering products sold at prices higher than the average product price in the database.

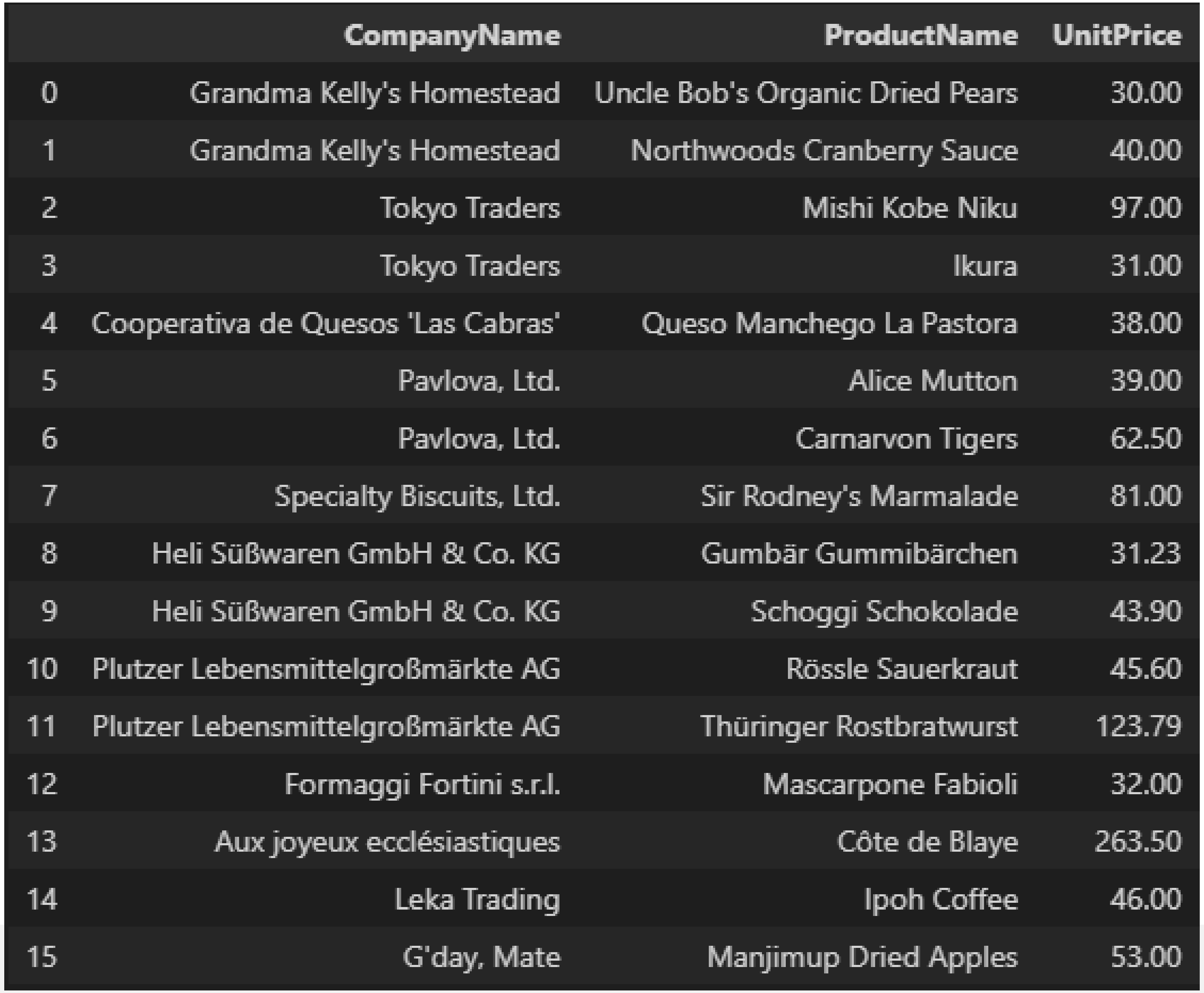
```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT S.[CompanyName], P.[ProductName], P.[UnitPrice]
        FROM [dbo].[Suppliers] AS S
        JOIN [dbo].[Products] AS P ON S.[SupplierID] = P.[SupplierID]
        WHERE P.[UnitPrice] > (SELECT AVG([UnitPrice]) FROM [dbo].[Products]);
