

Project Report: Analyzing Amazon Sales Data

BY - Trijeta Ghosh

ROLE - Data Analytics Intern

DATE - 25-03-2024

Introduction:

The project aims to analyse sales data from Amazon to gain insights into sales performance, identify opportunities for growth and optimization, and make informed business decisions. By leveraging data analysis techniques, we seek to understand sales trends, profitability factors, and regional variations in sales performance.

Acknowledgements:

We acknowledge the contributions of the Unified Mentor team and express gratitude for their guidance and support throughout the project.

This report serves as a foundational analysis of Amazon sales data, providing a roadmap for further exploration and actionable insights to drive business success.

Method:

**1. Data Import and Structure:**

The Amazon sales data is imported from a CSV file into a Pandas DataFrame named sale data.

Initial exploration includes accessing column names, checking for missing values, and examining the dataset's structure.

**2. Data Integrity Check:**

The number of rows and columns in the dataset is determined to assess its size.

Missing values are identified in each column, ensuring data integrity before analysis.

**3. Summary Statistics:**

Descriptive statistics are generated for numerical columns, providing insights into data distribution and central tendencies.

**4. Profit Margin Calculation:**

Profit margins are calculated for each entry in the dataset, offering insights into profitability factors.

**5. Date Conversion:**

Date columns ('Order Date' and 'Ship Date') are converted to datetime format to facilitate temporal analysis.

**6. Average Unit Price by Region:**

The average unit price is computed for each region, allowing for region-specific pricing insights.

Key Points:

**1. Data Import and Structure:** The project begins by importing Amazon sales data from a CSV file into a Pandas DataFrame. Accessing column names provides insight into the dataset's structure.

**2. Data Integrity Check:** The number of rows and columns in the dataset is determined, and missing values are checked to ensure data integrity before analysis.

**3. Summary Statistics:** Descriptive statistics are computed for numerical columns, offering insights into data distribution and central tendencies.

**4. Profit Margin Calculation**: Profit margins are calculated for each entry in the dataset, providing insights into profitability factors.

**5. Date Conversion:** Date columns ('Order Date' and 'Ship Date') are converted to datetime format to facilitate temporal analysis.

**6. Average Unit Price by Region**: The average unit price is calculated for each region, allowing for region-specific pricing insights.

**7. Initial Analysis Insights:** The initial analysis highlights sales performance, profitability factors, and regional variations, setting the stage for further exploration.

**8. Recommendations:** Future analysis could include predictive analytics, customer segmentation, and incorporation of external datasets for a comprehensive understanding of sales dynamics.

**9. Data-Driven Decision Making:** The project underscores the importance of data-driven approaches in informing strategic decision-making and driving business growth.

Recommendations:

* Conduct further analysis to identify key drivers of profitability and optimize pricing strategies.
* Explore regional sales trends to tailor marketing and sales strategies to specific geographic markets.
* Implement data visualization techniques to present findings in a visually compelling manner for stakeholders.
* Continuously monitor sales data and performance metrics to adapt strategies and capitalize on emerging opportunities.

Code:

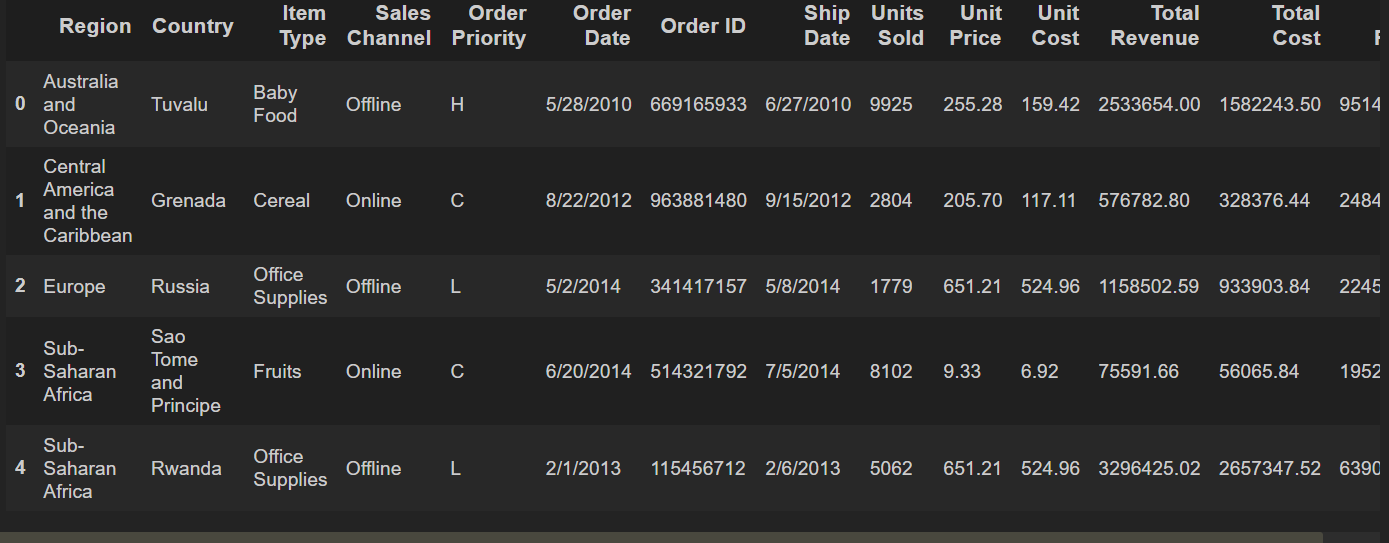
**This code snippet imports the Pandas library as `pd` and the Matplotlib.pyplot module as `plt`.**

import pandas as pd

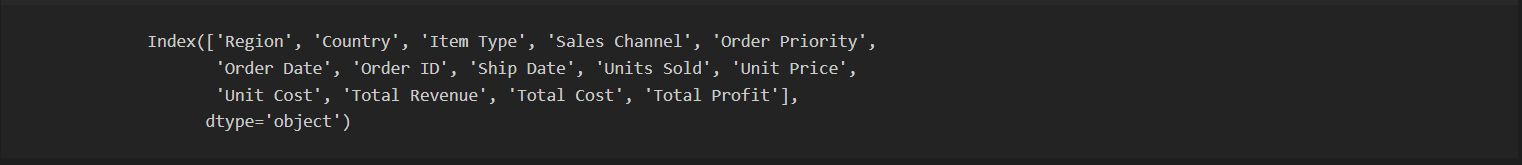
import matplotlib.pyplot as plt

**This code reads the Amazon sales data from a CSV file named "Amazon Sales data - Amazon Sales data.csv" into a Pandas DataFrame named `sale\_data` and displays the first few rows of the DataFrame using the `head()` function.**

sale\_data=pd.read\_csv("Amazon Sales data - Amazon Sales data.csv")

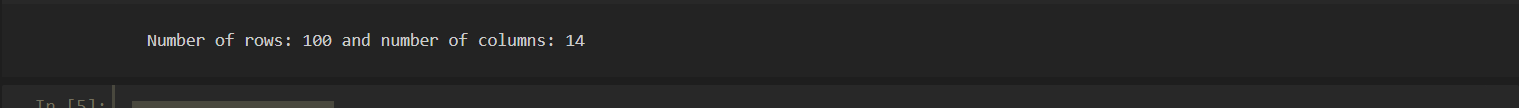
sale\_data.head()

