sales Analysis

import Necessary Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
```

12 month sales dataset

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

# Get the list of uploaded files from the 'uploaded' variable
uploaded_files = uploaded.keys()
# Create an empty list to store dataframes
all_months_data = []
# Read each uploaded file and append to the list
for file in uploaded_files:
  if file.endswith('.xlsx'):
    df_temp = pd.read_excel(file)
    all_months_data.append(df_temp)

# Concatenate all dataframes into a single dataframe
all_data = pd.concat(all_months_data)

# Assign the combined dataframe to df
df = all_data
```

Choose Files 12 files

- **sales_April_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34185 bytes, last modified: 8/13/2025 - 100% done
- **sales_August_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34250 bytes, last modified: 8/13/2025 - 100% done
- **sales_December_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34214 bytes, last modified: 8/13/2025 - 100% done
- **sales_February_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34047 bytes, last modified: 8/13/2025 - 100% done
- **sales_January_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34208 bytes, last modified: 8/13/2025 - 100% done
- **sales_July_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34299 bytes, last modified: 8/13/2025 - 100% done
- **sales_June_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34165 bytes, last modified: 8/13/2025 - 100% done
- **sales_March_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34340 bytes, last modified: 8/13/2025 - 100% done
- **sales_May_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34435 bytes, last modified: 8/13/2025 - 100% done
- **sales_November_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34309 bytes, last modified: 8/13/2025 - 100% done
- **sales_October_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34231 bytes, last modified: 8/13/2025 - 100% done
- **sales_September_2024.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 34170 bytes, last modified: 8/13/2025 - 100% done

```
Saving sales_April_2024.xlsx to sales_April_2024 (1).xlsx
Saving sales_August_2024.xlsx to sales_August_2024 (1).xlsx
Saving sales_December_2024.xlsx to sales_December_2024 (1).xlsx
Saving sales_February_2024.xlsx to sales_February_2024 (1).xlsx
Saving sales_January_2024.xlsx to sales_January_2024 (1).xlsx
Saving sales_July_2024.xlsx to sales_July_2024 (1).xlsx
Saving sales_June_2024.xlsx to sales_June_2024 (1).xlsx
Saving sales_March_2024.xlsx to sales_March_2024 (1).xlsx
Saving sales_May_2024.xlsx to sales_May_2024 (1).xlsx
Saving sales_November_2024.xlsx to sales_November_2024 (1).xlsx
Saving sales_October_2024.xlsx to sales_October_2024 (1).xlsx
Saving sales_September_2024.xlsx to sales_September_2024 (1).xlsx
User uploaded file "sales_April_2024 (1).xlsx" with length 34185 bytes
User uploaded file "sales_August_2024 (1).xlsx" with length 34250 bytes
User uploaded file "sales_December_2024 (1).xlsx" with length 34214 bytes
User uploaded file "sales_February_2024 (1).xlsx" with length 34047 bytes
User uploaded file "sales_January_2024 (1).xlsx" with length 34208 bytes
User uploaded file "sales_July_2024 (1).xlsx" with length 34299 bytes
User uploaded file "sales_June_2024 (1).xlsx" with length 34165 bytes
User uploaded file "sales_March_2024 (1).xlsx" with length 34340 bytes
User uploaded file "sales_May_2024 (1).xlsx" with length 34435 bytes
User uploaded file "sales_November_2024 (1).xlsx" with length 34309 bytes
User uploaded file "sales_October_2024 (1).xlsx" with length 34231 bytes
User uploaded file "sales_September_2024 (1).xlsx" with length 34170 bytes
```

Read updated Dataframe

```
# Display the first few rows of the combined dataframe
display(df.head())
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	101500	Wired Headphones	3	11.99	04/07/24 17:03	9508 Jason Radial Apt. 611, Port Yolandashire,...
1	101501	Lightning Charging Cable	3	14.95	04/13/24 15:28	58919 Mallory Island, Lake Brian, RI 19544
2	101502	Lightning Charging Cable	1	14.95	04/10/24 19:55	8641 Roberts Inlet, North Paul, CA 04296
3	101503	Vareebadd Phone	3	400.00	04/18/24 04:07	57769 Powell Springs, Thompsonmouth, VA 63086
4	101504	AA Batteries (4-pack)	1	3.84	04/20/24 00:39	707 Brooke Estates Apt. 885, South Brianland, ...

clean the data

```
# Remove rows with NaN values
df.dropna(inplace=True)

