SalesAnalysis

November 19, 2024

1 Sales Analysis

1.1 PROBLEM STATEMENT :

To perform a detailed analysis of a dataset containing sales information and order quantities, we would typically follow a series of steps in the data analysis process. Below is a structured approach to performing the analysis, which includes data cleaning, exploratory data analysis (EDA), and generating insights based on the data.

1.1.1 Objectives:

Importing necessary Libraries/Modules: - Import the modules necessary for Data Manipulation and Visualization.

Loading dataset: - Read the dataset containing sales information.

Task 1 - Data Cleaning:

Task 2 - Add month column:

Task 3 - Add sales column to the dataframe

Task 4 - Add City column to the dataframe

Task 5 - Add hour column to the dataframe

1.1.2 Questions:

Question 1 - What was the best month for sales? How much was earned that month?

Question 2 - Which city had the highest number of sales?

Question 3 - What time should we display the advertisements to maximize likelihood of customer's buying

Question 4 - What products are sold together

Question 5 - SHow the sales, prices and product in same graph.

1.1.3 CONCLUSION

1.1.4 IMPORTING LIBRARIES/MODULES

[184]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib.ticker import MultipleLocator, FuncFormatter
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

1.1.5 Get the list of all CSV files in the directory

[195]:
```
files=[file for file in os.listdir("./DirtySalesDate")if file.endswith(".csv")]
```

1.1.6 We have 12 CSV files, one for each month of the year 2019. Let's combine them into a single CSV file for easier analysis.

[198]:
```
# Create an empty Dataframe to hold the combined data
all_df=pd.DataFrame()

# Read and concatenate each CSV file
for i in files:
    month_data=pd.read_csv("./DirtySalesDate"+"//"+i)
    all_df=pd.concat([all_df,month_data])
```

1.1.7 Read the updated DataFrame

[9]:
```
all_df.head()
```

[9]:
```
  Order ID               Product Quantity Ordered Price Each      Order Date  \
0   295665    Macbook Pro Laptop                1       1700  12/30/19 00:01
1   295666    LG Washing Machine                1      600.0  12/29/19 07:03
2   295667  USB-C Charging Cable                1      11.95  12/12/19 18:21
3   295668      27in FHD Monitor                1     149.99  12/22/19 15:13
4   295669  USB-C Charging Cable                1      11.95  12/18/19 12:38

                      Purchase Address
0  136 Church St, New York City, NY 10001
1     562 2nd St, New York City, NY 10001
2    277 Main St, New York City, NY 10001
3     410 6th St, San Francisco, CA 94016
4           43 Hill St, Atlanta, GA 30301
```

1.1.8 DataFrame Specifics

```
[11]: all_df.shape
```

```
[11]: (186850, 6)
```

```
[12]: all_df.size
```

```
[12]: 1121100
```

```
[13]: all_df.describe()
```

```
[13]:         Order ID               Product Quantity Ordered Price Each  \
count     186305                186305           186305     186305
unique    178438                    20               10         24
top     Order ID  USB-C Charging Cable                1      11.95
freq         355                 21903           168552      21903

        Order Date  Purchase Address
count       186305            186305
unique      142396            140788
top     Order Date  Purchase Address
freq           355               355
```

```
[14]: all_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 186850 entries, 0 to 13621
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
dtypes: object(6)
memory usage: 10.0+ MB
```

1.1.9 Rows with NAN/ null values

```
[16]: all_df.isnull().sum()
```

```
[16]: Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
```

```
Order Date          545
Purchase Address    545
dtype: int64
```

[17]:

```
null_df = all_df[all_df.isna().any(axis=1)]
null_df.head()
```

[17]:

```
     Order ID Product Quantity Ordered Price Each Order Date Purchase Address
264       NaN     NaN              NaN        NaN        NaN              NaN
648       NaN     NaN              NaN        NaN        NaN              NaN
680       NaN     NaN              NaN        NaN        NaN              NaN
1385      NaN     NaN              NaN        NaN        NaN              NaN
1495      NaN     NaN              NaN        NaN        NaN              NaN
```

[18]:

```
null_df.shape
```

[18]:

```
(545, 6)
```

1.2 Task 1: Data Cleaning

1.2.1 Drop rows with NaN

[21]:

```
all_df = all_df.dropna()
all_df.shape
```

[21]:

```
(186305, 6)
```

1.2.2 Rows with random data

[23]:

```
or_df = all_df[all_df['Order Date'].str[0:2]=='Or']
or_df.head()
```

[23]:

```
      Order ID  Product  Quantity Ordered  Price Each  Order Date  \
254   Order ID  Product  Quantity Ordered  Price Each  Order Date
705   Order ID  Product  Quantity Ordered  Price Each  Order Date
1101  Order ID  Product  Quantity Ordered  Price Each  Order Date
2875  Order ID  Product  Quantity Ordered  Price Each  Order Date
3708  Order ID  Product  Quantity Ordered  Price Each  Order Date

      Purchase Address
254   Purchase Address
705   Purchase Address
1101  Purchase Address
2875  Purchase Address
3708  Purchase Address
```