**Accessing column names**

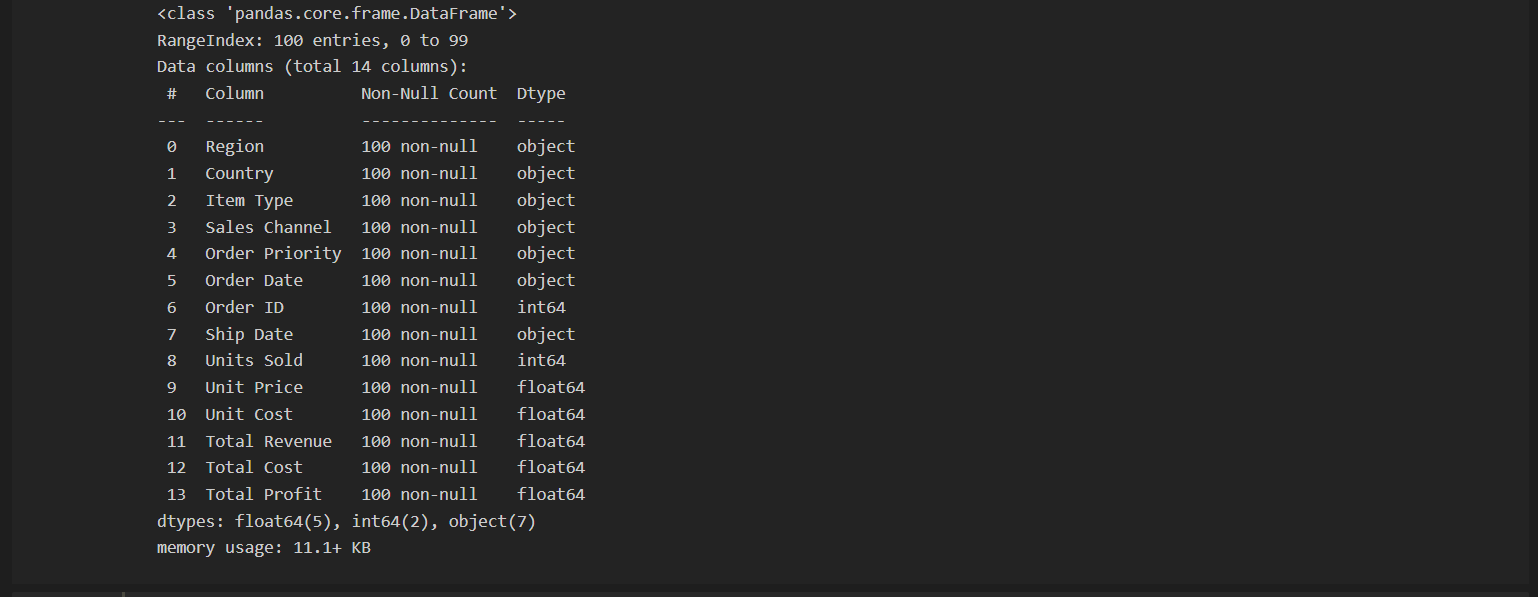
sale\_data.columns

**This code snippet prints the number of rows and columns in the `sale\_data` DataFrame using the `shape` attribute.**

print(f"Number of rows: {sale\_data.shape[0]} and number of columns: {sale\_data.shape[1]}")



**This code snippet provides information about the `sale\_data` DataFrame, including the data types of each column, the number of non-null values, and memory usage.**

sale\_data.info()

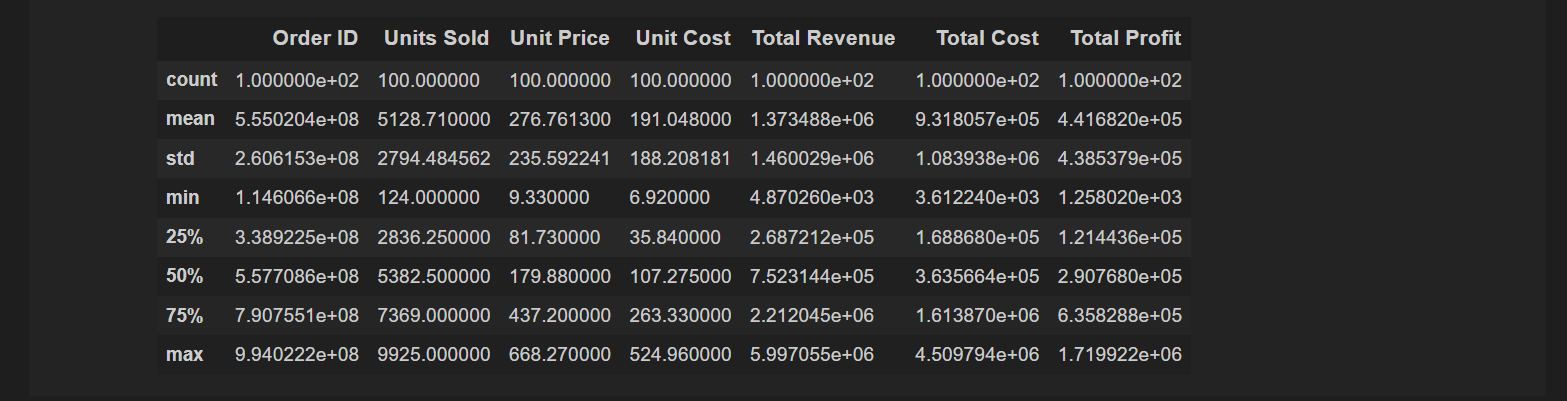
**Check for missing value**

sale\_data.isnull().sum()



**Summary statistics**

sale\_data.describe()



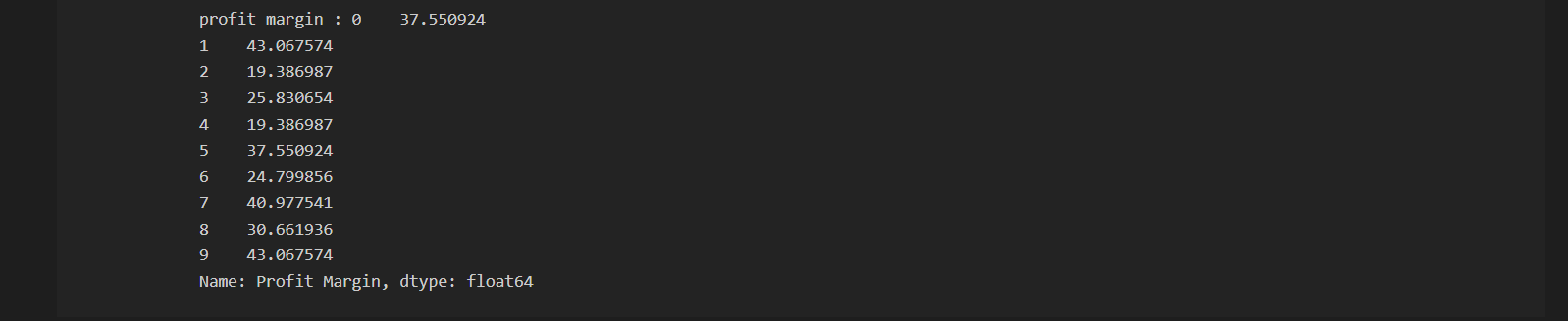
**This code snippet calculates the profit margin for each entry in the `sale\_data` DataFrame by dividing the total profit by the total revenue and multiplying by 100. It then displays the profit margin for the first ten entries using the `head()` function.**

check profit margin

sale\_data['Profit Margin'] = (sale\_data['Total Profit'] / sale\_data['Total Revenue']) \* 100

profit\_margin\_head = sale\_data['Profit Margin'].head(10)

print("profit margin :",profit\_margin\_head)