# Display the first few rows of the dataframe after removing NaN values
display(df.head())
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	101500	Wired Headphones	3	11.99	04/07/24 17:03	9508 Jason Radial Apt. 611, Port Yolandashire,...
1	101501	Lightning Charging Cable	3	14.95	04/13/24 15:28	58919 Mallory Island, Lake Brian, RI 19544
2	101502	Lightning Charging Cable	1	14.95	04/10/24 19:55	8641 Roberts Inlet, North Paul, CA 04296
3	101503	Vareebadd Phone	3	400.00	04/18/24 04:07	57769 Powell Springs, Thompsonmouth, VA 63086
4	101504	AA Batteries (4-pack)	1	3.84	04/20/24 00:39	707 Brooke Estates Apt. 885, South Brianland, ...

add the Month Columns

```
all_data["Month"] = all_data["Order Date"].str[0:2]
all_data["Month"] = all_data["Month"].astype('int64')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	101500	Wired Headphones	3	11.99	04/07/24 17:03	9508 Jason Radial Apt. 611, Port Yolandashire,...	4
1	101501	Lightning Charging Cable	3	14.95	04/13/24 15:28	58919 Mallory Island, Lake Brian, RI 19544	4
2	101502	Lightning Charging Cable	1	14.95	04/10/24 19:55	8641 Roberts Inlet, North Paul, CA 04296	4

Next steps: Generate code with `all_data` View recommended plots New interactive sheet

Add the sales column

```
# Create a Sales column
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	101500	Wired Headphones	3	11.99	04/07/24 17:03	9508 Jason Radial Apt. 611, Port Yolandashire,...	4	35.97
1	101501	Lightning Charging Cable	3	14.95	04/13/24 15:28	58919 Mallory Island, Lake Brian, RI 19544	4	44.85
2	101502	Lightning Charging Cable	1	14.95	04/10/24 19:55	8641 Roberts Inlet, North Paul, CA 04296	4	14.95

Next steps: Generate code with `all_data` View recommended plots New interactive sheet

add the city column

```
# Function to extract city from the Purchase Address
def get_city(address):
    return address.split(',')[1].strip()

# Apply the function to create the 'City' column
df['City'] = df['Purchase Address'].apply(get_city)

display(df.head())
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	101500	Wired Headphones	3	11.99	04/07/24 17:03	9508 Jason Radial Apt. 611, Port Yolandashire,...	4	35.97	Port Yolandashire
1	101501	Lightning Charging Cable	3	14.95	04/13/24 15:28	58919 Mallory Island, Lake Brian, RI 19544	4	44.85	Lake Brian
2	101502	Lightning Charging Cable	1	14.95	04/10/24 19:55	8641 Roberts Inlet, North Paul, CA 04296	4	14.95	North Paul
3	101503	Vareebadd Phone	3	400.00	04/18/24 04:07	57769 Powell Springs, Thompsonmouth, VA 63086	4	1200.00	Thompsonmouth
4	101504	AA Batteries (4-pack)	1	3.84	04/20/24 00:39	707 Brooke Estates Apt. 885, South Brianland, ...	4	3.84	South Brianland

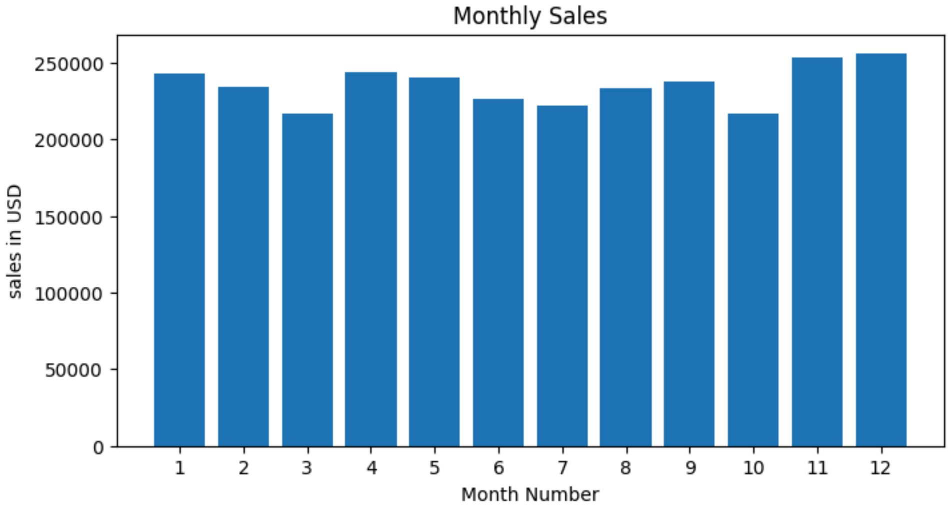
```
# Group by 'Month' and 'City' and sum the 'Sales'
monthly_city_sales = df.groupby(['Month',])['Sales'].sum()