        """

sup_high_prod_sql = pd.read_sql(query, conn)
sup_high_prod = pd.DataFrame(sup_high_prod_sql)
if sup_high_prod is not None:
       print(sup_high_prod.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.1s

Python

Sub-queries for Supplier Analysis

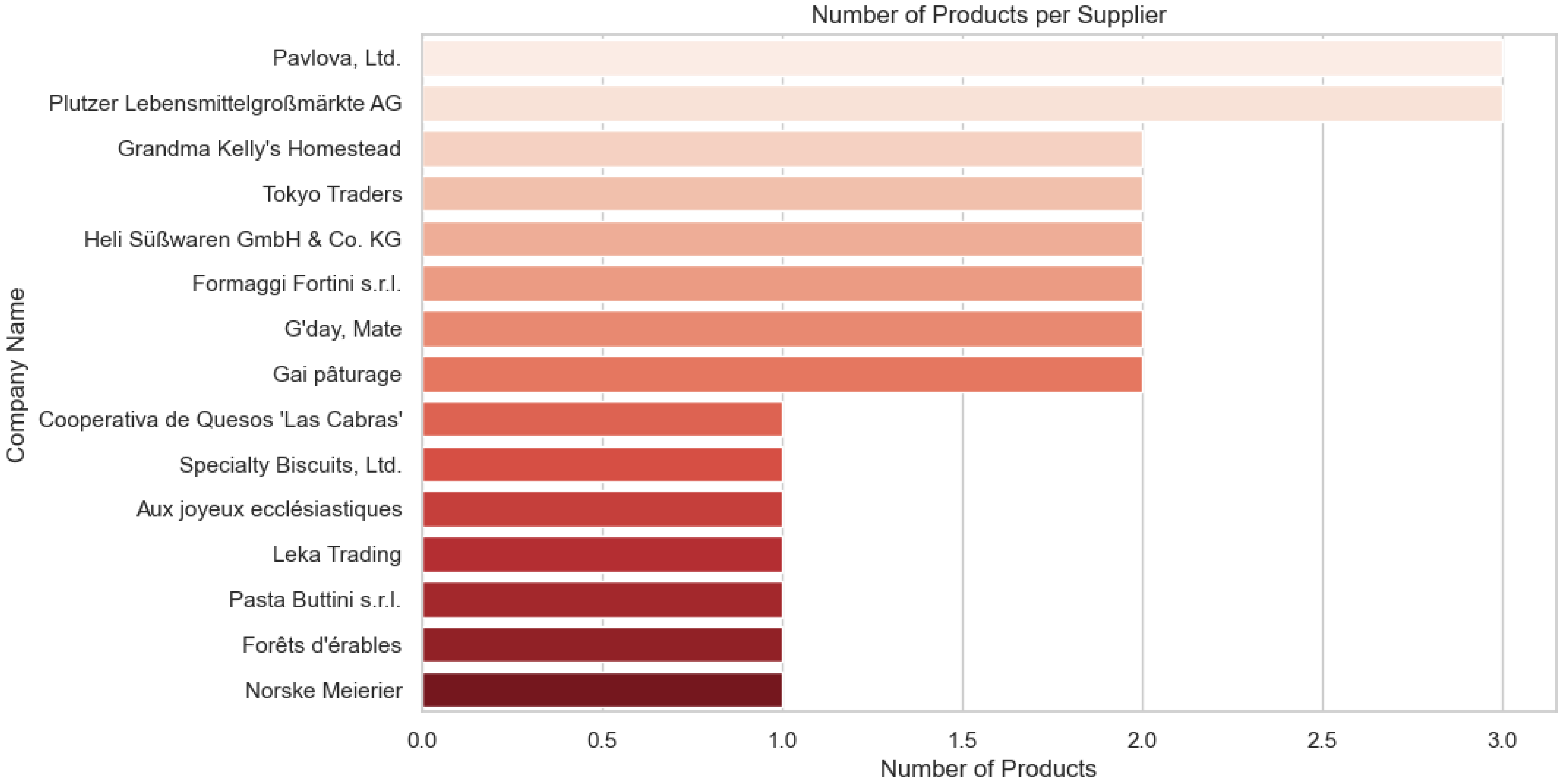
Portion of the Data Generated



	CompanyName	ProductName	UnitPrice
0	Grandma Kelly's Homestead	Uncle Bob's Organic Dried Pears	30.00