**This code snippet converts the 'Order Date' and 'Ship Date' columns in the `sale\_data` DataFrame to datetime format using the `pd.to\_datetime()` function.**

sale\_data['Order Date'] = pd.to\_datetime(sale\_data['Order Date'])

sale\_data['Ship Date'] = pd.to\_datetime(sale\_data['Ship Date'])

**Average unit price by region**

avg\_unit\_price\_by\_region = sale\_data.groupby('Region')['Unit Price'].mean()

print(avg\_unit\_price\_by\_region)



**This code snippet generates a bar chart depicting the average unit price by region. The chart is created using the `plot()` function with the `kind='bar'` parameter. It specifies the color of the bars as orange and sets the edge color to black. The x-axis represents different regions, while the y-axis represents the average unit price. Additionally, labels, a title, legend, and x-axis rotation are added for clarity. Finally, the chart is displayed using `plt.show()`.**

Bar chart of average unit price by region

avg\_unit\_price\_by\_region.plot(kind='bar',color='orange',edgecolor='black')

plt.xlabel('Region')

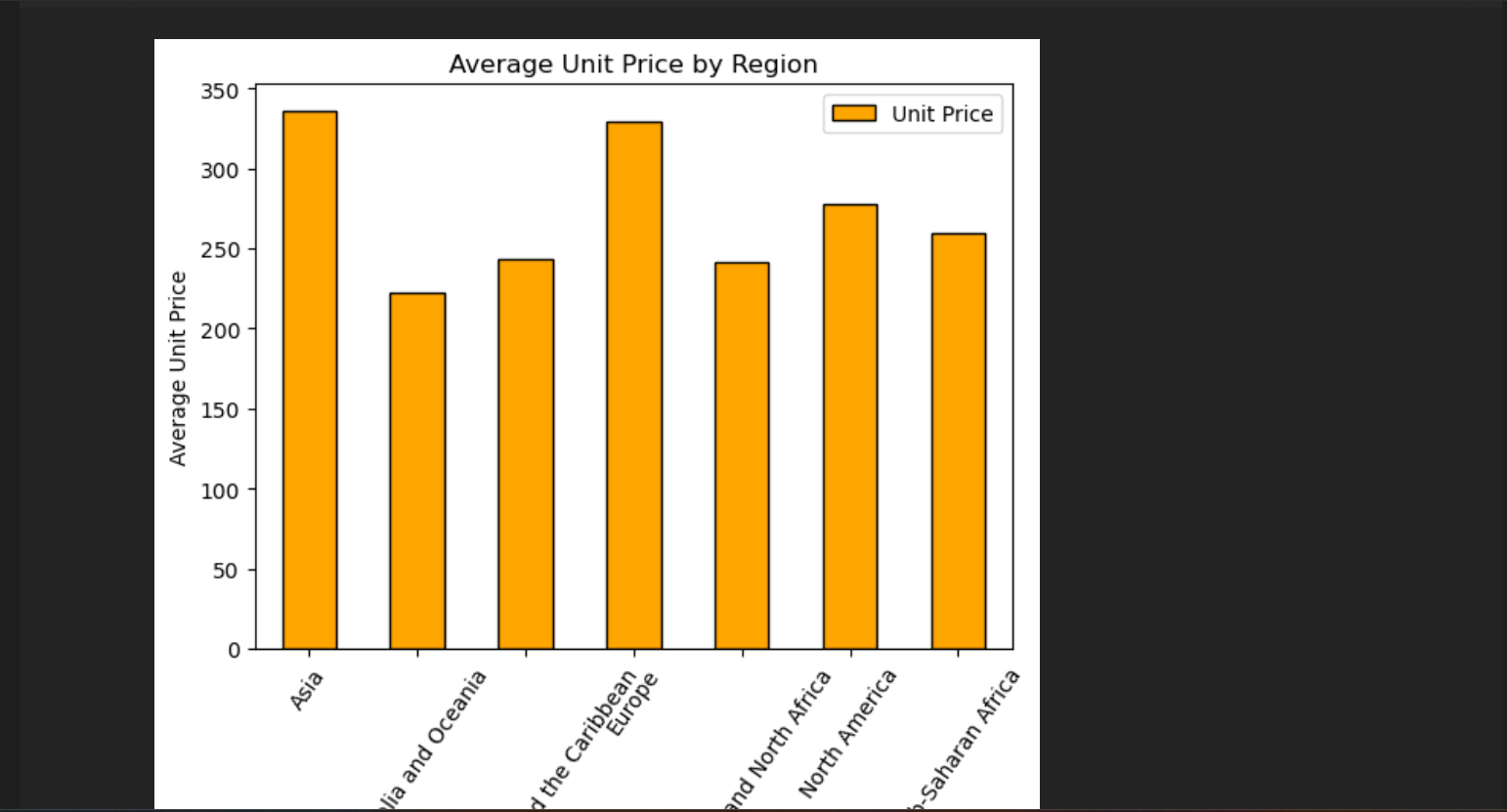
plt.ylabel('Average Unit Price')

plt.title('Average Unit Price by Region')

plt.legend()

plt.xticks(rotation=55)

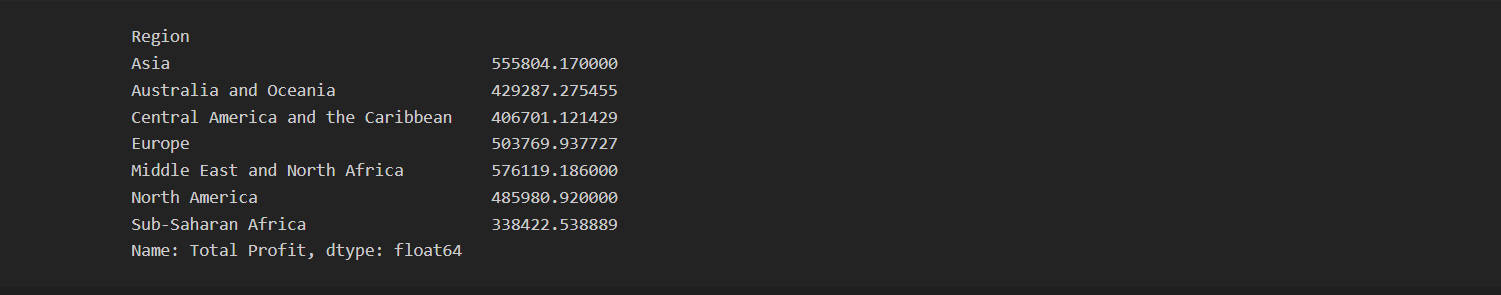
plt.show()



**This code snippet calculates the average total profit by region in the `sale\_data` DataFrame using the `groupby()` function to group the data by region and the `mean()` function to compute the average profit for each group. The resulting Series `avg\_total\_profit\_by\_region` contains the average profit values for each region.**

avg\_total\_profit\_by\_region = sale\_data.groupby('Region')['Total Profit'].mean()

avg\_total\_profit\_by\_region



**This code snippet creates a line chart illustrating the regions with the highest average total profit. It first selects the top regions with the largest average total profit using the `nlargest()` function. Then, it plots the data using the `plot()` function with `kind='line'` to generate a line chart. The line color is set to red, and markers are added to data points for emphasis. Additional attributes such as labels, title, rotation for x-axis labels, and gridlines are included for clarity. Finally, the chart is displayed using `plt.show()`.**

regions = avg\_total\_profit\_by\_region.nlargest()

plt.figure(figsize=(4, 4))

regions.plot(kind='line', color='r', marker='o', markerfacecolor='black', markeredgewidth=1.5)

plt.xlabel('Region')