display(monthly_city_sales)
```

	Sales
Month	
1	243052.10
2	234319.99
3	216919.00
4	243819.19
5	240123.43
6	226157.88
7	221537.41
8	233150.40
9	237773.07
10	216851.36
11	253413.62
12	255545.98

dtype: float64

Question 1: what was the best month for Sales? How much was earned that month

```
# Group by 'Month' and 'City' and sum the 'Sales'
monthly_sales = df.groupby(['Month'])['Sales'].sum()

months = monthly_sales.index

sales_values = monthly_sales.values
# Set the figure size
plt.figure(figsize=(8, 4))
plt.bar(months, sales_values)
plt.xticks(months)
plt.xlabel("Month Number")
plt.ylabel("sales in USD")
plt.title("Monthly Sales")
plt.show()
```

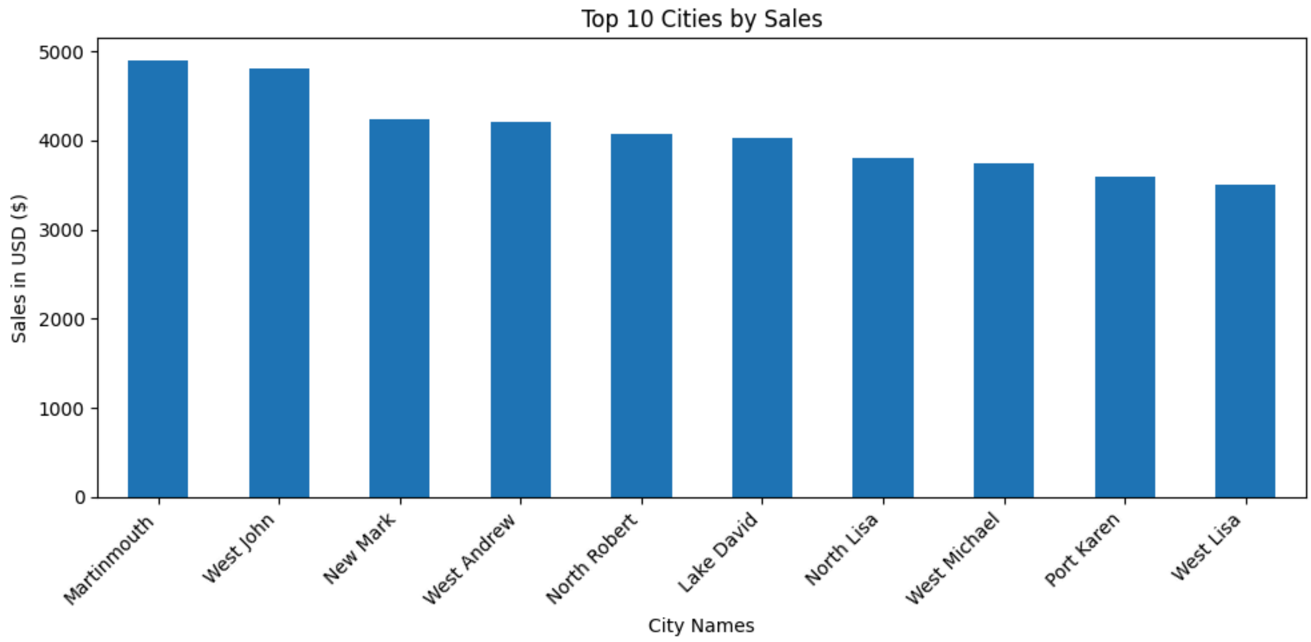


Question 2: what city had the highest number of sales ?