1	Grandma Kelly's Homestead	Northwoods Cranberry Sauce	40.00
2	Tokyo Traders	Mishi Kobe Niku	97.00
3	Tokyo Traders	Ikura	31.00
4	Cooperativa de Quesos 'Las Cabras'	Queso Manchego La Pastora	38.00
5	Pavlova, Ltd.	Alice Mutton	39.00
6	Pavlova, Ltd.	Carnarvon Tigers	62.50
7	Specialty Biscuits, Ltd.	Sir Rodney's Marmalade	81.00
8	Heli Süßwaren GmbH & Co. KG	Gumbär Gummibärchen	31.23
9	Heli Süßwaren GmbH & Co. KG	Schoggi Schokolade	43.90
10	Plutzer Lebensmittelgroßmärkte AG	Rössle Sauerkraut	45.60
11	Plutzer Lebensmittelgroßmärkte AG	Thüringer Rostbratwurst	123.79
12	Formaggi Fortini s.r.l.	Mascarpone Fabioli	32.00
13	Aux joyeux ecclésiastiques	Côte de Blaye	263.50
14	Leka Trading	Ipoh Coffee	46.00
15	G'day, Mate	Manjimup Dried Apples	53.00

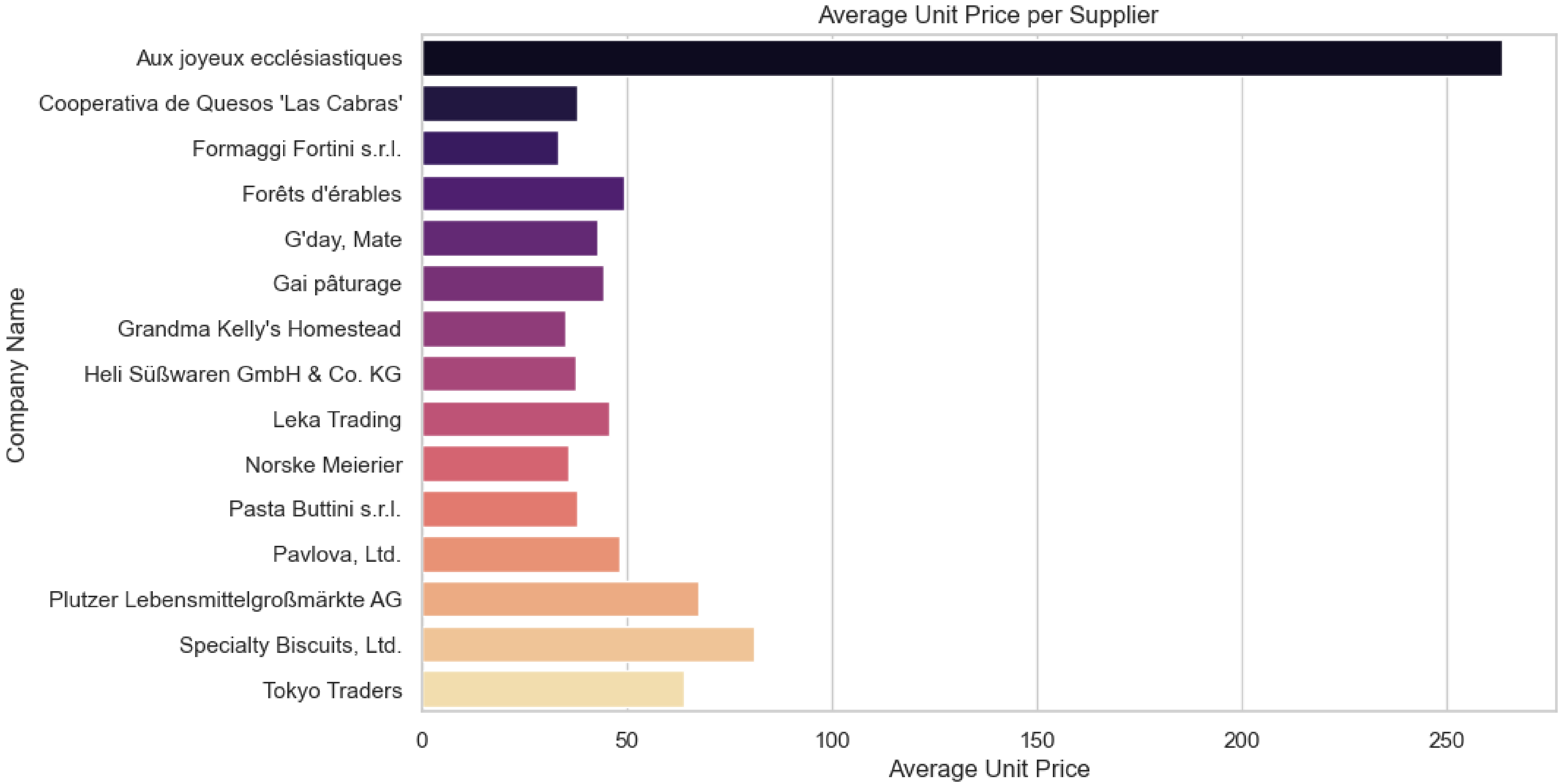
Sub-queries for Supplier Analysis