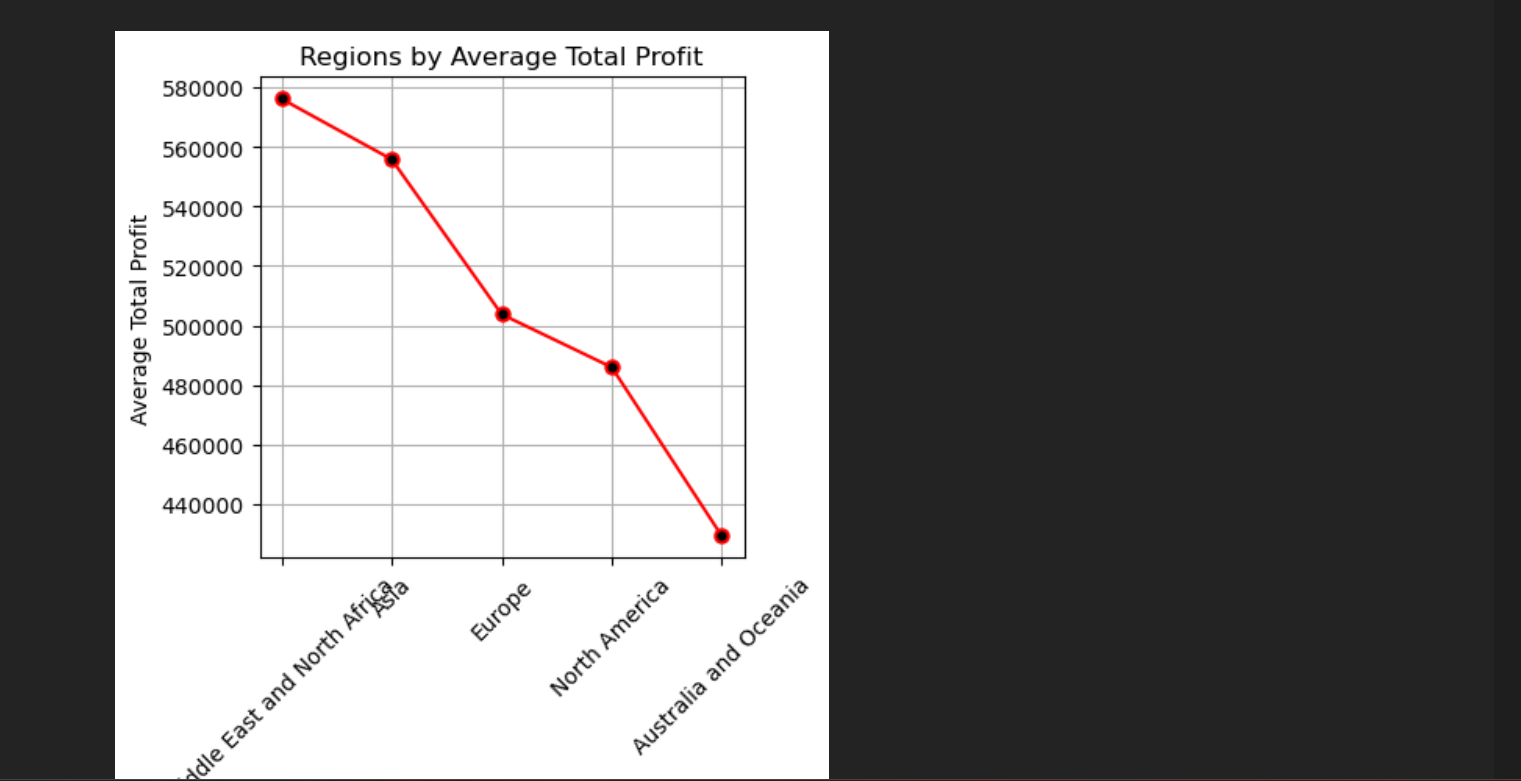
plt.ylabel('Average Total Profit')

plt.title('Regions by Average Total Profit')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()



**This code snippet generates a histogram to visualize the distribution of units sold in the `sale\_data` DataFrame. It uses the `hist()` function to create the histogram, specifying the column 'Units Sold' as the data to be plotted. The color of the bars is set to blue, and the edge color is set to black for better visibility. Additionally, the number of bins is set to 10 to control the granularity of the distribution. Labels for the x-axis, y-axis, and title are added for clarity. Finally, the histogram is displayed using `plt.show()`.**

plt.figure(figsize=(4, 4))

# Create a histogram of Units Sold

plt.hist(sale\_data['Units Sold'], color='b',edgecolor='black', bins=10)

# Labeling and title

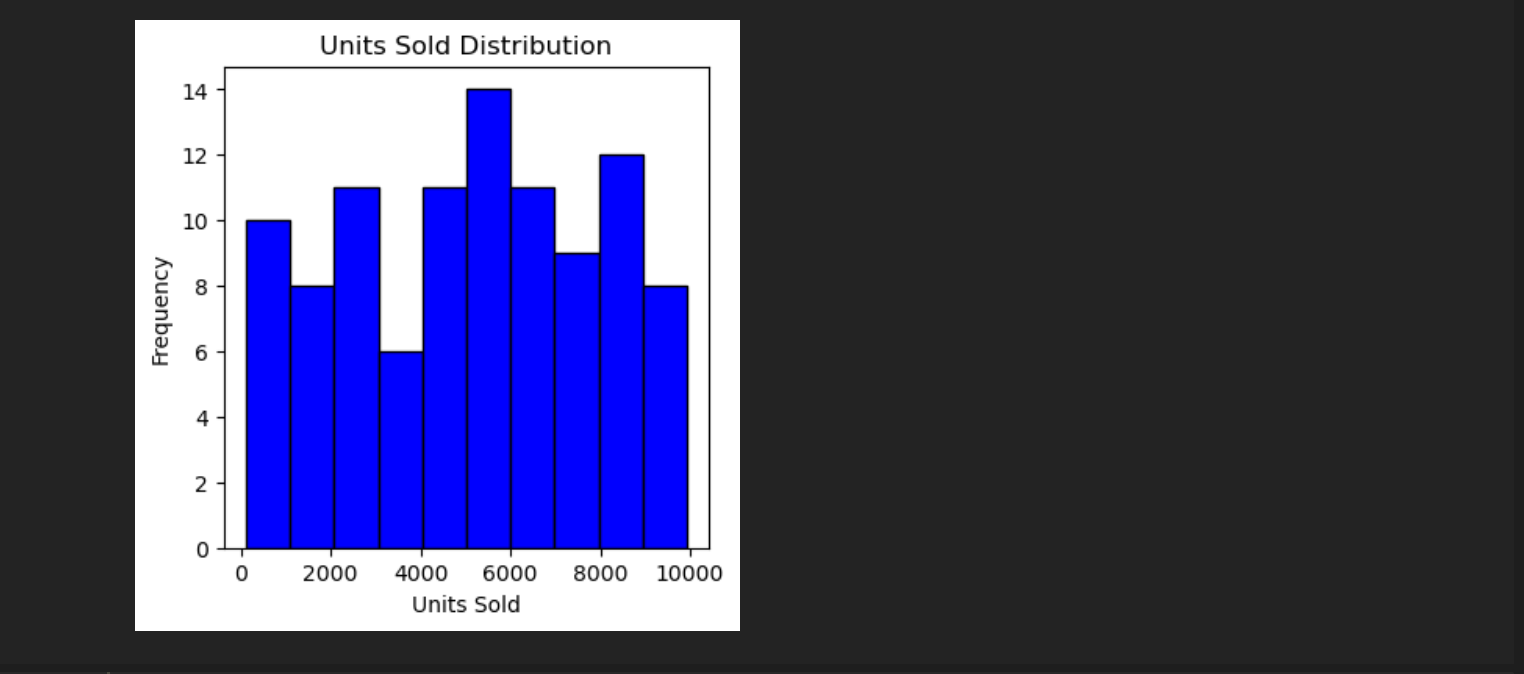
plt.xlabel('Units Sold')

plt.ylabel('Frequency')

plt.title('Units Sold Distribution')