1.2.3 Drop rows with random data

```
[25]: all_df=all_df[all_df['Order Date'].str[0:2]!='Or']
      all_df.head()
```

```
[25]:   Order ID                   Product Quantity Ordered Price Each        Order Date  \
      0   295665        Macbook Pro Laptop                1       1700    12/30/19 00:01
      1   295666        LG Washing Machine                1      600.0    12/29/19 07:03
      2   295667      USB-C Charging Cable                1      11.95    12/12/19 18:21
      3   295668          27in FHD Monitor                1     149.99    12/22/19 15:13
      4   295669      USB-C Charging Cable                1      11.95    12/18/19 12:38

                              Purchase Address
      0   136 Church St, New York City, NY 10001
      1      562 2nd St, New York City, NY 10001
      2     277 Main St, New York City, NY 10001
      3      410 6th St, San Francisco, CA 94016
      4            43 Hill St, Atlanta, GA 30301
```

1.2.4 Augment data with additional columns

1.3 Task 2: Add month column

```
[28]: all_df['Month']=all_df['Order Date'].str[0:2].astype(int)
      all_df.head()
```

```
[28]:   Order ID                   Product Quantity Ordered Price Each        Order Date  \
      0   295665        Macbook Pro Laptop                1       1700    12/30/19 00:01
      1   295666        LG Washing Machine                1      600.0    12/29/19 07:03
      2   295667      USB-C Charging Cable                1      11.95    12/12/19 18:21
      3   295668          27in FHD Monitor                1     149.99    12/22/19 15:13
      4   295669      USB-C Charging Cable                1      11.95    12/18/19 12:38

                              Purchase Address  Month
      0   136 Church St, New York City, NY 10001     12
      1      562 2nd St, New York City, NY 10001     12
      2     277 Main St, New York City, NY 10001     12
      3      410 6th St, San Francisco, CA 94016     12
      4            43 Hill St, Atlanta, GA 30301     12
```

1.3.1 Add DataFrame to new .csv

```
[30]: all_df.to_csv("All_unfiltered_data.csv")
```

2 Question 1: What was the best month for sales? How much was earned that month?

```
[32]: all_df
```

```
[32]:       Order ID                 Product Quantity Ordered Price Each  \
      0       295665      Macbook Pro Laptop                1       1700
      1       295666      LG Washing Machine                1      600.0
      2       295667    USB-C Charging Cable                1      11.95
      3       295668        27in FHD Monitor                1     149.99
      4       295669    USB-C Charging Cable                1      11.95
      …        …                      …              …        …
      13617   222905  AAA Batteries (4-pack)                1       2.99
      13618   222906        27in FHD Monitor                1     149.99
      13619   222907    USB-C Charging Cable                1      11.95
      13620   222908    USB-C Charging Cable                1      11.95
      13621   222909  AAA Batteries (4-pack)                1       2.99

                 Order Date                        Purchase Address  Month
      0      12/30/19 00:01  136 Church St, New York City, NY 10001     12
      1      12/29/19 07:03     562 2nd St, New York City, NY 10001     12
      2      12/12/19 18:21    277 Main St, New York City, NY 10001     12
      3      12/22/19 15:13     410 6th St, San Francisco, CA 94016     12
      4      12/18/19 12:38           43 Hill St, Atlanta, GA 30301     12
      …               …                                     …    …
      13617  06/07/19 19:02           795 Pine St, Boston, MA 02215      6
      13618  06/01/19 19:29   495 North St, New York City, NY 10001      6
      13619  06/22/19 18:57   319 Ridge St, San Francisco, CA 94016      6
      13620  06/26/19 18:35    916 Main St, San Francisco, CA 94016      6
      13621  06/25/19 14:33          209 11th St, Atlanta, GA 30301      6

      [185950 rows x 7 columns]
```

```
[33]: all_df['Quantity Ordered'] = pd.to_numeric(all_df['Quantity Ordered'])
      all_df['Price Each'] = pd.to_numeric(all_df['Price Each'])
```