Data Visualization for Number of Products per Supplier



Sub-queries for Supplier Analysis

Data Visualization for Average Unit Price per Supplier



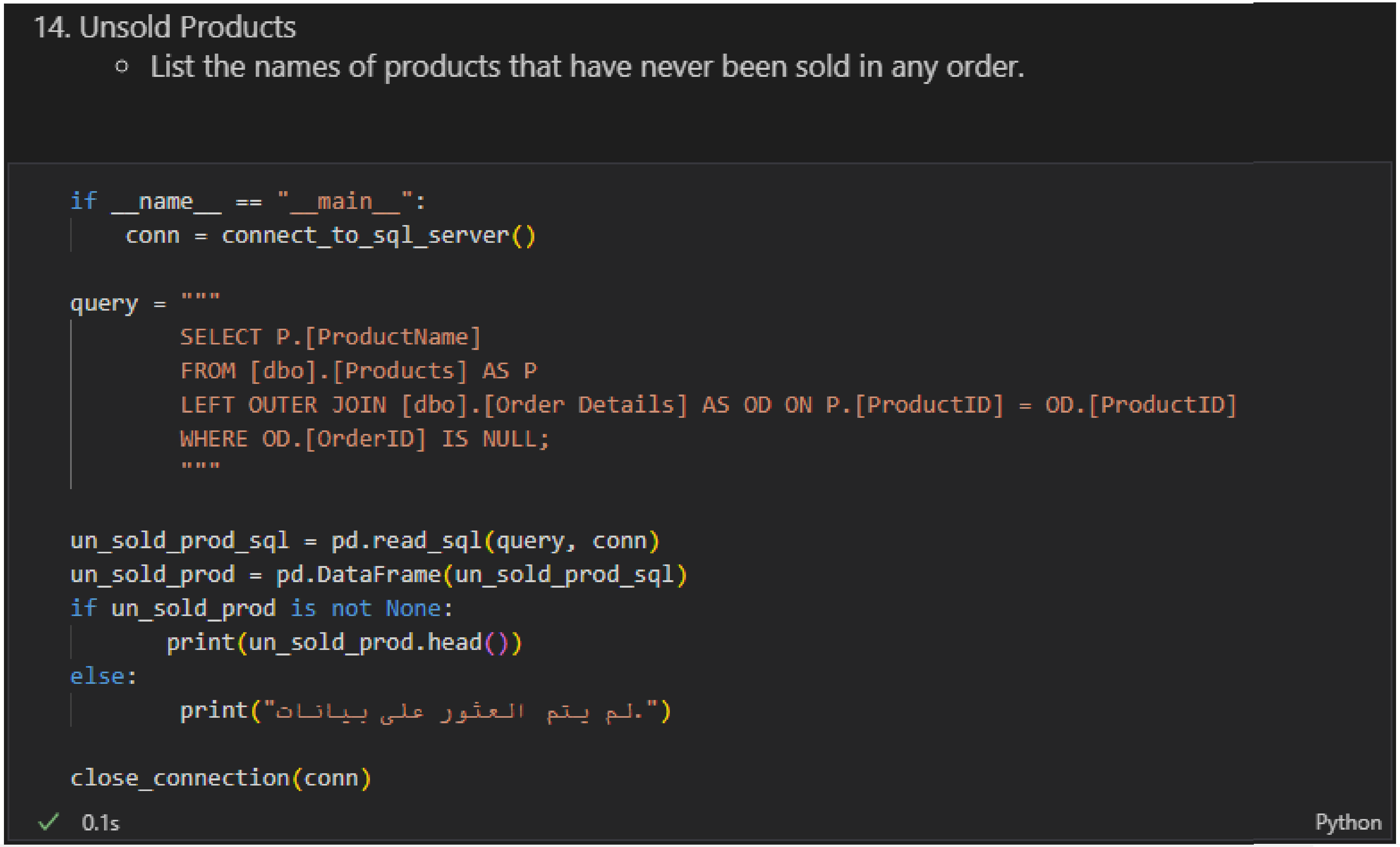
Question 14

Unsold Products

List the names of products that have never been sold in any order.

Unsold Products

SQL Query & Python Code



14. Unsold Products

- List the names of products that have never been sold in any order.