```
city_sales = df.groupby('City')['Sales'].sum()
city_sales
```

```
                       Sales
                City
           Aaronberg    11.95
        Aaronborough   700.00
        Aaronchester   199.98
           Aaronfort  2100.00
          Aaronmouth   800.00
                 ...      ...
            Zhangton   149.99
       Zimmermanton     11.52
      Zimmermanview    400.00
          Zunigaberg     3.84
          Zunigafurt   600.00

5180 rows × 1 columns

dtype: float64
```

```
top_cities = city_sales.sort_values(ascending=False).head(10)

# Create a bar plot of the top 10 cities by sales
plt.figure(figsize=(10, 5))
top_cities.plot(kind='bar')
plt.xlabel("City Names")
plt.ylabel("Sales in USD ($)")
plt.title("Top 10 Cities by Sales")
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for readability
plt.tight_layout() # Adjust layout to prevent labels from overlapping
plt.show()
```



Question 3: what time should we display advertisements to maximiza likelihood of customers buying product ?

```
all_data['Order Date']=pd.to_datetime(all_data['Order Date'])

# Extract the hour and minute from the 'Order Date'
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute

# Group by 'Hour' and 'Minute' and sum the 'Sales'
hourly_minute_sales = all_data.groupby(['Hour', 'Minute'])['Sales'].sum()

# Display the hourly and minute sales
display(hourly_minute_sales)
```

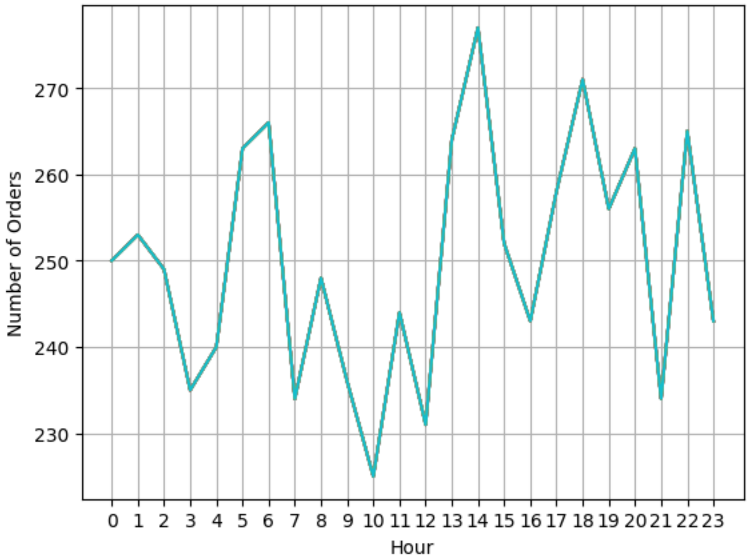
```
/tmp/ipython-input-3956599124.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back t
  all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
                  Sales
Hour  Minute
   0       0     630.31
           1    2049.99
           2    2847.96
           3    2447.34
           4       7.68
 ...     ...        ...
  23      55    2557.88
          56      15.83
          57    1038.90
          58    2267.47
          59    2689.79

1420 rows × 1 columns

dtype: float64
```

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
0	101500	Wired Headphones	3	11.99	2024-04-07 17:03:00	9508 Jason Radial Apt. 611, Port Yolandashire,...	4	35.97	Port Yolandashire	17	3
1	101501	Lightning Charging Cable	3	14.95	2024-04-13 15:28:00	58919 Mallory Island, Lake Brian, RI 19544	4	44.85	Lake Brian	15	28
2	101502	Lightning Charging Cable	1	14.95	2024-04-10 19:55:00	8641 Roberts Inlet, North Paul, CA 04296	4	14.95	North Paul	19	55
3	101503	Vareebadd Phone	3	400.00	2024-04-18 04:07:00	57769 Powell Springs, Thompsonmouth, VA 63086	4	1200.00	Thompsonmouth	4	7
4	101504	AA Batteries (4-	1	3.84	2024-04-20	707 Brooke Estates Apt.	4	3.84	South Brianland	0	39

Next steps: Generate code with all_data View recommended plots New interactive sheet