2.1 Task 3: Add sales column to the dataframe

```
[35]: all_df['Sales']=all_df['Quantity Ordered']*all_df['Price Each']
      all_df.head()
```

```
[35]:   Order ID               Product  Quantity Ordered  Price Each  \
      0   295665    Macbook Pro Laptop                 1     1700.00
      1   295666    LG Washing Machine                 1      600.00
      2   295667  USB-C Charging Cable                 1       11.95
      3   295668      27in FHD Monitor                 1      149.99
      4   295669  USB-C Charging Cable                 1       11.95
      Order Date                       Purchase Address  Month    Sales
0  12/30/19 00:01  136 Church St, New York City, NY 10001     12  1700.00
1  12/29/19 07:03     562 2nd St, New York City, NY 10001     12   600.00
2  12/12/19 18:21    277 Main St, New York City, NY 10001     12    11.95
3  12/22/19 15:13     410 6th St, San Francisco, CA 94016     12   149.99
4  12/18/19 12:38           43 Hill St, Atlanta, GA 30301     12    11.95
```

[36]:

```
sum_of_sales=all_df.groupby('Month')['Sales'].sum()
sum_of_sales_df = sum_of_sales.reset_index()
sum_of_sales_df.columns = ['Month', 'Total_Sales']
sum_of_sales_df
```

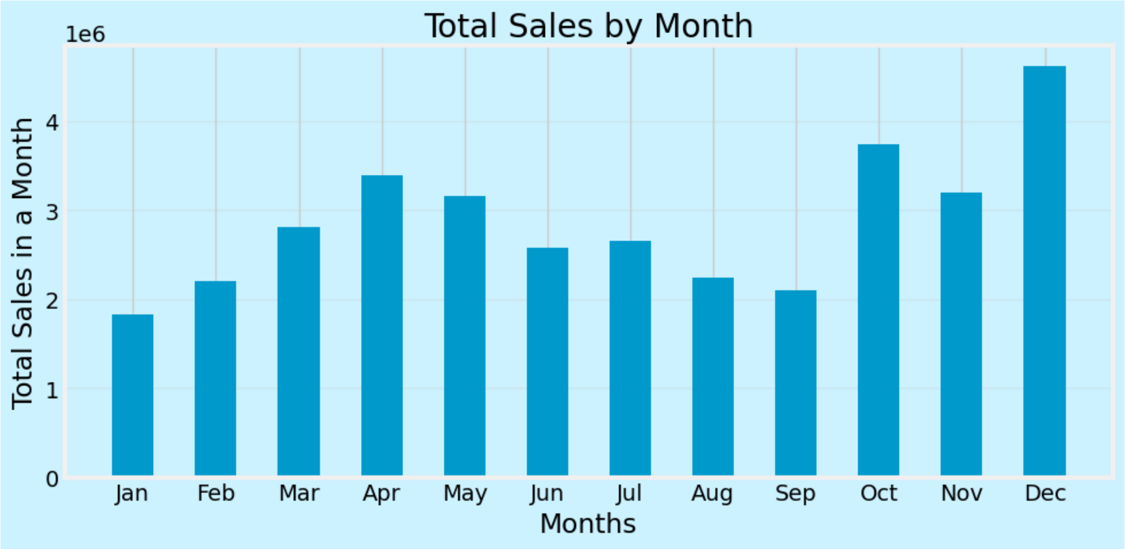
[36]:

```
    Month  Total_Sales
0       1   1822256.73
1       2   2202022.42
2       3   2807100.38
3       4   3390670.24
4       5   3152606.75
5       6   2577802.26
6       7   2647775.76
7       8   2244467.88
8       9   2097560.13
9      10   3736726.88
10     11   3199603.20
11     12   4613443.34
```

[161]:

```
plt.style.use('fivethirtyeight')

# Create the figure and set the background color
fig, ax = plt.subplots(figsize=(10, 5))
fig.patch.set_facecolor("#ccf2ff")  # Set figure background color
ax.set_facecolor("#ccf2ff")         # Set axes background color

# Plot bar chart with a customized bar color
plt.bar([i for i in range(1, 13)], sum_of_sales_df['Total_Sales'],␣
 ↪color="#0099cc",width=0.5)

# Set month names for the x-axis
plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May',␣
 ↪'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

# Label axes and title
plt.xlabel("Months")
plt.ylabel("Total Sales in a Month")
plt.title("Total Sales by Month")
```

```
# Adjust grid visibility
plt.grid(axis='y', alpha=0.3)

# Show plot
plt.tight_layout()
plt.show()
```



The best month for sales is December (Month 12), with a total of $4,613,443.34 in sales.** December (Month 12) has the highest sales at 4,613,443.34 dollars, nearly doubling the sales of January, which had the lowest sales at 1,822,256.73 dollars. This suggests that sales tend to increase toward the end of the year, likely due to seasonal factors such as the holiday shopping season.