```
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT P.[ProductName]
        FROM [dbo].[Products] AS P
        LEFT OUTER JOIN [dbo].[Order Details] AS OD ON P.[ProductID] = OD.[ProductID]
        WHERE OD.[OrderID] IS NULL;
        """

un_sold_prod_sql = pd.read_sql(query, conn)
un_sold_prod = pd.DataFrame(un_sold_prod_sql)
if un_sold_prod is not None:
       print(un_sold_prod.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

Unsold Products

Data Generated

No data is generated, means there is not any of the products that wasn’t sold before.

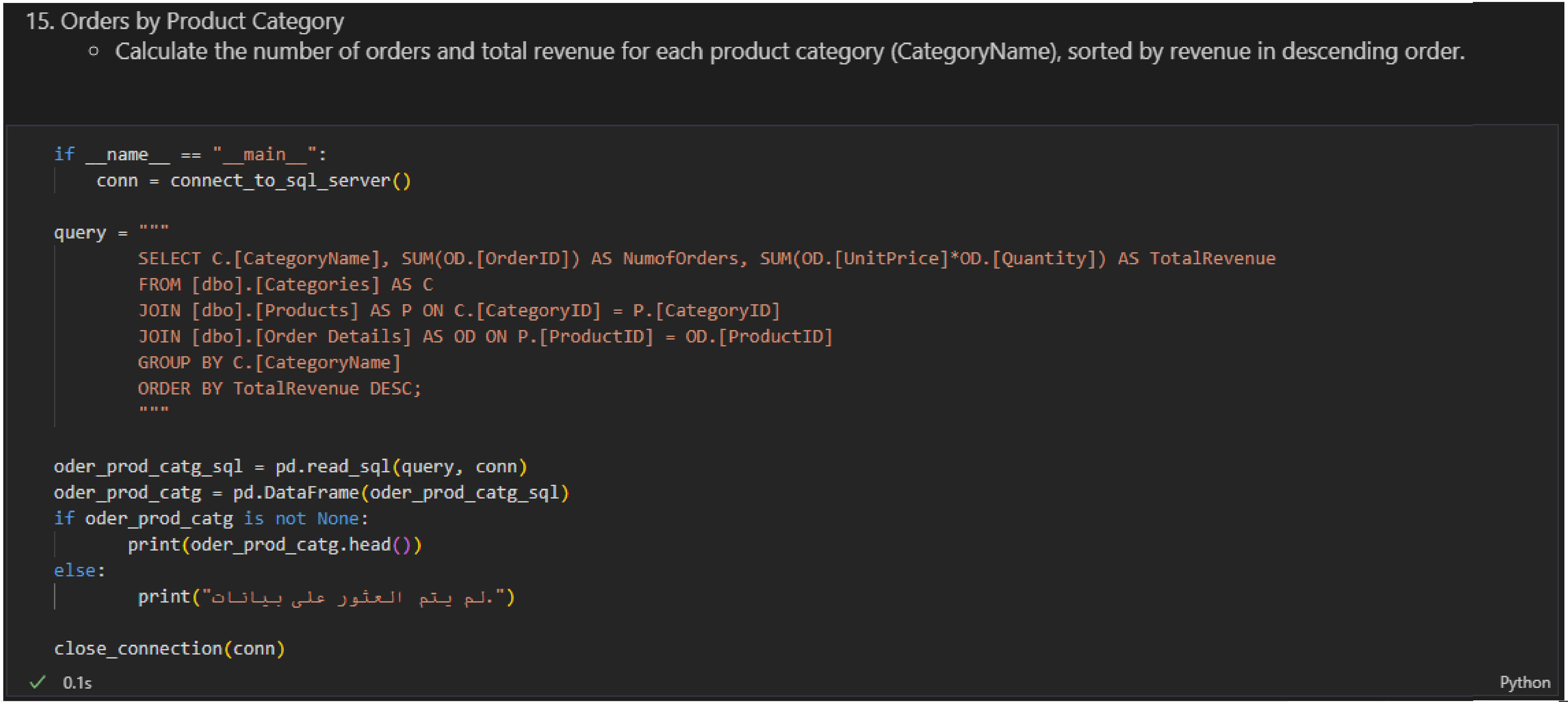
Question 15

Orders by Product Category

Calculate the number of orders and total revenue for each product category (CategoryName), sorted by revenue in descending order.

Orders by Product Category

SQL Query & Python Code



15. Orders by Product Category

- Calculate the number of orders and total revenue for each product category (CategoryName), sorted by revenue in descending order.