# Show the plot

plt.show()



**This code snippet generates a bar chart to visualize the top 10 regions by total revenue in the `sale\_data` DataFrame. It first sorts the DataFrame by 'Total Revenue' in descending order to identify the top revenue-generating regions. Then, it selects the top 10 regions and their corresponding total revenue. Next, it assigns colors to the bars using a gradient from light red to dark red. The bar chart is created using the `bar()` function, specifying the region names as the x-axis labels, total revenue as the y-axis values, and the assigned colors for visualization. Labels for the x-axis, y-axis, and title are added for clarity, and the chart is displayed using `plt.show()`.**

import numpy as np

sorted\_sale\_data = sale\_data.sort\_values(by='Total Revenue', ascending=False)

total\_revenue = sorted\_sale\_data['Total Revenue'].head(10)

colors = plt.cm.Reds(np.linspace(0.2, 1, len(total\_revenue)))

plt.figure(figsize=(4, 4))

plt.bar(sorted\_sale\_data['Region'].head(10), total\_revenue, color=colors, edgecolor='black')

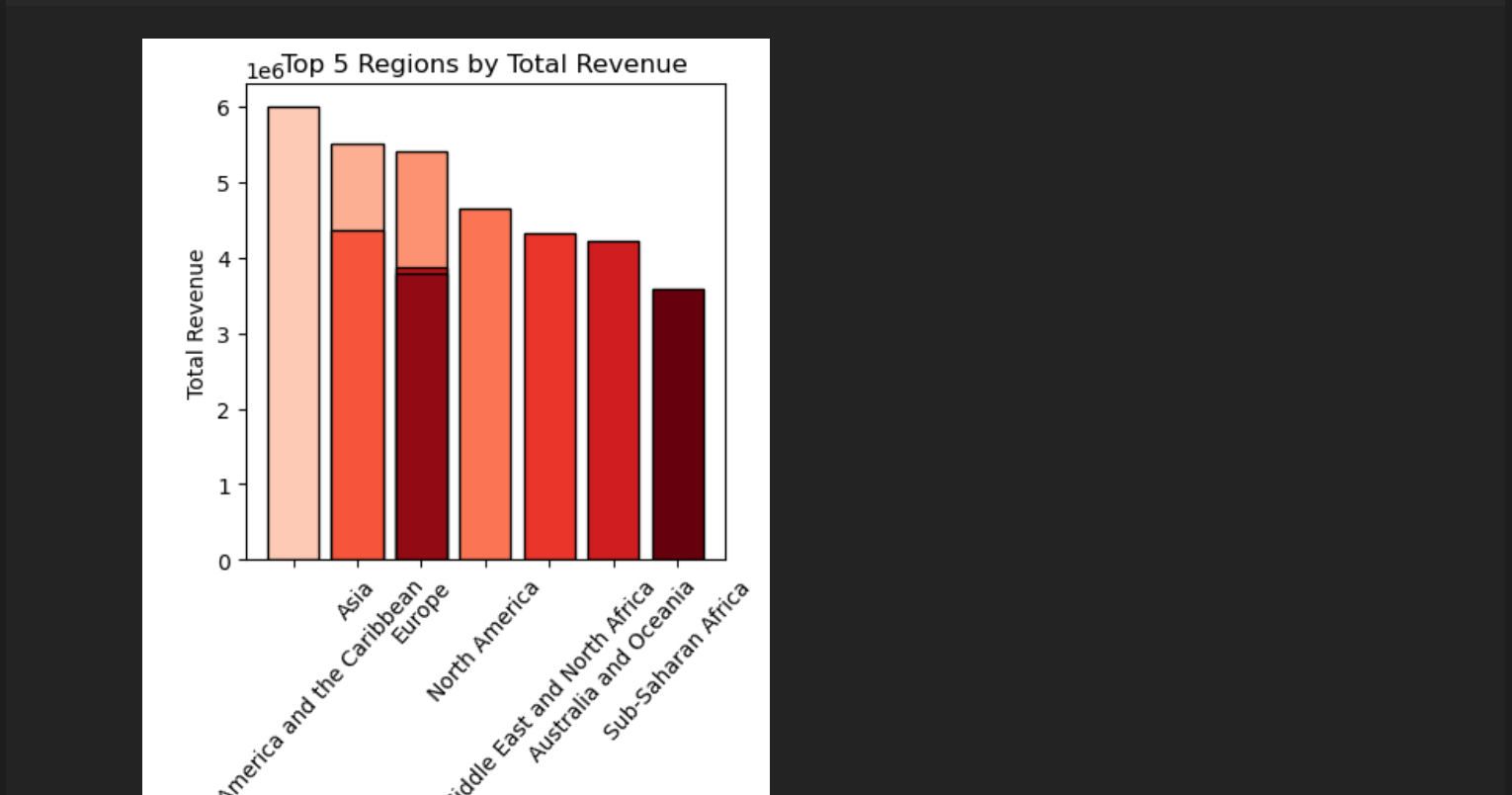
plt.xlabel('Region')

plt.ylabel('Total Revenue')

plt.title('Top 5 Regions by Total Revenue')

plt.xticks(rotation=48)

plt.show()



**This code snippet creates a pivot table from the `sale\_data` DataFrame, aggregating the total revenue, total cost, and total profit values based on the 'Region' column. The `pivot\_table()` function from the Pandas library is used to perform this operation. It specifies the values to be aggregated ('Total Revenue', 'Total Cost', 'Total Profit'), the index column ('Region'), and the aggregation function ('sum' to calculate the sum of values). The resulting pivot table aggregates the data by region, providing the total revenue, total cost, and total profit for each region. Finally, the pivot table is displayed using the `print()` function.**

# privote table

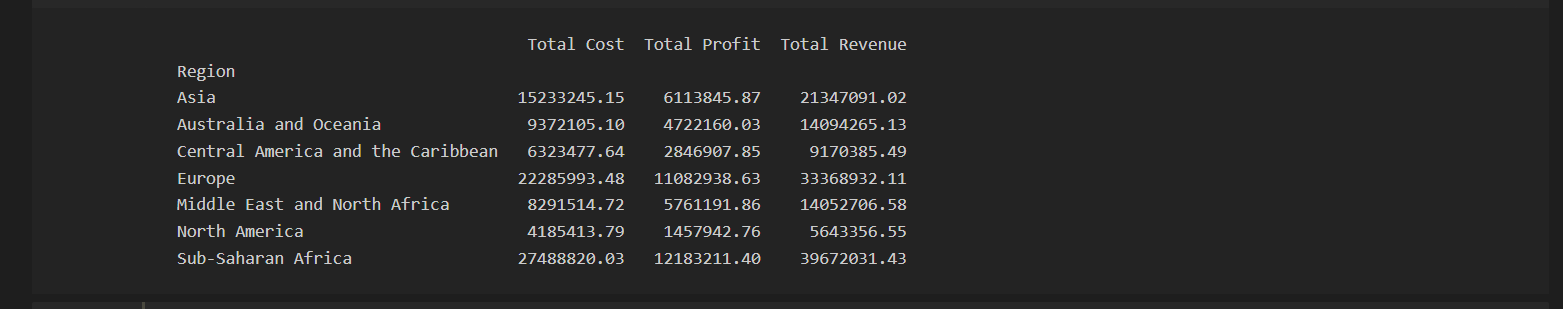
pivot\_table = pd.pivot\_table(sale\_data,

values=['Total Revenue', 'Total Cost', 'Total Profit'],

index='Region',

aggfunc='sum')

print(pivot\_table)