2.2 Task 4: Add City column to the dataframe

[39]:
```
all_df['City']=all_df["Purchase Address"].apply(lambda x : x.split(",")[1]+"␣
 ↪"+"("+x.split(",")[2].split(" ")[1]+")")
all_df.head()
```

[39]:
```
  Order ID                Product  Quantity Ordered  Price Each  \
0   295665     Macbook Pro Laptop                 1     1700.00
1   295666     LG Washing Machine                 1      600.00
2   295667   USB-C Charging Cable                 1       11.95
3   295668       27in FHD Monitor                 1      149.99
4   295669   USB-C Charging Cable                 1       11.95

      Order Date                       Purchase Address  Month    Sales  \
0  12/30/19 00:01  136 Church St, New York City, NY 10001     12  1700.00
1  12/29/19 07:03     562 2nd St, New York City, NY 10001     12   600.00
2  12/12/19 18:21    277 Main St, New York City, NY 10001     12    11.95
```

```
3  12/22/19 15:13     410 6th St, San Francisco, CA 94016     12   149.99
4  12/18/19 12:38           43 Hill St, Atlanta, GA 30301     12    11.95

                  City
0   New York City (NY)
1   New York City (NY)
2   New York City (NY)
3   San Francisco (CA)
4         Atlanta (GA)
```

3 Question 2: Which city had the highest number of sales?

[41]:

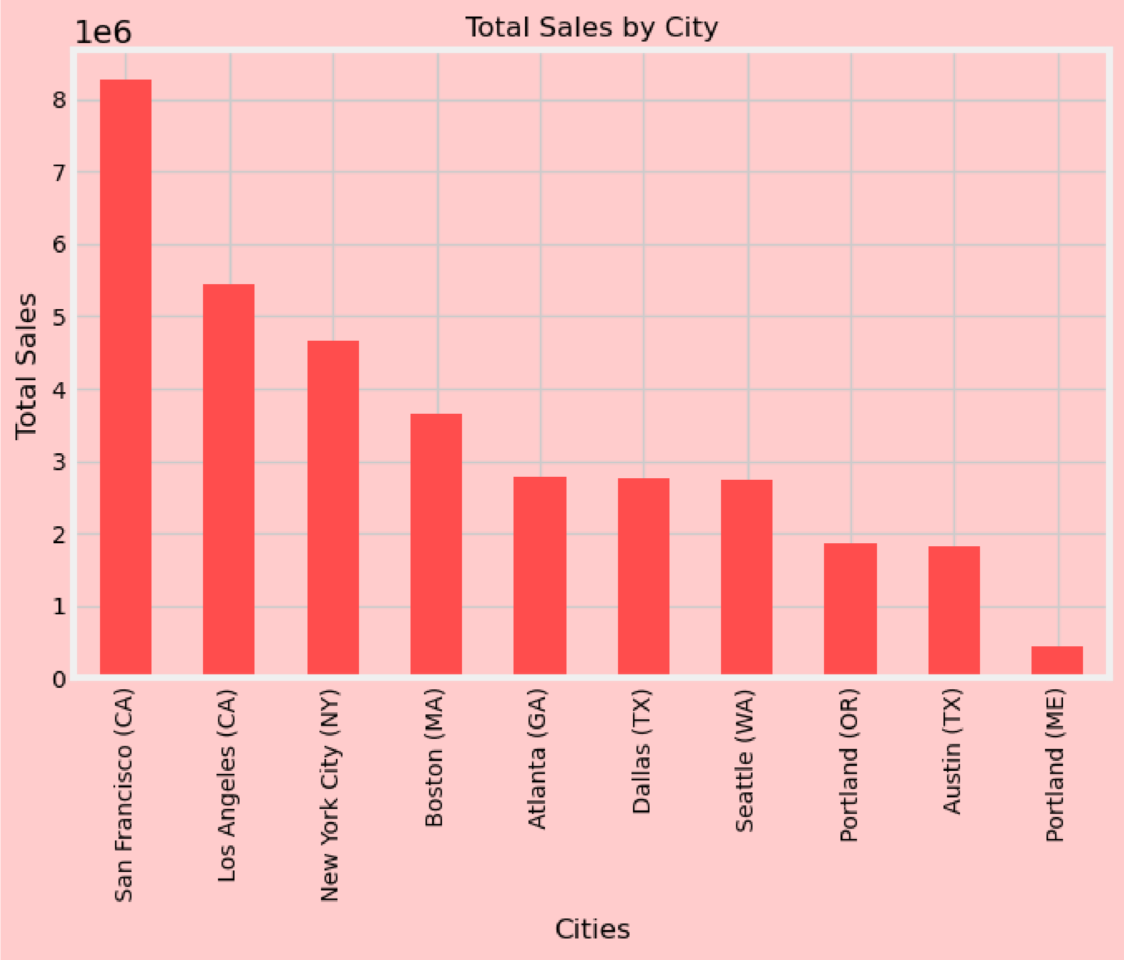
```
for_city=all_df.groupby('City')['Sales'].sum().sort_values(ascending=False)
sum_for_city = for_city.reset_index()
sum_for_city.columns=['City','Total sales']
sum_for_city
```