```
hour = [hour for hour,df in all_data.groupby('Hour')]
plt.plot(hour,all_data.groupby(["Hour"]).count())
plt.xticks(hour)
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.grid()
plt.show()
```



Question 4: What product sold the most?Why do you thinks if sold the most?

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
0	101500	Wired Headphones	3	11.99	2024-04-07 17:03:00	9508 Jason Radial Apt. 611, Port Yolandashire,...	4	35.97	Port Yolandashire	17	3
1	101501	Lightning Charging Cable	3	14.95	2024-04-13 15:28:00	58919 Mallory Island, Lake Brian, RI 19544	4	44.85	Lake Brian	15	28
2	101502	Lightning Charging Cable	1	14.95	2024-04-10 19:55:00	8641 Roberts Inlet, North Paul, CA 04296	4	14.95	North Paul	19	55
3	101503	Vareebadd Phone	3	400.00	2024-04-18 04:07:00	57769 Powell Springs, Thompsonmouth, VA 63086	4	1200.00	Thompsonmouth	4	7
4	101504	AA Batteries (4-	1	3.84	2024-04-20	707 Brooke Estates Apt.	4	3.84	South Brianland	0	39

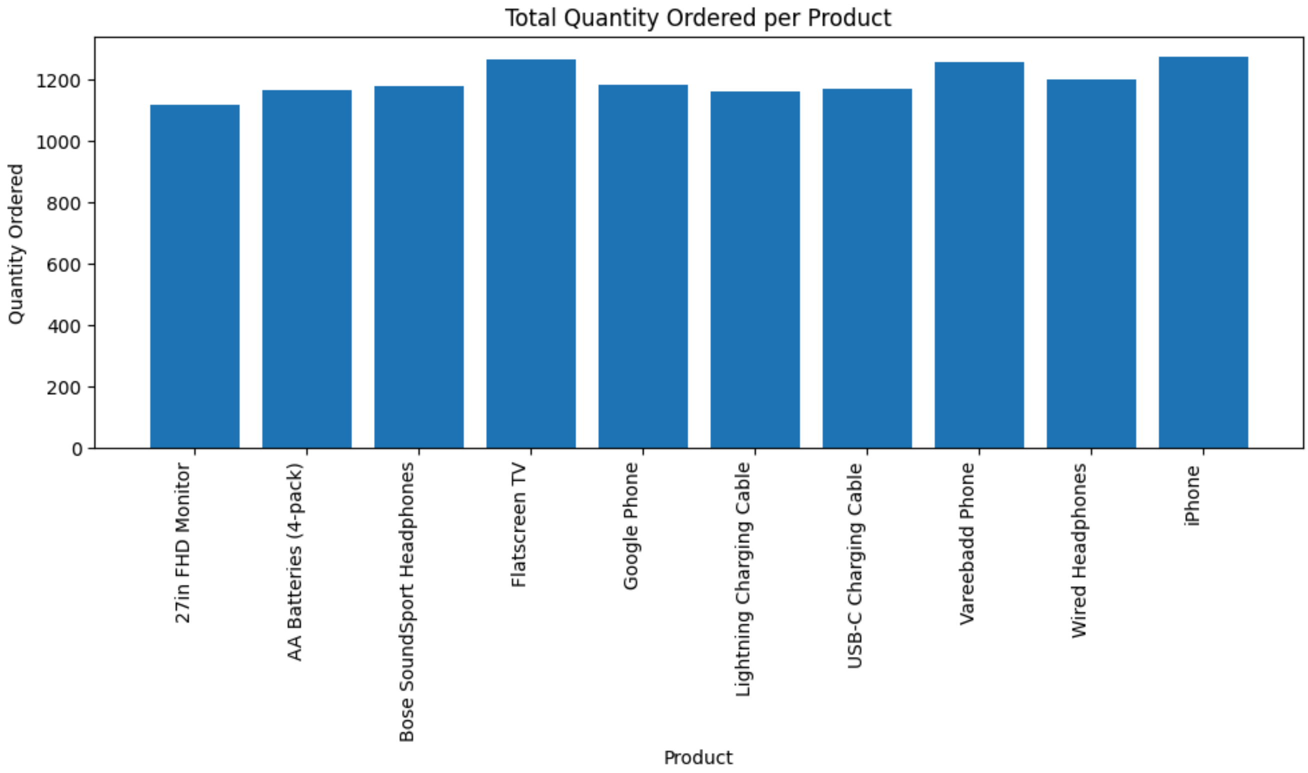
Next steps: Generate code with all_data View recommended plots New interactive sheet

```
product_group = all_data.groupby('Product')
Quantity_ordered = product_group['Quantity Ordered'].sum()
Quantity_ordered
```

	Quantity Ordered
Product	
27in FHD Monitor	1118
AA Batteries (4-pack)	1168
Bose SoundSport Headphones	1182
Flatscreen TV	1269
Google Phone	1183
Lightning Charging Cable	1162
USB-C Charging Cable	1171
Vareebadd Phone	1257
Wired Headphones	1204
iPhone	1276

dtype: int64