**Pivot table of 'Units Sold', 'Unit Price', 'Unit Cost'**

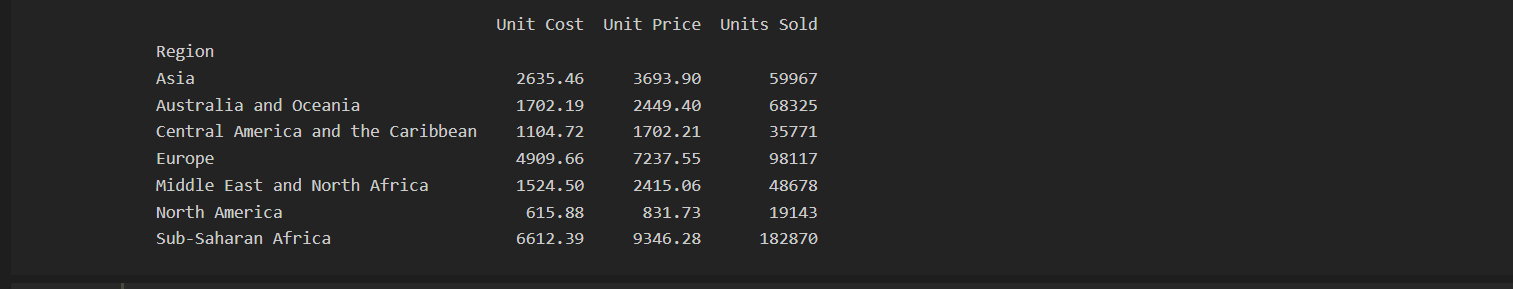
table = pd.pivot\_table(sale\_data,

values=['Units Sold', 'Unit Price', 'Unit Cost'],

index='Region',

aggfunc='sum')

print(table)



**Pivot table of ‘Total Revenue’, ‘Item Type’, ‘Sales Channnel ‘**

table1 = pd.pivot\_table(sale\_data,

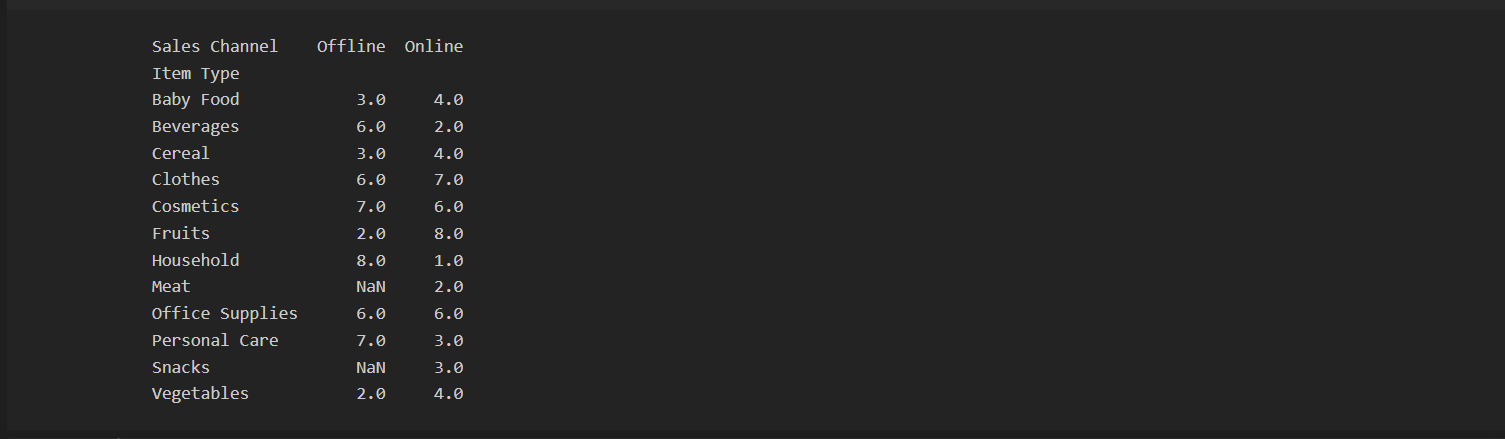
values='Total Revenue',

index='Item Type',

columns='Sales Channel',

aggfunc='count')

print(table1)



**This code snippet generates a pie chart to visualize the distribution of total sales by item type in the `sale\_data` DataFrame. It first groups the data by 'Item Type' and calculates the total revenue for each item type using the `groupby()` function and the `sum()` aggregation. Then, it creates a pie chart using the `pie()` function, specifying the total sales values as the data, item types as labels, and the autopct parameter to display the percentage contribution of each item type. The `startangle` parameter is set to 140 to adjust the starting angle of the pie chart. Additional attributes such as title, legend, and axis equalization are included for clarity, and the chart is displayed using `plt.show()`.**

total\_sales\_by\_item = sale\_data.groupby('Item Type')['Total Revenue'].sum()

plt.figure(figsize=(8,8))

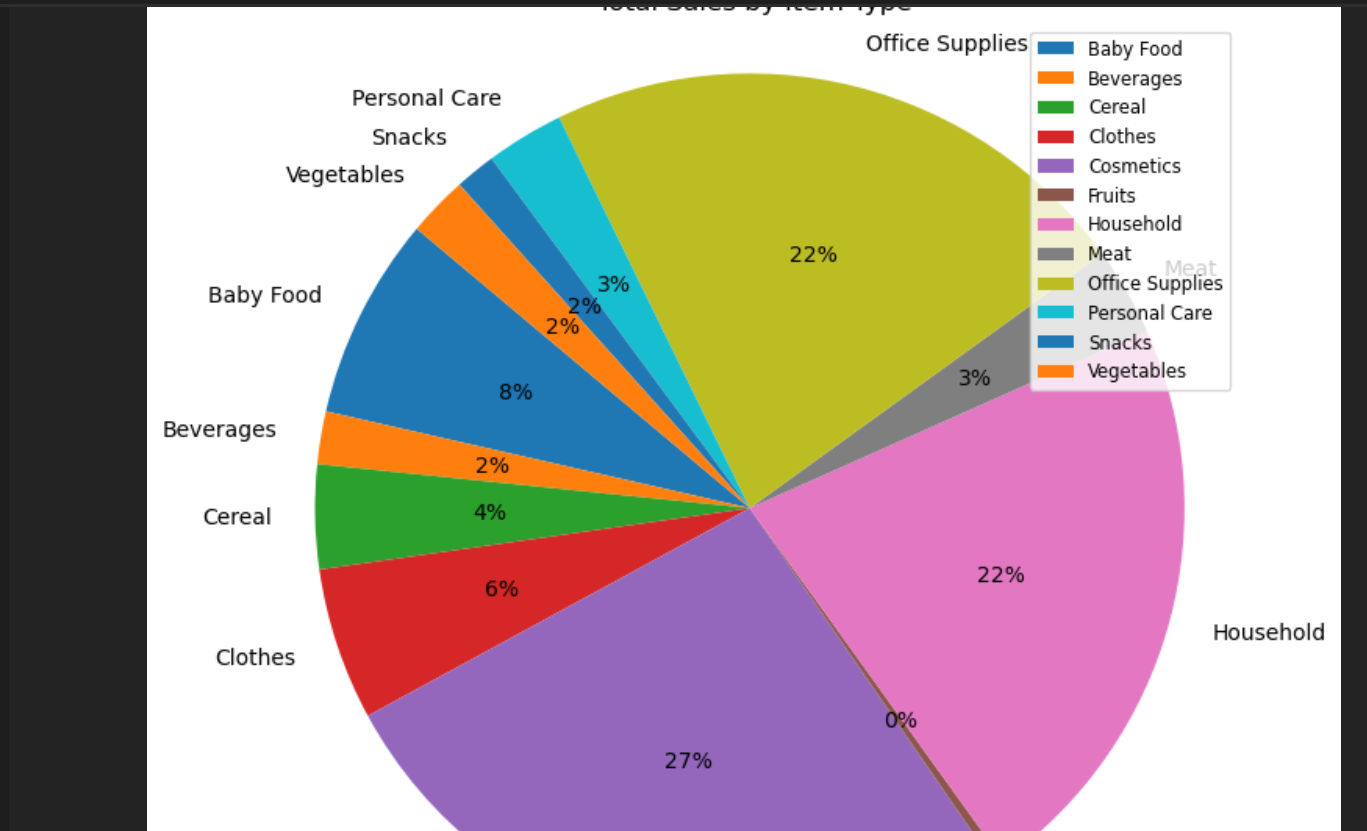
plt.pie(total\_sales\_by\_item, labels=total\_sales\_by\_item.index, autopct='%0.0f%%', startangle=140)