[41]:

```
                  City  Total sales
0   San Francisco (CA)   8262203.91
1     Los Angeles (CA)   5452570.80
2   New York City (NY)   4664317.43
3          Boston (MA)   3661642.01
4         Atlanta (GA)   2795498.58
5          Dallas (TX)   2767975.40
6         Seattle (WA)   2747755.48
7        Portland (OR)   1870732.34
8          Austin (TX)   1819581.75
9        Portland (ME)    449758.27
```

[140]:

```
plt.style.use("fivethirtyeight")

# Create the figure with a background color
fig, ax = plt.subplots(figsize=(7,6))
fig.patch.set_facecolor("#ffcccc")  # Figure background
ax.set_facecolor('#ffcccc')# Replace '#f0f0f0' with your preferred color

# Plot the data
for_city.plot(kind="bar",color="#ff4d4d")

# Label the axes and title
plt.xlabel('Cities', size=12)
plt.ylabel('Total Sales', size=12)
plt.title('Total Sales by City', size=12)

# Adjust tick size and rotation
plt.yticks(size=10)
plt.xticks(rotation=90, size=10)

# Ensure layout is tight
plt.tight_layout()

# Display the plot
plt.show()
```



San Francisco (CA) leads the cities in total sales revenue, with over 8.26 million dollars, while Portland (ME) shows the lowest sales at just under 450K, highlighting significant regional difference in sales performance.

4 Question 3: what time should we display the advertisements to maximize likelihood of customer's buying

4.1 Task 5: Add hour column to the dataframe

[45]:
```
all_df['Order Date']=pd.to_datetime(all_df['Order Date'])
all_df.head()
```

[45]:
```
  Order ID               Product  Quantity Ordered  Price Each  \
0   295665    Macbook Pro Laptop                 1     1700.00
1   295666    LG Washing Machine                 1      600.00
2   295667  USB-C Charging Cable                 1       11.95
3   295668      27in FHD Monitor                 1      149.99
4   295669  USB-C Charging Cable                 1       11.95

           Order Date                        Purchase Address  Month    Sales  \
0 2019-12-30 00:01:00  136 Church St, New York City, NY 10001     12  1700.00
1 2019-12-29 07:03:00     562 2nd St, New York City, NY 10001     12   600.00
2 2019-12-12 18:21:00    277 Main St, New York City, NY 10001     12    11.95
3 2019-12-22 15:13:00     410 6th St, San Francisco, CA 94016     12   149.99
4 2019-12-18 12:38:00           43 Hill St, Atlanta, GA 30301     12    11.95

                  City
0   New York City (NY)
1   New York City (NY)
2   New York City (NY)
3   San Francisco (CA)
4         Atlanta (GA)
```

[46]:
```
all_df['Hours']=all_df['Order Date'].dt.hour
all_df.head()
```

[46]:
```
  Order ID               Product  Quantity Ordered  Price Each  \
0   295665    Macbook Pro Laptop                 1     1700.00
1   295666    LG Washing Machine                 1      600.00
2   295667  USB-C Charging Cable                 1       11.95
3   295668      27in FHD Monitor                 1      149.99
4   295669  USB-C Charging Cable                 1       11.95

           Order Date                        Purchase Address  Month    Sales  \
0 2019-12-30 00:01:00  136 Church St, New York City, NY 10001     12  1700.00
1 2019-12-29 07:03:00     562 2nd St, New York City, NY 10001     12   600.00
2 2019-12-12 18:21:00    277 Main St, New York City, NY 10001     12    11.95
3 2019-12-22 15:13:00     410 6th St, San Francisco, CA 94016     12   149.99
4 2019-12-18 12:38:00           43 Hill St, Atlanta, GA 30301     12    11.95

                  City  Hours
0   New York City (NY)      0
```

```
1   New York City (NY)      7
2   New York City (NY)     18
3   San Francisco (CA)     15
4         Atlanta (GA)     12
```

[47]:

```
by_hours=all_df.groupby('Hours')['Sales'].sum()
gp_by_hours = by_hours.reset_index()
gp_by_hours.columns = ['Hours','Total sales']
gp_by_hours['Hours'] = gp_by_hours['Hours'] + 1
gp_by_hours
```