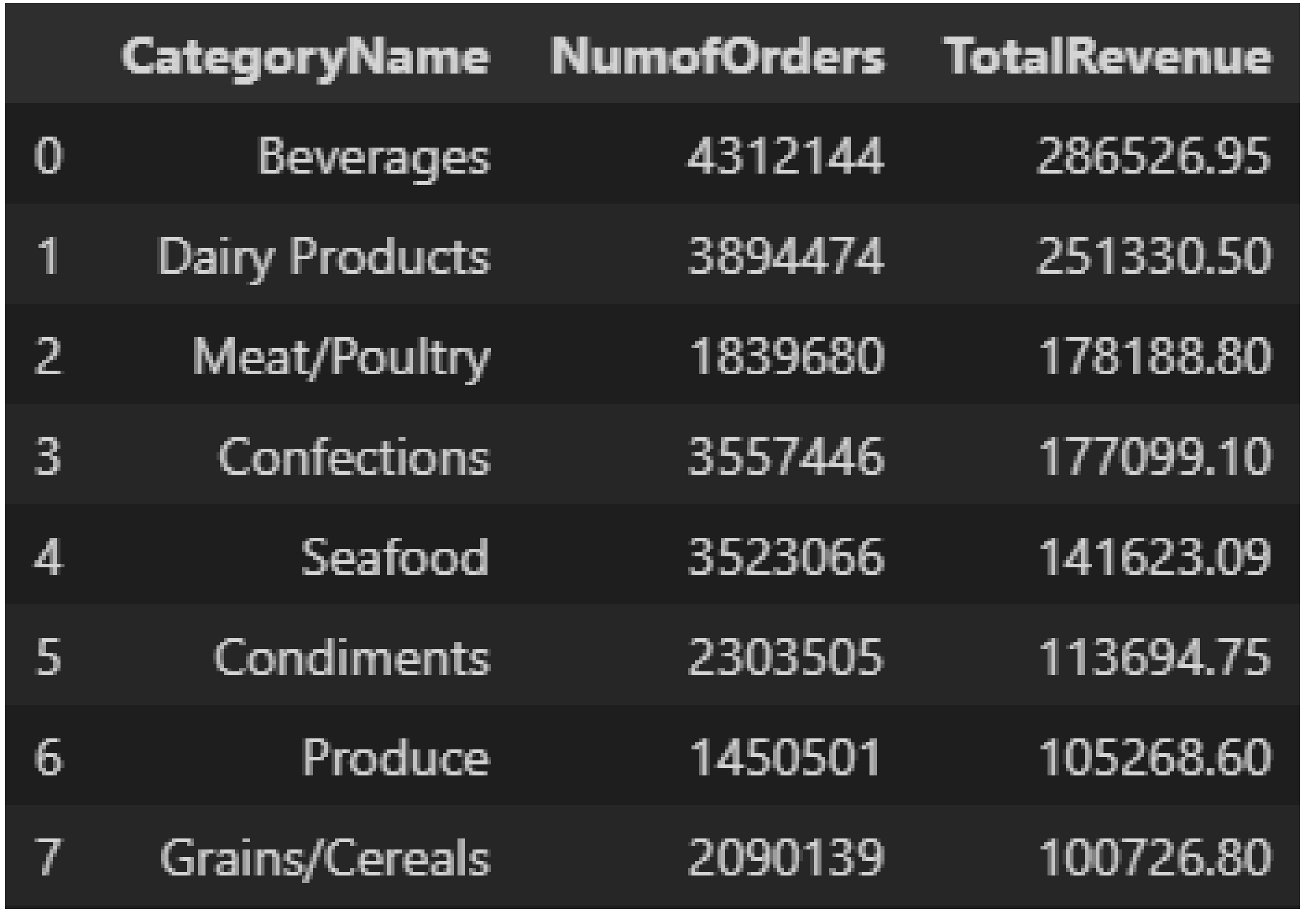
```python
if __name__ == "__main__":
    conn = connect_to_sql_server()

query = """
        SELECT C.[CategoryName], SUM(OD.[OrderID]) AS NumofOrders, SUM(OD.[UnitPrice]*OD.[Quantity]) AS TotalRevenue
        FROM [dbo].[Categories] AS C
        JOIN [dbo].[Products] AS P ON C.[CategoryID] = P.[CategoryID]
        JOIN [dbo].[Order Details] AS OD ON P.[ProductID] = OD.[ProductID]
        GROUP BY C.[CategoryName]
        ORDER BY TotalRevenue DESC;
        """

oder_prod_catg_sql = pd.read_sql(query, conn)
oder_prod_catg = pd.DataFrame(oder_prod_catg_sql)
if oder_prod_catg is not None:
       print(oder_prod_catg.head())
else:
        print("لم يتم العثور على بيانات.")

close_connection(conn)
```