```
# Get the product names and the ordered quantities
products = Quantity_ordered.index
quantities = Quantity_ordered.values

# Create the bar plot
plt.figure(figsize=(10, 6))
plt.bar(products, quantities)
plt.xlabel("Product")
plt.ylabel("Quantity Ordered")
plt.title("Total Quantity Ordered per Product")
plt.xticks(rotation=90, ha='right')
plt.tight_layout()
plt.show()
```



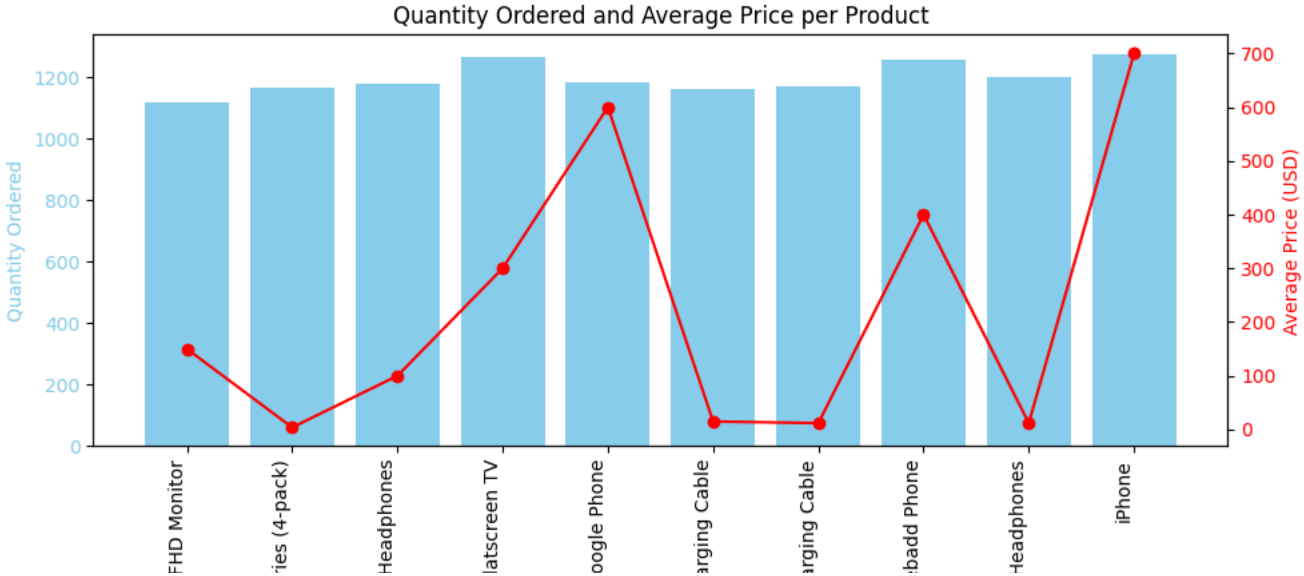
```
Prices = all_data.groupby('Product')['Price Each'].mean()
print(Prices)
```

```
Product
27in FHD Monitor              149.99
AA Batteries (4-pack)           3.84
Bose SoundSport Headphones     99.99
Flatscreen TV                 300.00
Google Phone                  600.00
Lightning Charging Cable       14.95
USB-C Charging Cable           11.95
Vareebadd Phone               400.00
Wired Headphones               11.99
```

```
iPhone                          700.00
Name: Price Each, dtype: float64
```

```
product_quantity = all_data.groupby('Product')['Quantity Ordered'].sum()

product_prices = all_data.groupby('Product')['Price Each'].mean()

products = product_quantity.index

fig, ax1 = plt.subplots(figsize=(10, 6))
ax1.bar(products, product_quantity.values, color='skyblue')
ax1.set_xlabel("Product")
ax1.set_ylabel("Quantity Ordered", color='skyblue')
ax1.tick_params(axis='y', labelcolor='skyblue')
ax2 = ax1.twinx()
ax2.plot(products, product_prices.values, color='red', marker='o')

ax2.set_ylabel("Average Price (USD)", color='red')
ax2.tick_params(axis='y', labelcolor='red')


ax1.set_xticklabels(products, rotation=90, ha='right')
plt.title("Quantity Ordered and Average Price per Product")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2294541481.py:19: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_
  ax1.set_xticklabels(products, rotation=90, ha='right')
```



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.