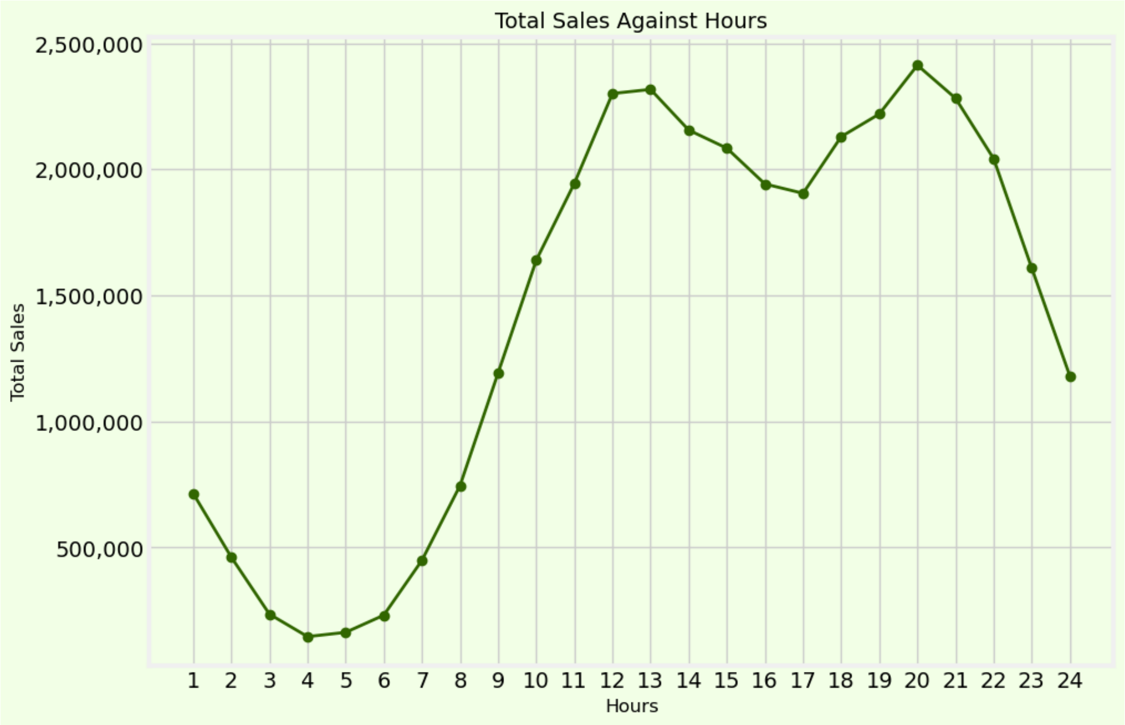
[47]:

```
    Hours  Total sales
0       1    713721.27
1       2    460866.88
2       3    234851.44
3       4    145757.89
4       5    162661.01
5       6    230679.82
6       7    448113.00
7       8    744854.12
8       9   1192348.97
9      10   1639030.58
10     11   1944286.77
11     12   2300610.24
12     13   2316821.34
13     14   2155389.80
14     15   2083672.73
15     16   1941549.60
16     17   1904601.31
17     18   2129361.61
18     19   2219348.30
19     20   2412938.54
20     21   2281716.24
21     22   2042000.86
22     23   1607549.21
23     24   1179304.44
```

[168]:

```
 #Set the style to fivethirtyeight
plt.style.use("fivethirtyeight")

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 7))

# Set the background color
fig.patch.set_facecolor("#f2ffe6")  # Figure background
ax.set_facecolor('#f2ffe6')  # Plot area background
```

```python
# Plotting the data
ax.plot(gp_by_hours['Hours'], gp_by_hours['Total sales'], color="#316600", marker="o", linestyle="-", linewidth=2, markersize=6)

# Set x and y labels
plt.xlabel("Hours", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)

# Set title
plt.title("Total Sales Against Hours", fontsize=14)

# Customize the x-axis ticks
ax.xaxis.set_major_locator(MultipleLocator(1))  # Set ticks every 1 hour
ax.set_xticks(gp_by_hours['Hours'])  # Align ticks with data points

# Format y-axis labels with commas
ax.yaxis.set_major_formatter(FuncFormatter(lambda x, pos: f'{x:,.0f}'))

# Show the graph
plt.show()
```



The graph shows:

Low sales during early morning (2-5 AM) and late night (after 9 PM). Peak sales in the mid-morning (10-11 AM) and evening (5-8 PM). Midday stability with a slight dip post-lunch (2-3

PM). Optimize resources during peak hours and reduce costs during low-activity periods.