plt.title('Total Sales by Item Type')

plt.legend(loc='upper right',fontsize='small')

plt.axis('equal')

plt.show()



**This code snippet creates a pivot table from the `sale\_data` DataFrame using the `pivot\_table()` function, aggregating the total profit values based on the 'Region' and 'Order Priority' columns. Then, it generates a bar chart to visualize the total profit by region and sales channel order priority. The `plot()` function is used with `kind='bar'` to create the bar chart, specifying the region as the x-axis labels, total profit as the y-axis values, and different order priorities as separate bars. Additional attributes such as figure size, axis labels, title, legend, rotation of x-axis labels, and gridlines are included for clarity. Finally, the chart is displayed using `plt.show()`.**

pivot\_table = sale\_data.pivot\_table(values='Total Profit', index='Region', columns='Order Priority', aggfunc='sum')

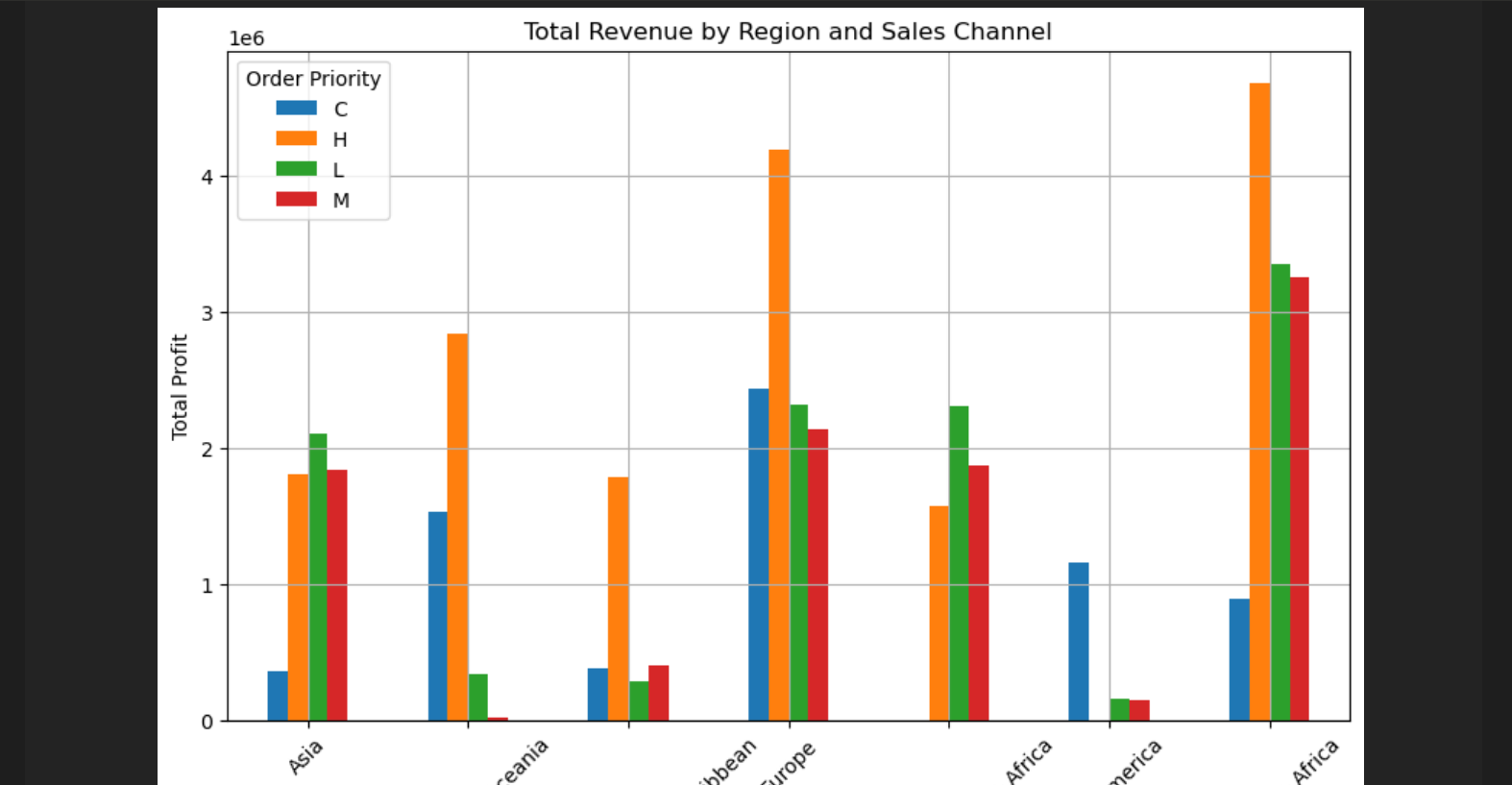
pivot\_table.plot(kind='bar', figsize=(10, 6))

plt.xlabel('Region')

plt.ylabel('Total Profit')

plt.title('Total Revenue by Region and Sales Channel')

plt.xticks(rotation=45)



plt.legend(title='Order Priority')

plt.grid()

plt.show()

**This code snippet creates a pivot table from the `sale\_data` DataFrame using the `pivot\_table()` function, aggregating the total revenue values based on the 'Region' and 'Sales Channel' columns. Then, it generates a bar chart to visualize the total revenue by region and sales channel. The `plot()` function is used with `kind='bar'` to create the bar chart, specifying the region as the x-axis labels, total revenue as the y-axis values, and different sales channels as separate bars. Additional attributes such as figure size, axis labels, title, legend, rotation of x-axis labels, and gridlines are included for clarity. Finally, the chart is displayed using `plt.show()`.**

pivot\_table = sale\_data.pivot\_table(values='Total Revenue', index='Region', columns='Sales Channel', aggfunc='sum')

pivot\_table.plot(kind='bar', figsize=(8, 6))

plt.xlabel('Region')

plt.ylabel('Total Revenue')