✓ 0.1s

Python

Orders by Product Category

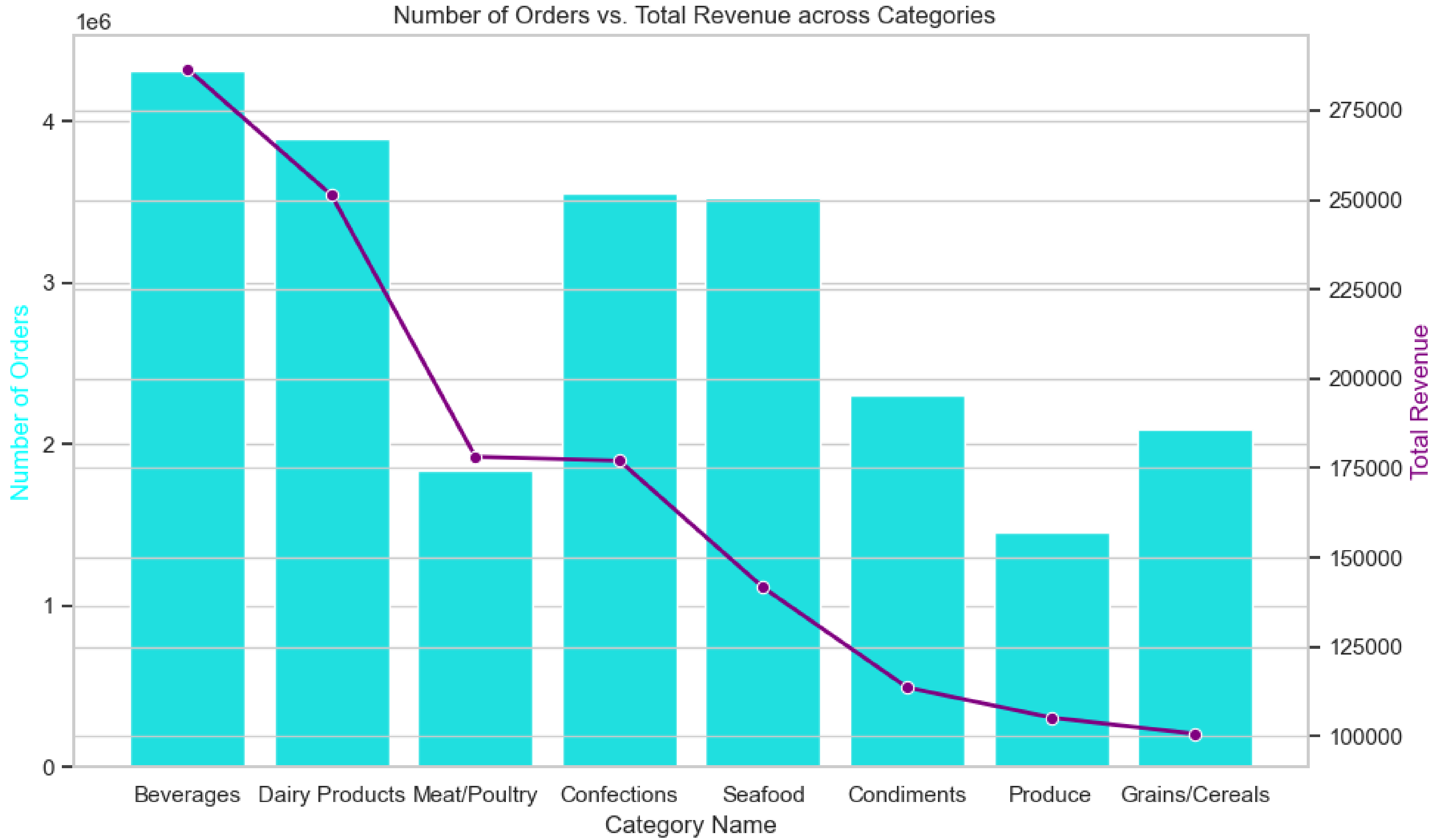
Data Generated



	CategoryName	NumofOrders	TotalRevenue
0	Beverages	4312144	286526.95
1	Dairy Products	3894474	251330.50
2	Meat/Poultry	1839680	178188.80
3	Confections	3557446	177099.10
4	Seafood	3523066	141623.09
5	Condiments	2303505	113694.75
6	Produce	1450501	105268.60
7	Grains/Cereals	2090139	100726.80

Orders by Product Category

Data Visualization



Thank You

Marwa Aboelenin

+1060664560

marwamostafa92@hotmail.com

NTI – National
Telecommunication Institute