[203]:

```
all_df.to_excel("Sales Analysis.xlsx")
```

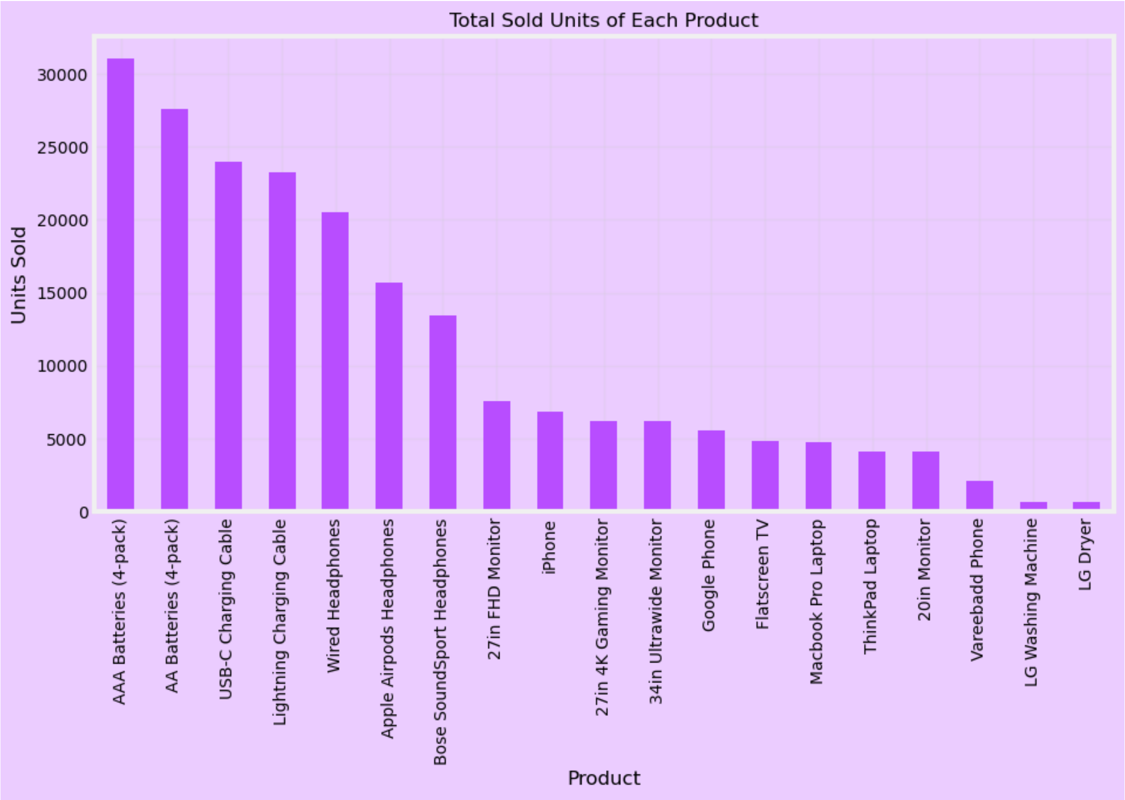
5 Question 4: What products are sold together

[50]:

```
sold_together = all_df.groupby('Product')['Quantity Ordered'].sum().
 ↪sort_values(ascending=False)
```

[176]:

```
#Apply fivethirtyeight style
plt.style.use("fivethirtyeight")

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 5))

# Set the background colors
fig.patch.set_facecolor('#ebccff')  # Set figure background color
ax.set_facecolor('#ebccff')            # Set plot area (axes) background color

# Plot the data as a bar chart
sold_together.plot(kind="bar", ax=ax, color="#b84dff")

# Add title and labels
plt.title("Total Sold Units of Each Product", fontsize=12)
plt.ylabel("Units Sold",size=12)
plt.xlabel("Product",size=12)

plt.xticks(size=10)
plt.yticks(size=10)
# Add a grid with lower alpha for transparency
plt.grid(alpha=0.2)

# Show the plot
plt.show()
```