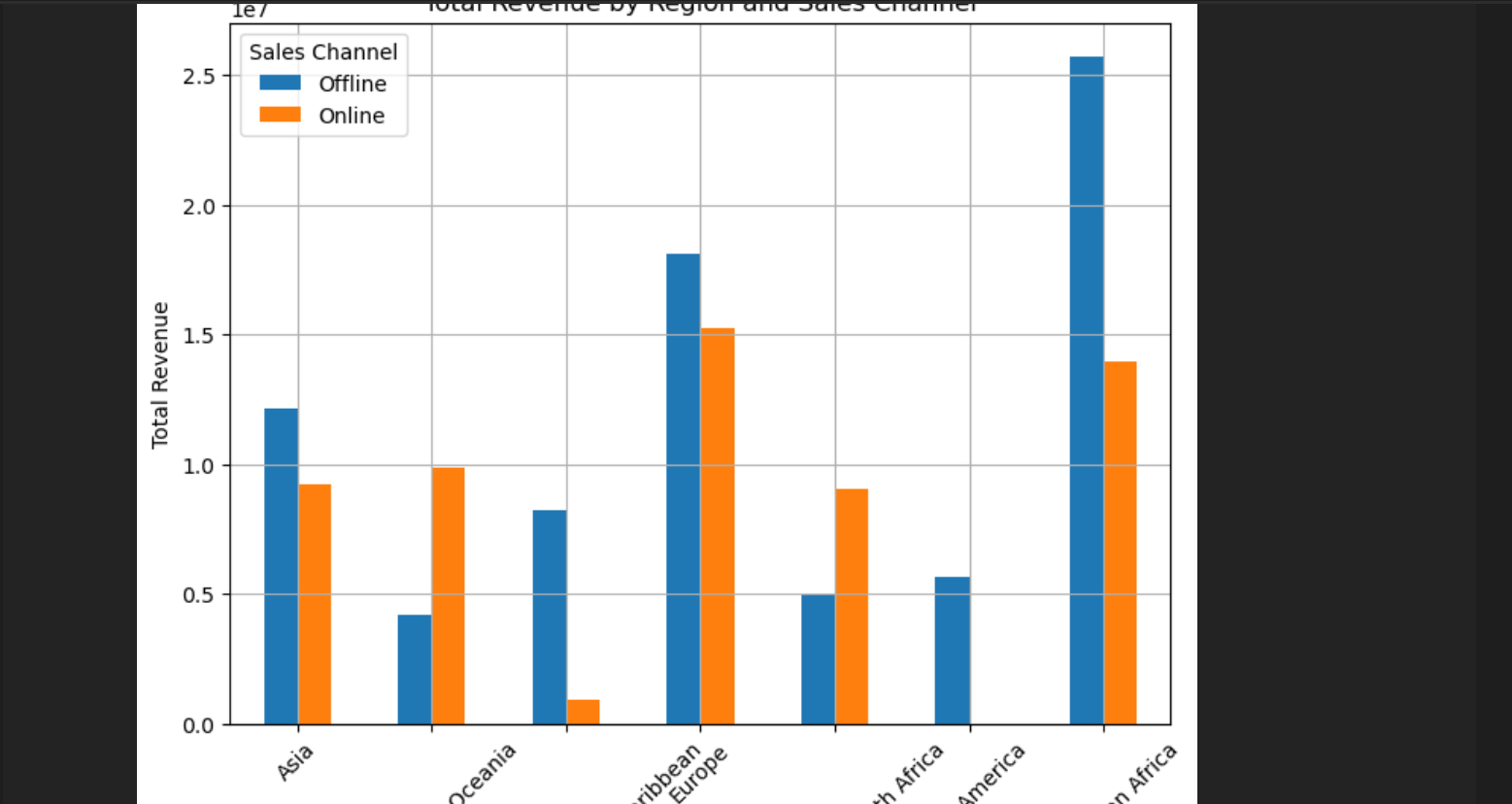
plt.title('Total Revenue by Region and Sales Channel')

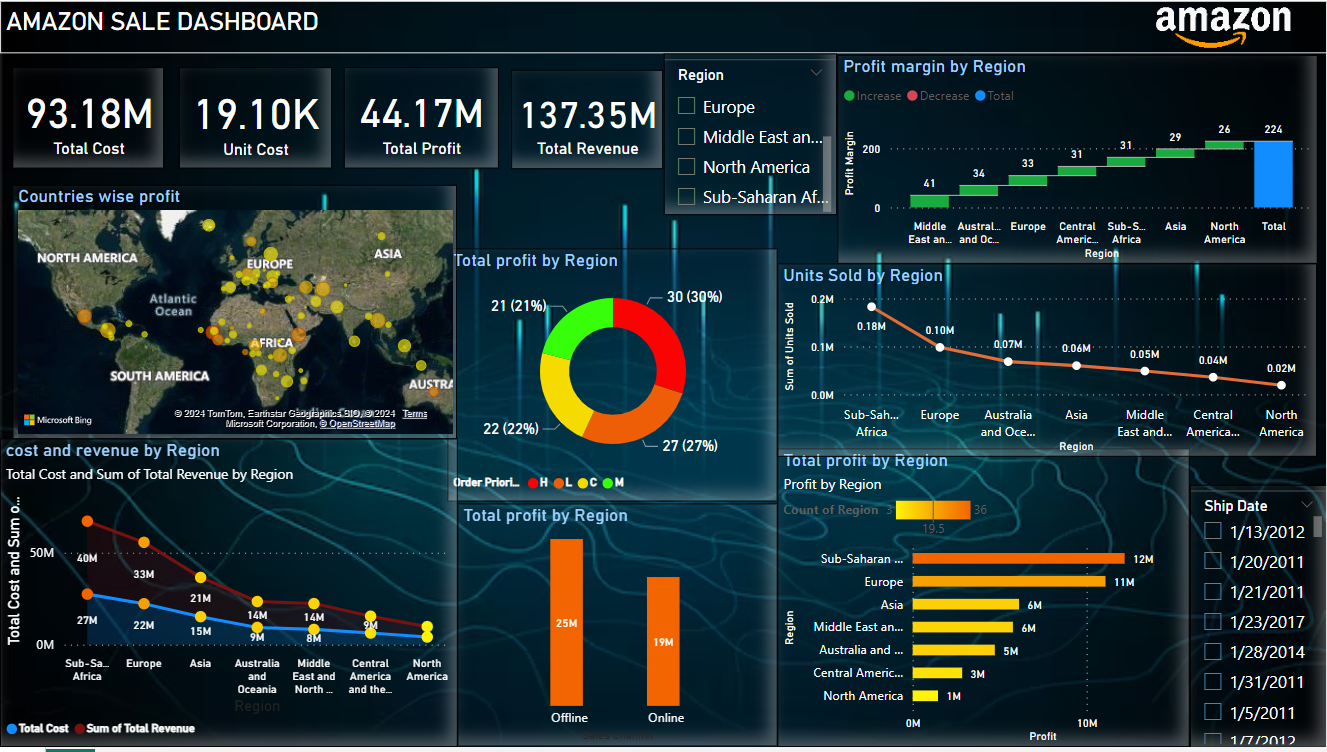
plt.xticks(rotation=45)

plt.legend(title='Sales Channel')

plt.grid()

plt.show()



DashBoard:

Future Work:

Explore predictive analytics techniques to forecast future sales trends and anticipate market demand.

Incorporate external datasets, such as economic indicators or competitor data, for a comprehensive analysis.

Conduct customer segmentation analysis to personalize marketing efforts and enhance customer engagement.

Conclusion:

The initial analysis of Amazon sales data has provided valuable insights into sales performance, profitability factors, and regional variations. Further exploration and analysis can uncover additional insights, inform strategic decision-making, and drive business growth. This project highlights the importance of data-driven approaches in understanding and optimizing sales operations.