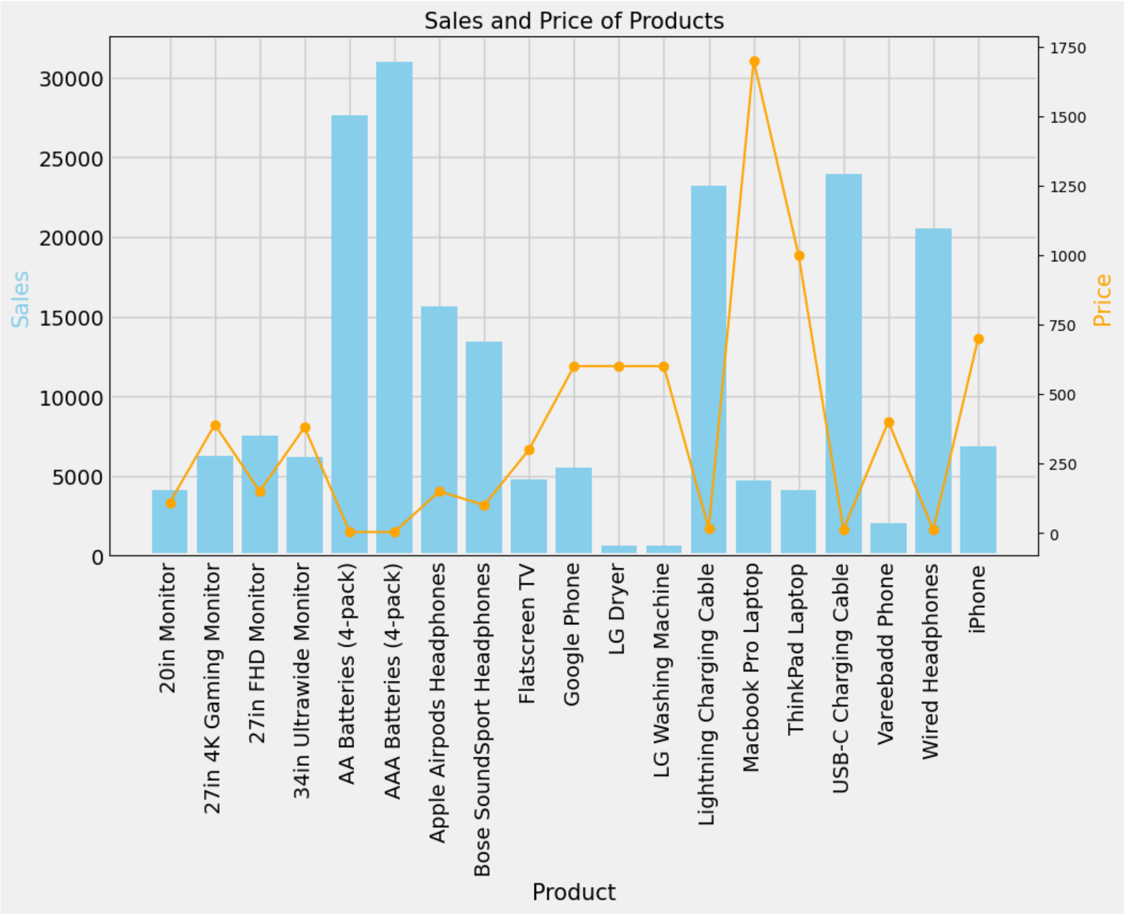


The best-selling product is the AAA Batteries (4-pack), with a total of 31,017 units sold, followed by the AA Batteries (4-pack) with a total of 27,635 units sold. The high sales of AAA batteries can likely be attributed to their widespread use in small electronic devices such as remote controls, flashlights, toys, clocks, and more.**

6 Question 5: SHow the sales, prices and product in same graph.

[54]:
```
price_sale=all_df.groupby(["Product","Price Each"])["Quantity Ordered"].sum()

price_sale_df=pd.DataFrame(price_sale)

price_sale_df.reset_index(inplace=True)

fig, ax1 = plt.subplots(figsize=(10, 6))
plt.style.use("default")
# Bar plot for Sales
ax1.bar(price_sale_df['Product'], price_sale_df['Quantity Ordered'],␣
 ↪color='skyblue', label='Sales')
ax2 = ax1.twinx()
# line plot for Price
```

```
ax2.plot(price_sale_df['Product'], price_sale_df['Price Each'], color='orange',␣
 ↪marker='o', label='Price')


ax1.set_xlabel('Product',fontsize=15)
ax1.set_ylabel('Sales', color='skyblue',fontsize=15)
ax1.tick_params(axis="y",color="skyblue")
ax2.set_ylabel('Price', color='orange',fontsize=15)
# ax1.set_xticklabels(rotation="vertical")
ax1.set_xticklabels(price_sale_df['Product'], rotation=90)

# Title and show plot
plt.title('Sales and Price of Products',fontsize=15)
plt.show()
```



Products like USB-C Charging Cable and AAA Batteries are low in price but show decent sales, making them affordable and accessible.MacBook Pro Laptop has a high price but moderate sales.Bose SoundSport Headphones and Apple AirPods Headphones have a balanced combination of moderate pricing and high sales, suggesting good value and popularity. FlatScreen TV has moderate sales but a noticeable price.

6.1 CONCLUSION

This analysis highlights key trends in sales performance, regional differences, and product popularity. - December was the best month for sales, nearly doubling January's figures, suggesting a strong seasonal boost driven by holiday shopping. - Regionally, San Francisco led with over 8.26 million dollars in sales, while Portland (ME) showed the lowest sales at under 450K, indicating significant regional variations. - In terms of products, the AAA Batteries (4-pack) were the best-sellers, with 31,017 units sold, reflecting strong demand for affordable, everyday items. - Other popular products like the AA Batteries and USB-C Charging Cables also performed well, while higher-ticket items such as the MacBook Pro and Apple AirPods showed solid sales at higher price points.

Overall, this data underscores the importance of seasonal trends, regional markets, and consumer preferences in shaping sales strategies. Products with low price points and high utility, like batteries, drive volume sales, while premium items cater to customers seeking value in quality and performance.

[57]: