# Week11

March 27, 2025

## 1 LAB 21 QUESTION 1

[3]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv('Automobile_data.csv')
```

[4]:
```python
# Lab 21_1a
# Print the first and last five rows
print("First 5 rows:")
print(df.head())
print("\nLast 5 rows:")
print(df.tail())
```

```
First 5 rows:
   index      company   body-style  wheel-base  length engine-type  \
0      0  alfa-romero  convertible        88.6   168.8        dohc
1      1  alfa-romero  convertible        88.6   168.8        dohc
2      2  alfa-romero    hatchback        94.5   171.2        ohcv
3      3         audi        sedan        99.8   176.6         ohc
4      4         audi        sedan        99.4   176.6         ohc

  num-of-cylinders  horsepower  average-mileage    price
0              four         111               21  13495.0
1              four         111               21  16500.0
2               six         154               19  16500.0
3              four         102               24  13950.0
4              five         115               18  17450.0

Last 5 rows:
    index     company body-style  wheel-base  length engine-type  \
56      81  volkswagen      sedan        97.3   171.7         ohc
57      82  volkswagen      sedan        97.3   171.7         ohc
58      86  volkswagen      sedan        97.3   171.7         ohc
59      87       volvo      sedan       104.3   188.8         ohc
60      88       volvo      wagon       104.3   188.8         ohc

   num-of-cylinders  horsepower  average-mileage    price
```

```
56             four          85                27   7975.0
57             four          52                37   7995.0
58             four         100                26   9995.0
59             four         114                23  12940.0
60             four         114                23  13415.0
```

[5]:

```
# Lab 21_1b
# Drop rows with missing values and save the cleaned file
df_cleaned = df.dropna()
df_cleaned.to_csv('Automobile_data_cleaned.csv', index=False)
print("Cleaned data saved as Automobile_data_cleaned.csv")
```

```
Cleaned data saved as Automobile_data_cleaned.csv
```

[6]:

```
# Lab 21_1c
# Find the company with the most expensive car
most_expensive_car = df.loc[df['price'].idxmax()]
print("Company with the most expensive car:", most_expensive_car['company'])
```

```
Company with the most expensive car: mercedes-benz
```

[7]:

```
# Lab 21_1d
# Print all Toyota car details
toyota_cars = df[df['company'].str.lower() == 'toyota']
display(toyota_cars)
```

```
    index company body-style  wheel-base  length engine-type num-of-cylinders  \
48     66  toyota  hatchback        95.7   158.7         ohc             four
49     67  toyota  hatchback        95.7   158.7         ohc             four
50     68  toyota  hatchback        95.7   158.7         ohc             four
51     69  toyota      wagon        95.7   169.7         ohc             four
52     70  toyota      wagon        95.7   169.7         ohc             four
53     71  toyota      wagon        95.7   169.7         ohc             four
54     79  toyota      wagon       104.5   187.8        dohc              six

    horsepower  average-mileage    price
48          62               35   5348.0
49          62               31   6338.0
50          62               31   6488.0
51          62               31   6918.0
52          62               27   7898.0
53          62               27   8778.0
54         156               19  15750.0
```

[8]:

```
# Lab 21_1e
# Count total cars per company
car_counts = df['company'].value_counts()
display(car_counts)
```

```
company
```

```
toyota            7
bmw               6
mazda             5
nissan            5
audi              4
mercedes-benz     4
mitsubishi        4
volkswagen        4
alfa-romero       3
chevrolet         3
honda             3
isuzu             3
jaguar            3
porsche           3
dodge             2
volvo             2
Name: count, dtype: int64
```

[9]:

```python
# Lab 21_1f
# Find each company's highest priced car
highest_priced_cars = df.loc[df.groupby('company')['price'].idxmax()]
display(highest_priced_cars[['company', 'price']])
```

```
          company    price
1     alfa-romero  16500.0
6            audi  18920.0
11            bmw  41315.0
15      chevrolet   6575.0
16          dodge   6377.0
19          honda  12945.0
21          isuzu   6785.0
26         jaguar  36000.0
31          mazda  18344.0
35  mercedes-benz  45400.0
39     mitsubishi   8189.0
44         nissan  13499.0
46        porsche  37028.0
54         toyota  15750.0
58     volkswagen   9995.0
60          volvo  13415.0
```

[10]:

```python
# Lab 21_1g
# Find the average mileage of each company
average_mileage = df.groupby('company')['average-mileage'].mean()
display(average_mileage)
```

```
company
alfa-romero      20.333333
audi             20.000000
```

```
bmw              19.000000
chevrolet        41.000000
dodge            31.000000
honda            26.333333
isuzu            33.333333
jaguar           14.333333
mazda            28.000000
mercedes-benz    18.000000
mitsubishi       29.500000
nissan           31.400000
porsche          17.000000
toyota           28.714286
volkswagen       31.750000
volvo            23.000000
Name: average-mileage, dtype: float64
```

```
[11]: # Lab 21_1h
      # Sort all cars by price
      sorted_cars = df.sort_values(by='price', ascending=True)
      display(sorted_cars[['company', 'price']])
```

```
          company    price
13      chevrolet   5151.0
27          mazda   5195.0
48         toyota   5348.0
36     mitsubishi   5389.0
28          mazda   6095.0
..            …        …
11            bmw  41315.0
35  mercedes-benz  45400.0
22          isuzu      NaN
23          isuzu      NaN
47        porsche      NaN

[61 rows x 2 columns]
```

```
[12]: # Lab 21_1i
      # Create and concatenate German and Japanese car dataframes
      GermanCars = pd.DataFrame({'Company': ['Ford', 'Mercedes', 'BMV', 'Audi'],␣
        ↪'Price': [23845, 171995, 135925 , 71400]})
      japaneseCars = pd.DataFrame({'Company': ['Toyota', 'Honda', 'Nissan',␣
        ↪'Mitsubishi'], 'Price': [29995, 23600, 61500 , 58900]})
      combined_cars = pd.concat([GermanCars, japaneseCars], keys=['German',␣
        ↪'Japanese'])
      display(combined_cars)
```

```
              Company   Price
German    0      Ford   23845
```

```
         1    Mercedes  171995
         2         BMV  135925
         3        Audi   71400
Japanese 0      Toyota   29995
         1       Honda   23600
         2      Nissan   61500
         3  Mitsubishi   58900
```

[13]:

```
# Lab 21_1j
# Merge Car_Price and car_Horsepower dataframes
Car_Price = pd.DataFrame({'Company': ['Toyota', 'Honda', 'BMV', 'Audi'],␣
 ↪'Price': [23845, 17995, 135925 , 71400]})
car_Horsepower = pd.DataFrame({'Company': ['Toyota', 'Honda', 'BMV', 'Audi'],␣
 ↪'horsepower': [141, 80, 182 , 160]})
merged_cars = pd.merge(Car_Price, car_Horsepower, on='Company')
display(merged_cars)
```

```
  Company   Price  horsepower
0  Toyota   23845         141
1   Honda   17995          80
2     BMV  135925         182
3    Audi   71400         160
```

## 2 LAB 21 QUESTION 2

[14]:

```
#Lab21_2
import pandas as pd

# Load the dataset
banklist_df = pd.read_csv('banklist.csv')
```

[15]:

```
#Lab21_2a
# What are the column names?
print("Column names:", banklist_df.columns)
```

```
Column names: Index(['Bank Name', 'City', 'ST', 'CERT', 'Acquiring Institution',
       'Closing Date', 'Updated Date'],
      dtype='object')
```

[16]:

```
#Lab21_2b
# How many States (ST) are represented in this dataset?
num_unique_states = banklist_df['ST'].nunique()
print("Number of unique states:", num_unique_states)
```

```
Number of unique states: 44
```

[17]:

```
#Lab21_2c
# Get an array of all the states in the dataset.
```

```
unique_states = banklist_df['ST'].unique()
print("Array of all states:", unique_states)
```

```
Array of all states: ['AR' 'GA' 'PA' 'TN' 'WI' 'WA' 'CO' 'IL' 'PR' 'FL' 'MN'
'CA' 'MD' 'OK'
 'OH' 'SC' 'VA' 'ID' 'TX' 'CT' 'AZ' 'NV' 'NC' 'KY' 'MO' 'KS' 'AL' 'NJ'
 'MI' 'IN' 'LA' 'IA' 'UT' 'NE' 'MS' 'NM' 'OR' 'NY' 'MA' 'SD' 'WY' 'WV'
 'NH' 'HI']
```

[18]:

```
#Lab21_2d
# What are the top 5 states with the most failed banks?
top_5_states_failed_banks = banklist_df['ST'].value_counts().head(5)
print("Top 5 states with most failed banks:\n", top_5_states_failed_banks)
```

```
Top 5 states with most failed banks:
 ST
GA    93
FL    75
IL    66
CA    41
MN    23
Name: count, dtype: int64
```

[19]:

```
#Lab21_2e
# What are the top 5 acquiring institutions?
top_5_acquiring_institutions = banklist_df['Acquiring Institution'].
  ↪value_counts().head(5)
print("Top 5 acquiring institutions:\n", top_5_acquiring_institutions)
```

```
Top 5 acquiring institutions:
 Acquiring Institution
No Acquirer                           31
State Bank and Trust Company          12
Ameris Bank                           10
First-Citizens Bank & Trust Company    9
U.S. Bank N.A.                         9
Name: count, dtype: int64
```

[20]:

```
#Lab21_2f
# How many banks has the State Bank of Texas acquired? How many of them were in␣
  ↪Texas?
state_bank_of_texas_acquired = banklist_df[banklist_df['Acquiring Institution']␣
  ↪== 'State Bank of Texas']
num_acquired_by_state_bank_of_texas = state_bank_of_texas_acquired.shape[0]
num_in_texas = state_bank_of_texas_acquired[state_bank_of_texas_acquired['ST']␣
  ↪== 'TX'].shape[0]
print("Number of banks acquired by State Bank of Texas:",␣
  ↪num_acquired_by_state_bank_of_texas)
```

```
print("Number of these banks in Texas:", num_in_texas)
```

```
Number of banks acquired by State Bank of Texas: 2
Number of these banks in Texas: 1
```

[21]:

```
#Lab21_2g
# What is the most common city in California for a bank to fail in?
california_failed_banks = banklist_df[banklist_df['ST'] == 'CA']
most_common_city_in_ca = california_failed_banks['City'].value_counts().idxmax()
print("Most common city for a bank to fail in California:",␣
  ↪most_common_city_in_ca)
```

```
Most common city for a bank to fail in California: Los Angeles
```

[22]:

```
#Lab21_2h
# How many failed banks don't have the word 'Bank' in their name?
banks_without_bank_word = banklist_df[~banklist_df['Bank Name'].str.
  ↪contains('Bank', case=False, na=False)]
num_banks_without_bank_word = banks_without_bank_word.shape[0]
print("Number of failed banks without 'Bank' in their name:",␣
  ↪num_banks_without_bank_word)
```

```
Number of failed banks without 'Bank' in their name: 10
```

[23]:

```
#Lab21_2i
# How many bank names start with the letter 's'?
banks_starting_with_s = banklist_df[banklist_df['Bank Name'].str.lower().str.
  ↪startswith('s', na=False)]
num_banks_starting_with_s = banks_starting_with_s.shape[0]
print("Number of bank names starting with 'S':", num_banks_starting_with_s)
```

```
Number of bank names starting with 'S': 53
```

[24]:

```
#Lab21_2j
# How many CERT values are above 20000?
cert_above_20000 = banklist_df[banklist_df['CERT'] > 20000].shape[0]
print("Number of CERT values above 20000:", cert_above_20000)
```

```
Number of CERT values above 20000: 415
```

[25]:

```
#Lab21_2k
# How many bank names consist of just two words?
banks_two_words = banklist_df['Bank Name'].apply(lambda x: len(str(x).split())␣
  ↪== 2)
num_banks_two_words = banks_two_words.sum()
print("Number of bank names with just two words:", num_banks_two_words)
```

```
Number of bank names with just two words: 113
```

# 3 LAB 21 QUESTION 3

[26]:

```
#Lab21_3 Genai
import pandas as pd

# Load the dataset
file_path = "Gen_AI_sales_dataset.csv"
df = pd.read_csv(file_path)

# a. Data Selection and Filtering

# Filter for "Electronics" category
electronics_df = df[df["product_category"] == "Electronics"].copy()

# Filter rows where total_sales > 1000
high_sales_df = electronics_df[electronics_df["total_sales"] > 1000].copy()

# b. Adding New Columns

# Add revenue_per_unit column
high_sales_df.loc[:, "revenue_per_unit"] = high_sales_df["total_sales"] /␣
 ↪high_sales_df["units_sold"]

# Ensure date column is properly converted to datetime format
high_sales_df["date"] = pd.to_datetime(high_sales_df["date"], errors='coerce')

# Drop rows where date conversion failed
high_sales_df = high_sales_df.dropna(subset=["date"])

# Confirm date is in datetime format before extracting month
if not pd.api.types.is_datetime64_any_dtype(high_sales_df["date"]):
    print("Error: Date column is not in datetime format after conversion")
else:
    high_sales_df["month"] = high_sales_df["date"].dt.month


# c. Handling Missing Data

# Replace missing unit_price values with the average unit price
avg_unit_price = high_sales_df["unit_price"].mean()
high_sales_df.loc[:, "unit_price"] = high_sales_df["unit_price"].
 ↪fillna(avg_unit_price)

# Drop rows where units_sold is zero or less
high_sales_df = high_sales_df[high_sales_df["units_sold"] > 0].copy()

# d. Sorting and Ranking
```

```python
# Sort by total_sales in descending order and select top 10 rows
top_10_sales = high_sales_df.sort_values(by="total_sales", ascending=False).
 ↪head(10)

# Rank products by total_sales within each store
high_sales_df.loc[:, "rank_in_store"] = high_sales_df.
 ↪groupby("store_id")["total_sales"].rank(method="dense", ascending=False)

# e. Aggregation

# Group by store_id and calculate required aggregations
store_aggregates = high_sales_df.groupby("store_id").agg(
    total_units_sold=("units_sold", "sum"),
    avg_unit_price=("unit_price", "mean"),
    total_sales=("total_sales", "sum")
).reset_index()

# Display results
print(high_sales_df.dtypes)  # Debugging step to check column types
print(top_10_sales)
print(store_aggregates.head())
```

```
store_id                     int64
date                datetime64[ns]
product_id                   int64
product_category            object
units_sold                   int64
unit_price                 float64
total_sales                float64
revenue_per_unit           float64
month                        int32
rank_in_store              float64
dtype: object
       store_id       date  product_id product_category  units_sold  \
28848        30 2025-12-02         120      Electronics         200
63505         1 2025-05-23         104      Electronics         200
23133        15 2025-06-20         118      Electronics         198
94231        42 2025-03-08         110      Electronics         199
15874        21 2025-04-22         112      Electronics         199
20133        46 2025-02-24         133      Electronics         198
10795        48 2025-01-04         143      Electronics         198
75714        44 2025-06-27         137      Electronics         198
75874        19 2025-03-14         150      Electronics         198
29070        42 2025-11-17         109      Electronics         199

       unit_price  total_sales  revenue_per_unit  month
```

```
28848      649.30    129860.00           649.30     12
63505      642.32    128464.00           642.32      5
23133      648.21    128345.58           648.21      6
94231      644.37    128229.63           644.37      3
15874      642.84    127925.16           642.84      4
20133      644.79    127668.42           644.79      2
10795      644.56    127622.88           644.56      1
75714      644.38    127587.24           644.38      6
75874      644.04    127519.92           644.04      3
29070      638.06    126973.94           638.06     11
   store_id  total_units_sold  avg_unit_price  total_sales
0         1             65835      345.864057  22355094.18
1         2             70451      339.111400  23842039.78
2         3             68064      339.568563  23096447.22
3         4             66811      344.172762  22982318.64
4         5             71152      329.926730  22748954.67
```

[27]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (2.2.4)
Requirement already satisfied: packaging>=20.0 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
```

```
(from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages
(from python-dateutil>=2.7->matplotlib) (1.17.0)

[notice] A new release of pip is
available: 24.3.1 -> 25.0.1
[notice] To update, run:
pip3 install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

# 4 LAB 22 QUESTION 1

[28]:

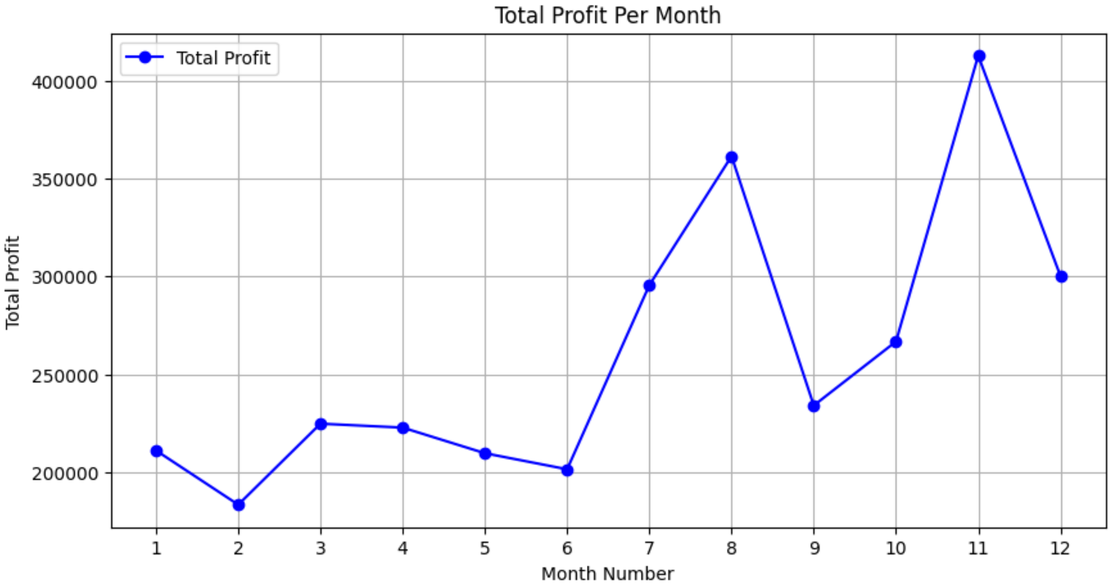
```
#Lab22_1
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'company_sales_data.csv'
df = pd.read_csv(file_path)
```

[29]:

```
#Lab22_1a
# Display basic information about the dataset
print(df.info())
print(df.head())  # Show first few rows to understand structure

# Assuming 'total_profit' is the column representing profit and 'month_number'␣
  ↪represents months
plt.figure(figsize=(10,5))
plt.plot(df['month_number'], df['total_profit'], marker='o', linestyle='-',␣
  ↪color='b', label='Total Profit')

# Labeling
plt.xlabel('Month Number')
plt.ylabel('Total Profit')
plt.title('Total Profit Per Month')
plt.xticks(df['month_number'])  # Ensure month numbers are displayed correctly
plt.legend()
plt.grid(True)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------        --------------  -----
 0   month_number  12 non-null     int64
 1   facecream     12 non-null     int64
 2   facewash      12 non-null     int64
 3   toothpaste    12 non-null     int64
 4   bathingsoap   12 non-null     int64
 5   shampoo       12 non-null     int64
 6   moisturizer   12 non-null     int64
 7   total_units   12 non-null     int64
 8   total_profit  12 non-null     int64
dtypes: int64(9)
memory usage: 996.0 bytes
None
   month_number  facecream  facewash  toothpaste  bathingsoap  shampoo  \
0             1       2500      1500        5200         9200     1200
1             2       2630      1200        5100         6100     2100
2             3       2140      1340        4550         9550     3550
3             4       3400      1130        5870         8870     1870
4             5       3600      1740        4560         7760     1560

   moisturizer  total_units  total_profit
0         1500        21100        211000
1         1200        18330        183300
2         1340        22470        224700
3         1130        22270        222700
4         1740        20960        209600
```

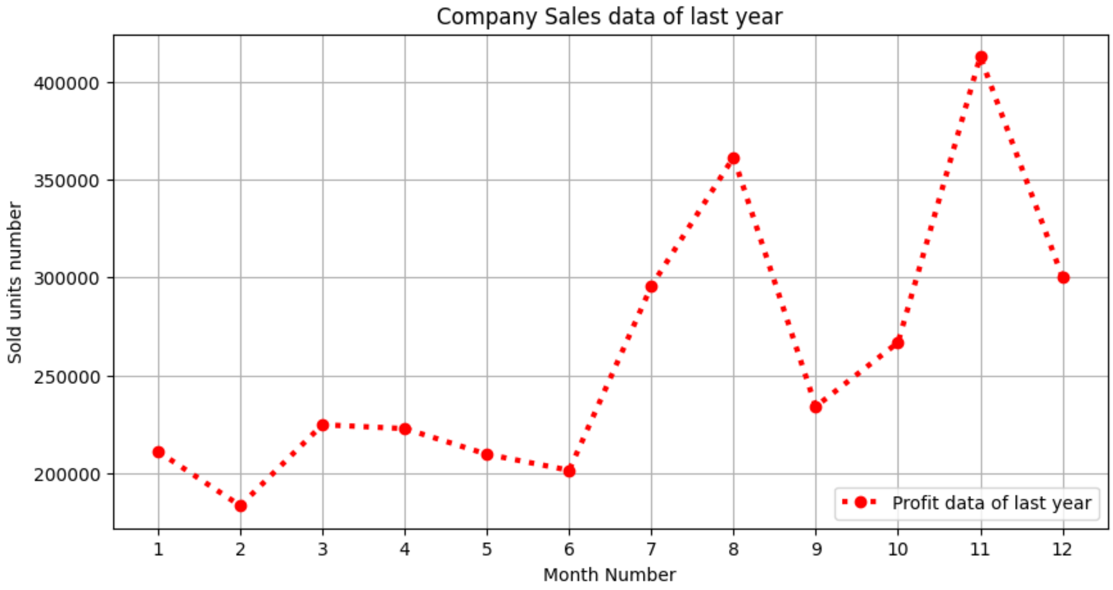


[30]:

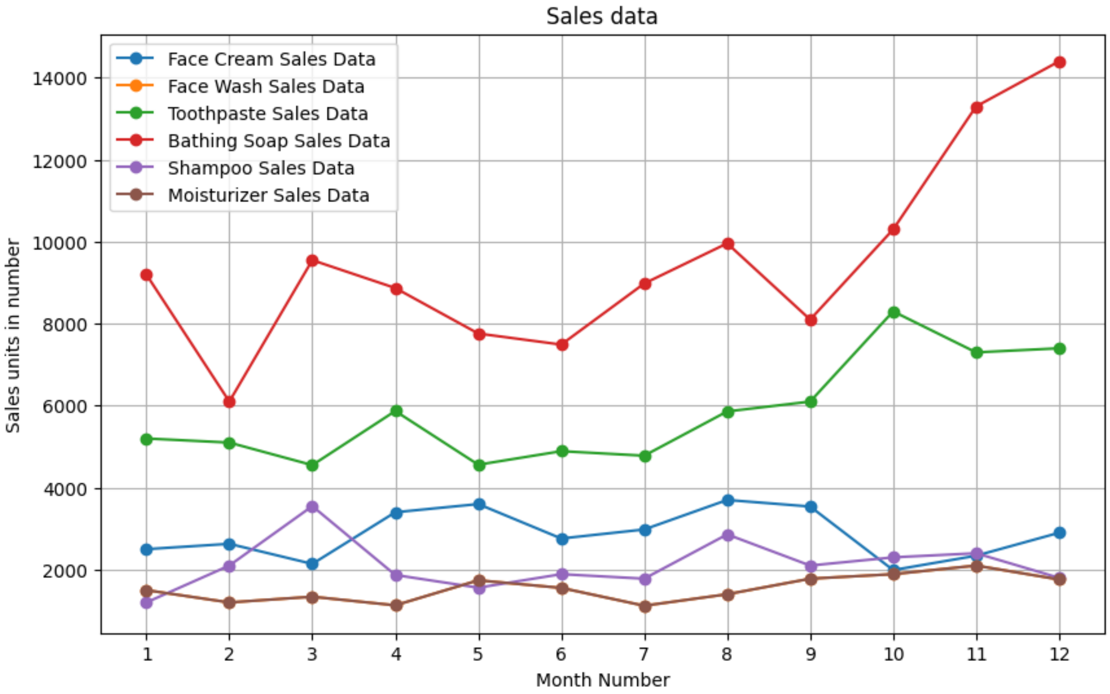
```python
#Lab22_1b
# Display basic information about the dataset
print(df.info())
print(df.head())  # Show first few rows to understand structure

# Assuming 'total_profit' is the column representing profit and 'month_number'␣
 ↪represents months
plt.figure(figsize=(10,5))
plt.plot(df['month_number'], df['total_profit'], marker='o', linestyle=':',␣
 ↪color='r', linewidth=3, label='Profit data of last year')

# Labeling
plt.xlabel('Month Number')
plt.ylabel('Sold units number')
plt.title('Company Sales data of last year')
plt.xticks(df['month_number'])  # Ensure month numbers are displayed correctly
plt.legend(loc='lower right')
plt.grid(True)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   month_number  12 non-null     int64
 1   facecream     12 non-null     int64
 2   facewash      12 non-null     int64
 3   toothpaste    12 non-null     int64
 4   bathingsoap   12 non-null     int64
 5   shampoo       12 non-null     int64
 6   moisturizer   12 non-null     int64
 7   total_units   12 non-null     int64
 8   total_profit  12 non-null     int64
dtypes: int64(9)
memory usage: 996.0 bytes
None
   month_number  facecream  facewash  toothpaste  bathingsoap  shampoo  \
0             1       2500      1500        5200         9200     1200
1             2       2630      1200        5100         6100     2100
2             3       2140      1340        4550         9550     3550
3             4       3400      1130        5870         8870     1870
4             5       3600      1740        4560         7760     1560

   moisturizer  total_units  total_profit
0         1500        21100        211000
```

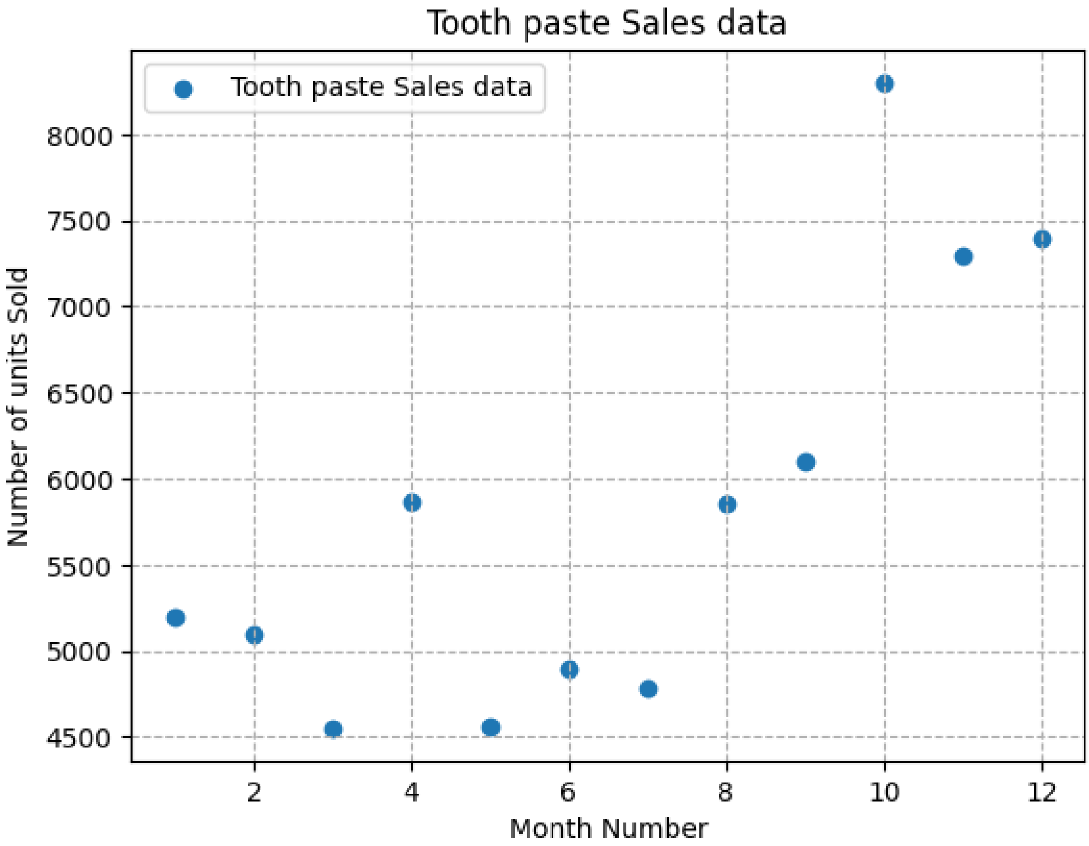
```
1          1200        18330        183300
2          1340        22470        224700
3          1130        22270        222700
4          1740        20960        209600
```



```
[31]: #Lab22_1c
      # Multiline plot for units sold per product
      plt.figure(figsize=(10,6))
      plt.plot(df['month_number'], df['facecream'], marker='o', linestyle='-',␣
        ↪label='Face Cream Sales Data')
      plt.plot(df['month_number'], df['facewash'], marker='o', linestyle='-',␣
        ↪label='Face Wash Sales Data')
      plt.plot(df['month_number'], df['toothpaste'], marker='o', linestyle='-',␣
        ↪label='Toothpaste Sales Data')
      plt.plot(df['month_number'], df['bathingsoap'], marker='o', linestyle='-',␣
        ↪label='Bathing Soap Sales Data')
      plt.plot(df['month_number'], df['shampoo'], marker='o', linestyle='-',␣
        ↪label='Shampoo Sales Data')
      plt.plot(df['month_number'], df['moisturizer'], marker='o', linestyle='-',␣
        ↪label='Moisturizer Sales Data')

      # Labeling
      plt.xlabel('Month Number')
      plt.ylabel('Sales units in number')
      plt.title('Sales data')
      plt.xticks(df['month_number'])
      plt.legend()
      plt.grid(True)

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

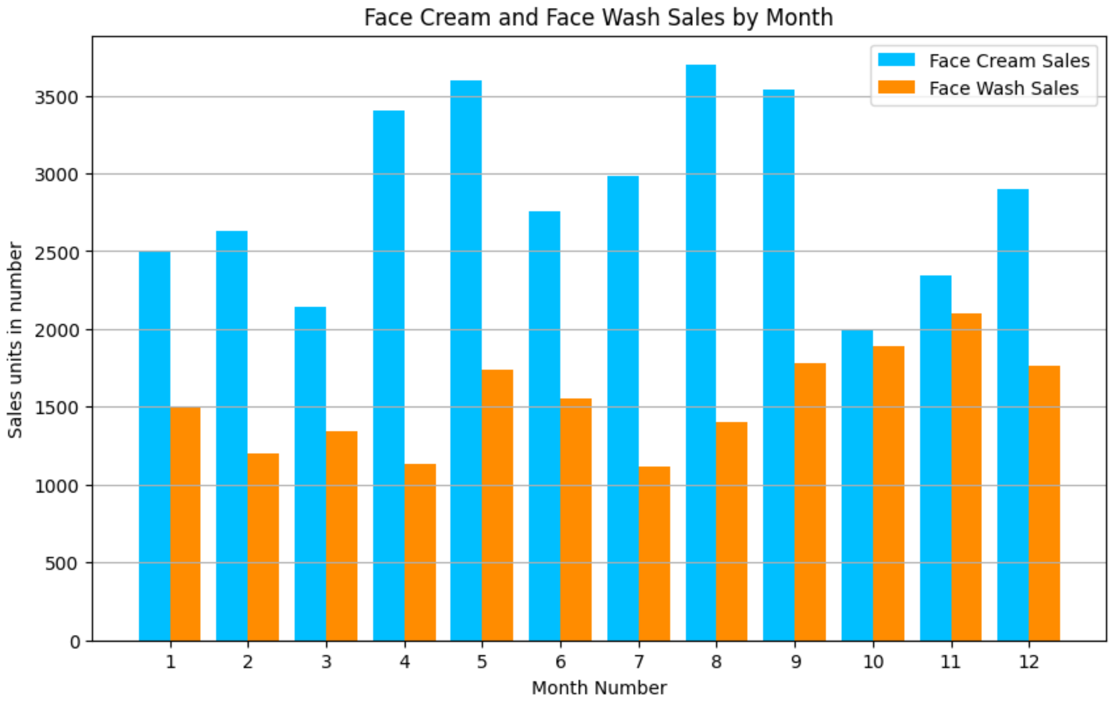


[32]:
```
#Lab22_1d
# Load the data
data = {
    'month_number': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'toothpaste': [5200, 5100, 4550, 5870, 4560, 4890, 4780, 5860, 6100, 8300,␣
  ↪7300, 7400]
}

# Create a scatter plot
plt.scatter(data['month_number'], data['toothpaste'], label='Tooth paste Sales␣
  ↪data')
plt.title('Tooth paste Sales data')
plt.xlabel('Month Number')
plt.ylabel('Number of units Sold')
plt.grid(True, linestyle='--') # Dashed gridlines
plt.legend()
plt.show()
```



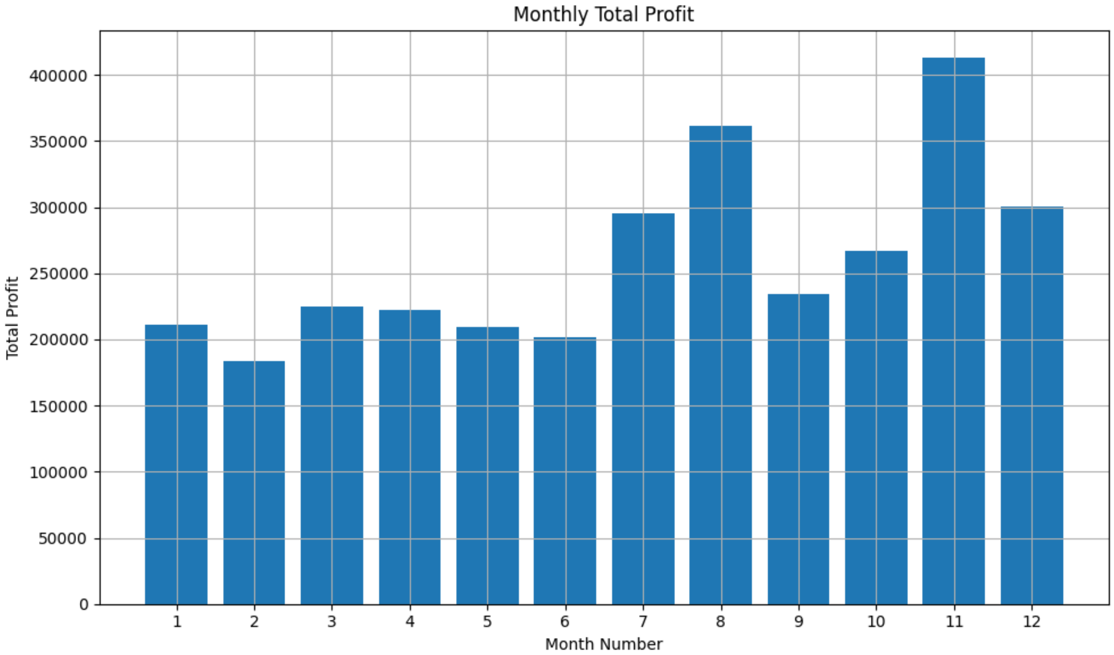
[33]:

```
#Lab22_1e
# Bar plot for Face Cream and Face Wash Sales
plt.figure(figsize=(10,6))
bar_width = 0.4
plt.bar(df['month_number'] - bar_width/2, df['facecream'], width=bar_width,␣
 ↪label='Face Cream Sales', color='deepskyblue')
plt.bar(df['month_number'] + bar_width/2, df['facewash'], width=bar_width,␣
 ↪label='Face Wash Sales', color='darkorange')

# Labeling
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.title('Face Cream and Face Wash Sales by Month')
plt.xticks(df['month_number'])
plt.legend()
plt.grid(axis='y')

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



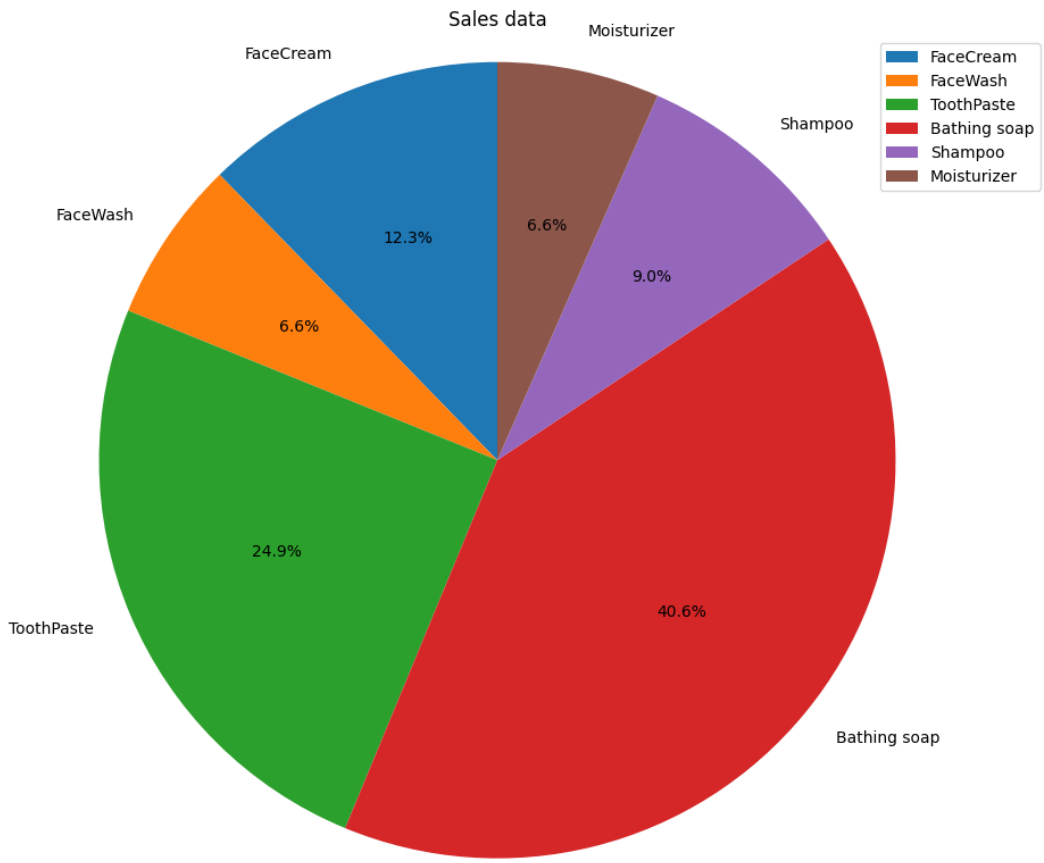
[34]:

```
#Lab22_1f
plt.figure(figsize=(10, 6))
plt.bar(df['month_number'], df['total_profit'])
plt.xlabel('Month Number')
plt.ylabel('Total Profit')
plt.title('Monthly Total Profit')
plt.xticks(df['month_number'])
plt.grid(True)
plt.tight_layout()
plt.show()
plt.close()
```



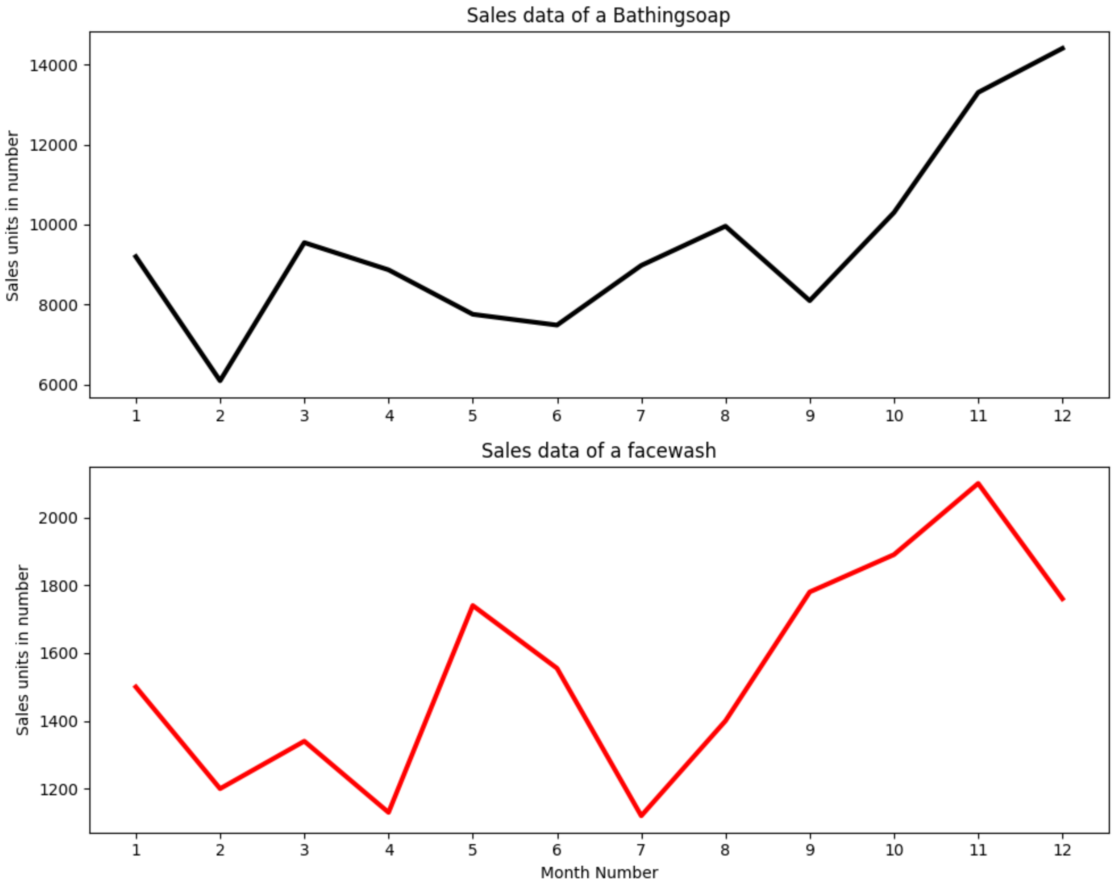
[35]:

```
#Lab22_1g
# Calculate total annual sales for each product
facecream_sales = df['facecream'].sum()
facewash_sales = df['facewash'].sum()
toothpaste_sales = df['toothpaste'].sum()
bathingsoap_sales = df['bathingsoap'].sum()
shampoo_sales = df['shampoo'].sum()
moisturizer_sales = df['moisturizer'].sum()
# Combine the sales data and labels
sales = [facecream_sales, facewash_sales, toothpaste_sales, bathingsoap_sales,␣
 ↪shampoo_sales, moisturizer_sales]
products = ['FaceCream', 'FaceWash', 'ToothPaste', 'Bathing soap', 'Shampoo',␣
 ↪'Moisturizer']
# Calculate percentages
total = sum(sales)
percentages = [round((x/total)*100, 1) for x in sales]
plt.figure(figsize=(10, 8))
plt.pie(sales, labels=products, autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Sales data')
plt.legend(products)
plt.tight_layout()
plt.show()
plt.close()
```



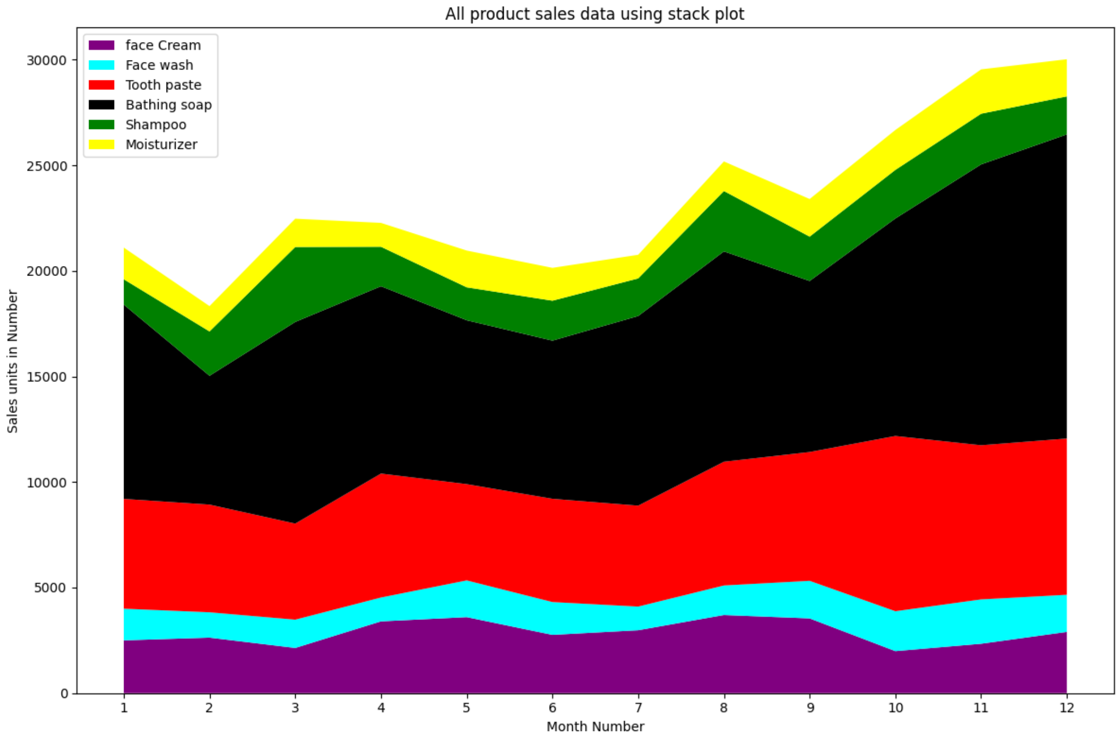
[36]:

```
#Lab22_1h
plt.figure(figsize=(10, 8))

# First subplot - Bathing soap
plt.subplot(2, 1, 1)
plt.plot(df['month_number'], df['bathingsoap'], color='black', linewidth=3)
plt.title('Sales data of a Bathingsoap')
plt.ylabel('Sales units in number')
plt.xticks(df['month_number'])
plt.grid(False)

# Second subplot - Facewash
plt.subplot(2, 1, 2)
plt.plot(df['month_number'], df['facewash'], color='red', linewidth=3)
plt.title('Sales data of a facewash')
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.xticks(df['month_number'])
plt.grid(False)
plt.tight_layout()
plt.show()
plt.close()
```


Sales data of a Bathingsoap

Sales data of a facewash

[37]:
```
#Lab22_1i
plt.figure(figsize=(12, 8))

months = df['month_number']
facecream = df['facecream']
facewash = df['facewash']
toothpaste = df['toothpaste']
bathingsoap = df['bathingsoap']
shampoo = df['shampoo']
moisturizer = df['moisturizer']

plt.stackplot(months,
              facecream,
              facewash,
              toothpaste,
              bathingsoap,
              shampoo,
              moisturizer,
              colors=['purple', 'cyan', 'red', 'black', 'green', 'yellow'])

plt.xlabel('Month Number')
plt.ylabel('Sales units in Number')
plt.title('All product sales data using stack plot')
plt.legend(['face Cream', 'Face wash', 'Tooth paste', 'Bathing soap',␣
 ↪'Shampoo', 'Moisturizer'], loc='upper left')
plt.xticks(df['month_number'])
plt.tight_layout()
plt.show()
plt.close()
```



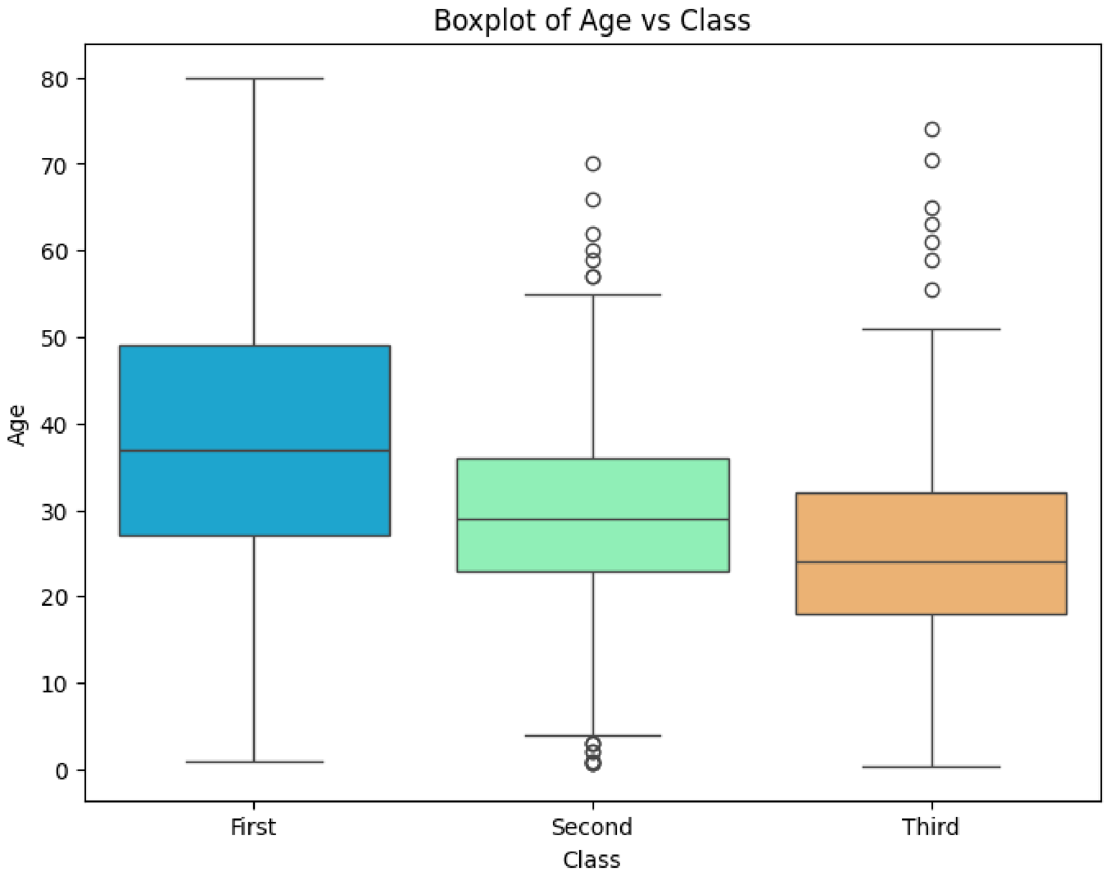
## 5 LAB 22 QUESTION 2

[38]:

```
#Lab22_2
import ssl
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

ssl._create_default_https_context = ssl._create_unverified_context

titanic = sns.load_dataset('titanic')
```

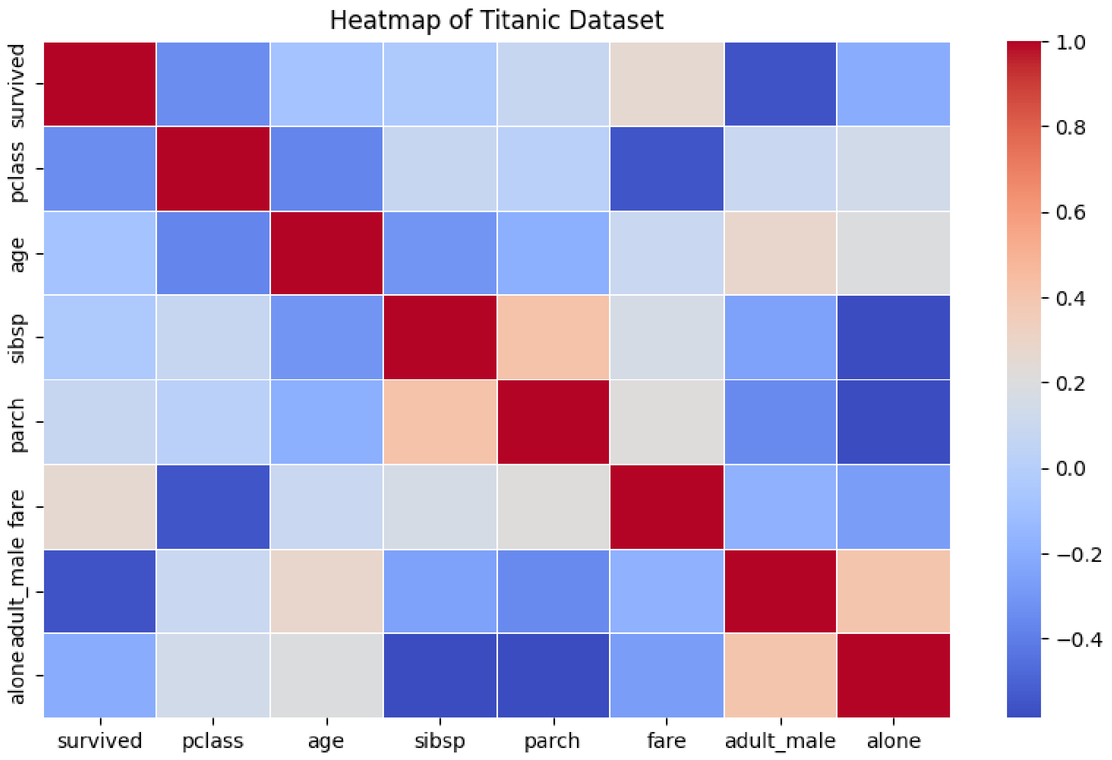
[39]:
```
#Lab22_2a
# Create a boxplot of Age vs Class using the rainbow palette
plt.figure(figsize=(8, 6))
sns.boxplot(x='class', y='age', data=titanic, palette='rainbow')
plt.title('Boxplot of Age vs Class')
plt.xlabel('Class')
plt.ylabel('Age')
plt.show()
```

```
/var/folders/9m/f5jl8xnd4ls2_3ykntk1_rpm0000gn/T/ipykernel_10622/3215136423.py:4
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='class', y='age', data=titanic, palette='rainbow')
```

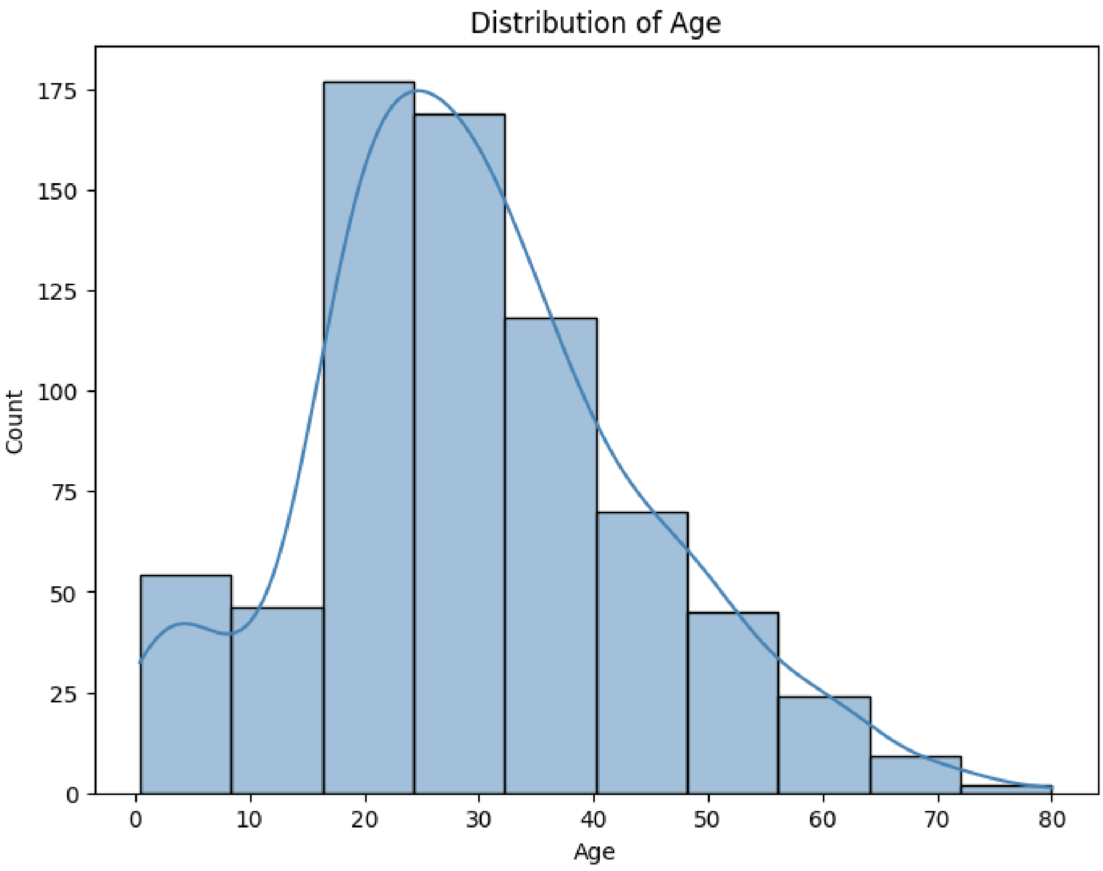


[40]:

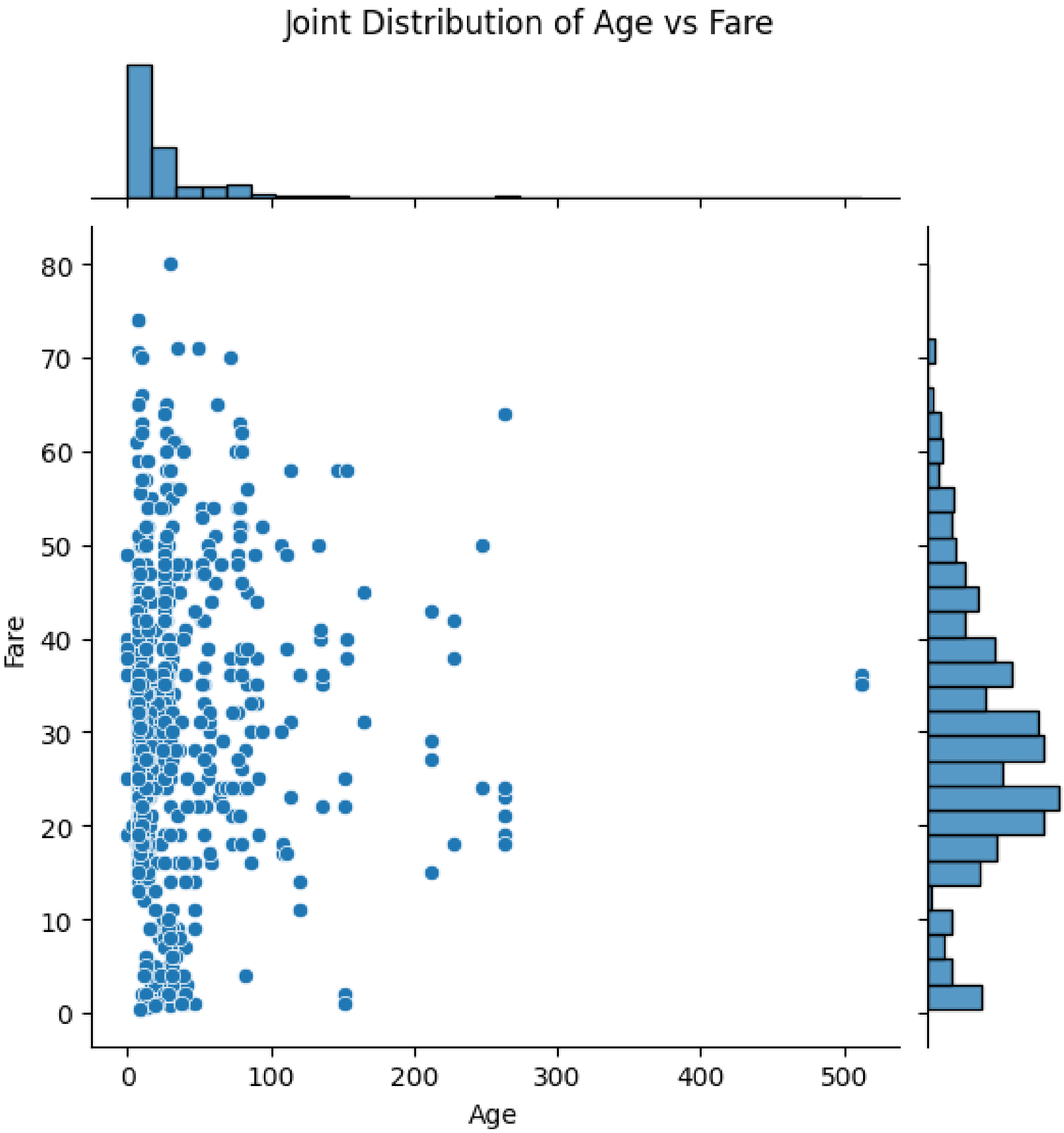
```
#Lab22_2b
# Create a heatmap of all the variables
plt.figure(figsize=(10, 6))
sns.heatmap(titanic.corr(numeric_only=True), annot=False, cmap='coolwarm',␣
 ↪linewidths=0.5)
plt.title('Heatmap of Titanic Dataset')
plt.show()
```



[41]:
```
#Lab22_2c
# Create a histogram with KDE for Age distribution
plt.figure(figsize=(8, 6))
sns.histplot(titanic['age'].dropna(), kde=True, bins=10, color='steelblue',␣
 ↪edgecolor='black')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



```
[42]: #Lab22_2d
g = sns.jointplot(data=titanic, x="fare", y="age",kind="scatter",␣
 ↪marginal_kws=dict(bins=30, fill=True))
g.fig.suptitle('Joint Distribution of Age vs Fare', y=1.02)
g.set_axis_labels('Age', 'Fare')
plt.show()
```



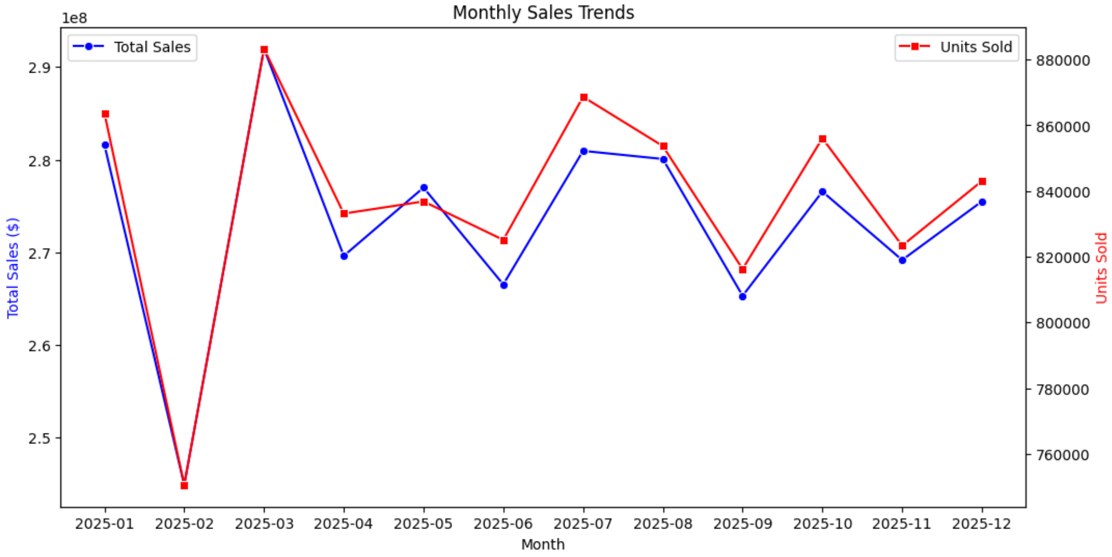
## 6 LAB 22 QUESTION 3 GEN AI

[43]:

```
#Lab22_3
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("Gen_AI_sales_dataset.csv")
```

[44]:

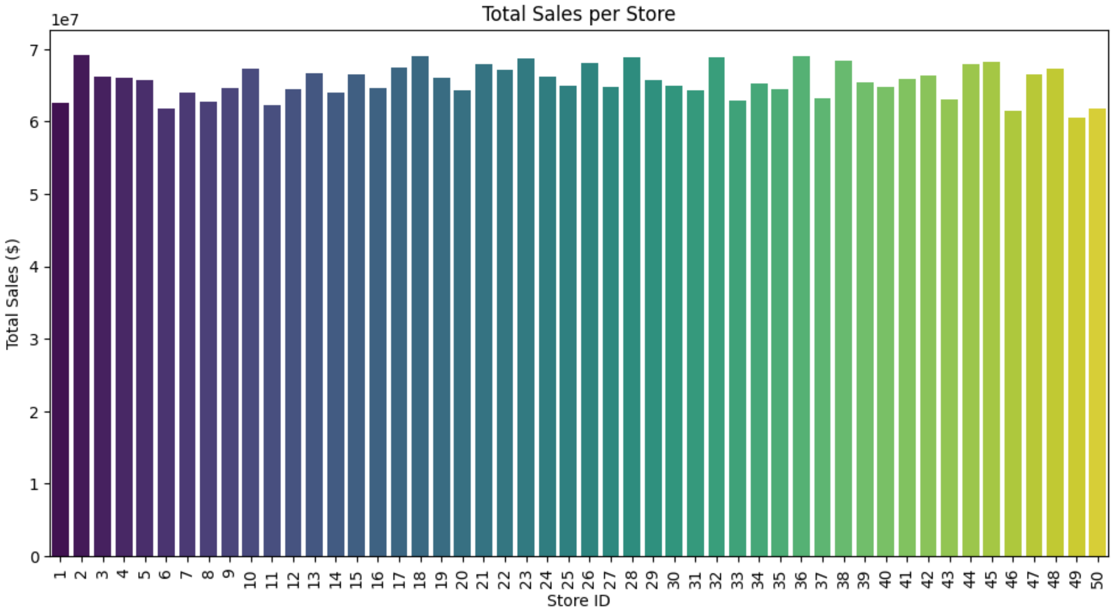
```
#Lab22_3a
# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Aggregate data by month
df['month'] = df['date'].dt.to_period('M')
monthly_sales = df.groupby('month').agg({'total_sales': 'sum', 'units_sold':␣
 ↪'sum'}).reset_index()
monthly_sales['month'] = monthly_sales['month'].astype(str)

# Plot dual-axis line chart
fig, ax1 = plt.subplots(figsize=(12, 6))
ax2 = ax1.twinx()

sns.lineplot(x='month', y='total_sales', data=monthly_sales, ax=ax1, color='b',␣
 ↪marker='o', label='Total Sales')
sns.lineplot(x='month', y='units_sold', data=monthly_sales, ax=ax2, color='r',␣
 ↪marker='s', label='Units Sold')

# Labels and title
ax1.set_xlabel('Month')
ax1.set_ylabel('Total Sales ($)', color='b')
ax2.set_ylabel('Units Sold', color='r')
plt.title('Monthly Sales Trends')
plt.xticks(rotation=45)
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```

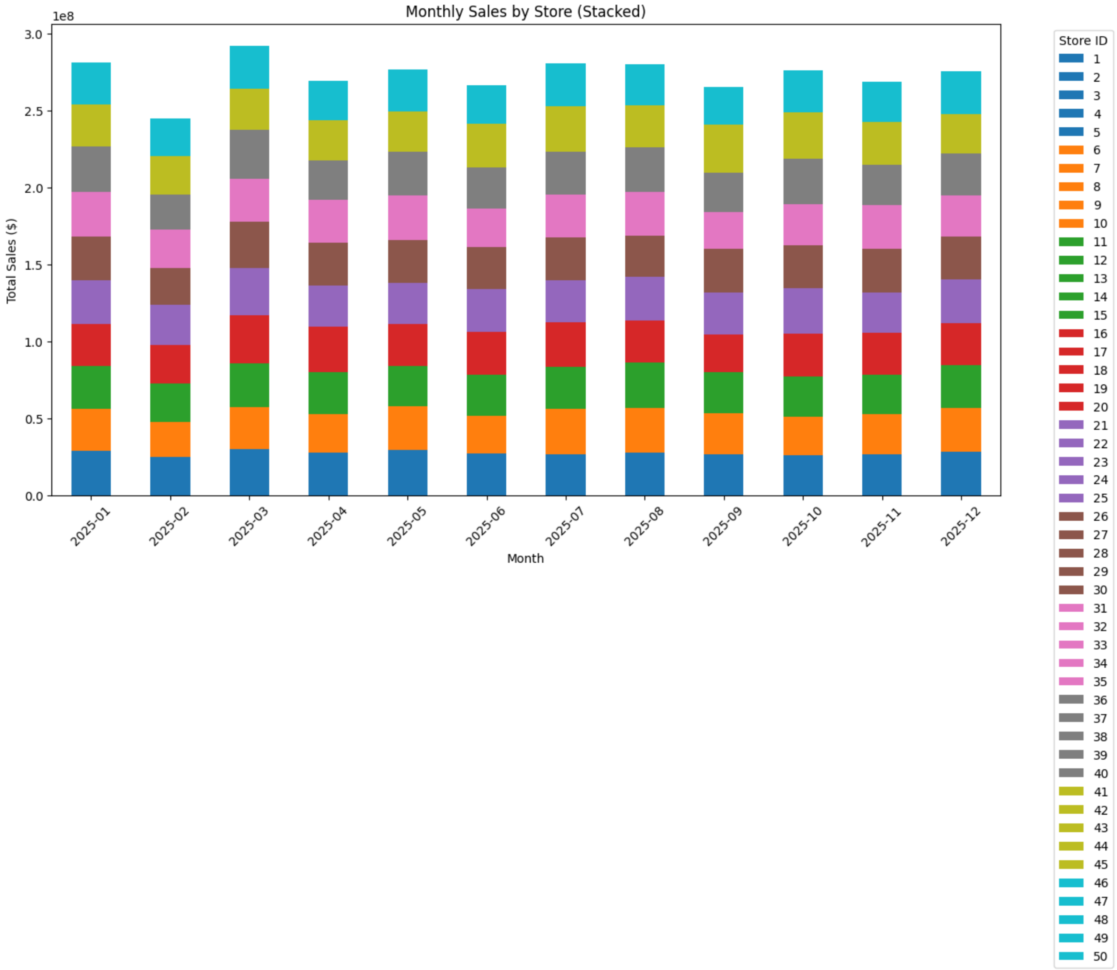


[45]:

```
#Lab22_3b
# Sales by Store - Bar Chart
total_sales_by_store = df.groupby('store_id')['total_sales'].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.barplot(x='store_id', y='total_sales', data=total_sales_by_store,␣
  ↪palette='viridis')
plt.xlabel('Store ID')
plt.ylabel('Total Sales ($)')
plt.title('Total Sales per Store')
plt.xticks(rotation=90)
plt.show()

# Sales Performance by Store Over Time (Stacked Bar Chart)
monthly_store_sales = df.groupby(['month', 'store_id'])['total_sales'].sum().
  ↪unstack().fillna(0)
monthly_store_sales.plot(kind='bar', stacked=True, figsize=(14, 7),␣
  ↪colormap='tab10')
plt.xlabel('Month')
plt.ylabel('Total Sales ($)')
plt.title('Monthly Sales by Store (Stacked)')
plt.xticks(rotation=45)
plt.legend(title='Store ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

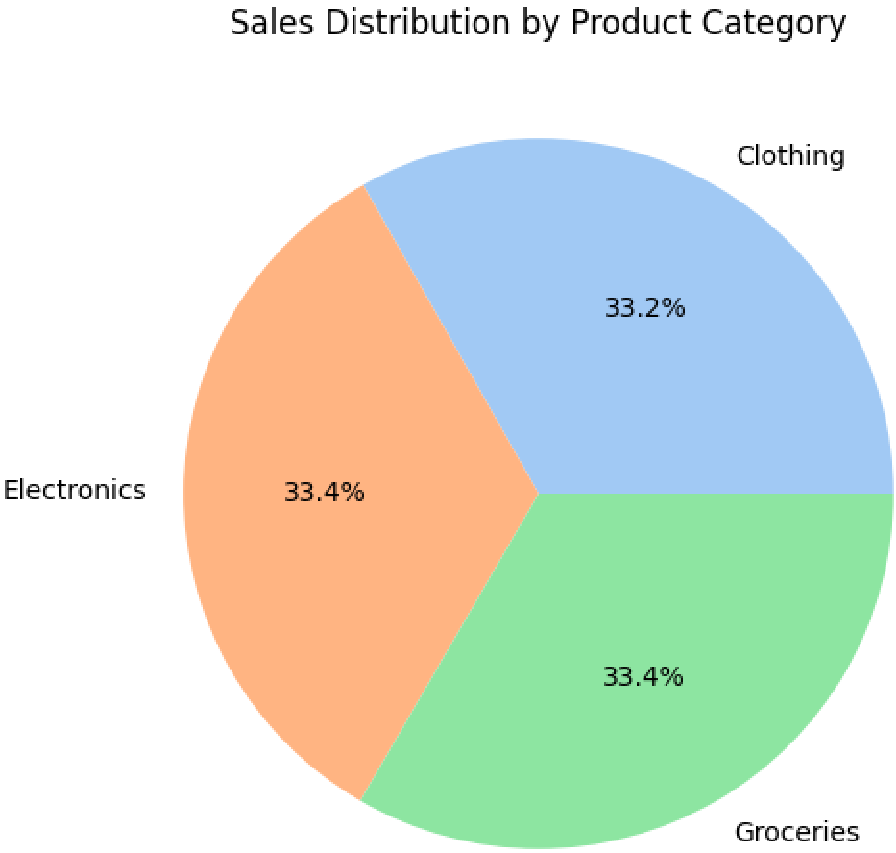
```
/var/folders/9m/f5jl8xnd4ls2_3ykntk1_rpm0000gn/T/ipykernel_10622/1364985754.py:5
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='store_id', y='total_sales', data=total_sales_by_store,
palette='viridis')
```


Total Sales per Store
1e7
7
6
5
4
3
2
1
0
Total Sales ($)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Store ID

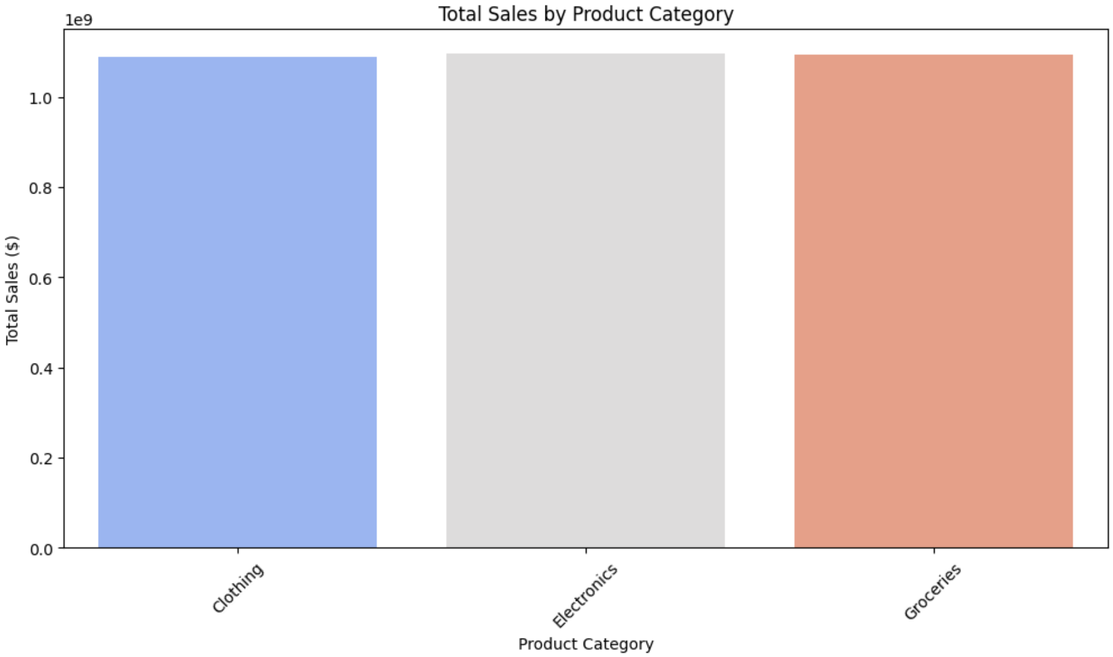

Monthly Sales by Store (Stacked)
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Sales ($)
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
2025-12
Month
Store ID
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

[46]:

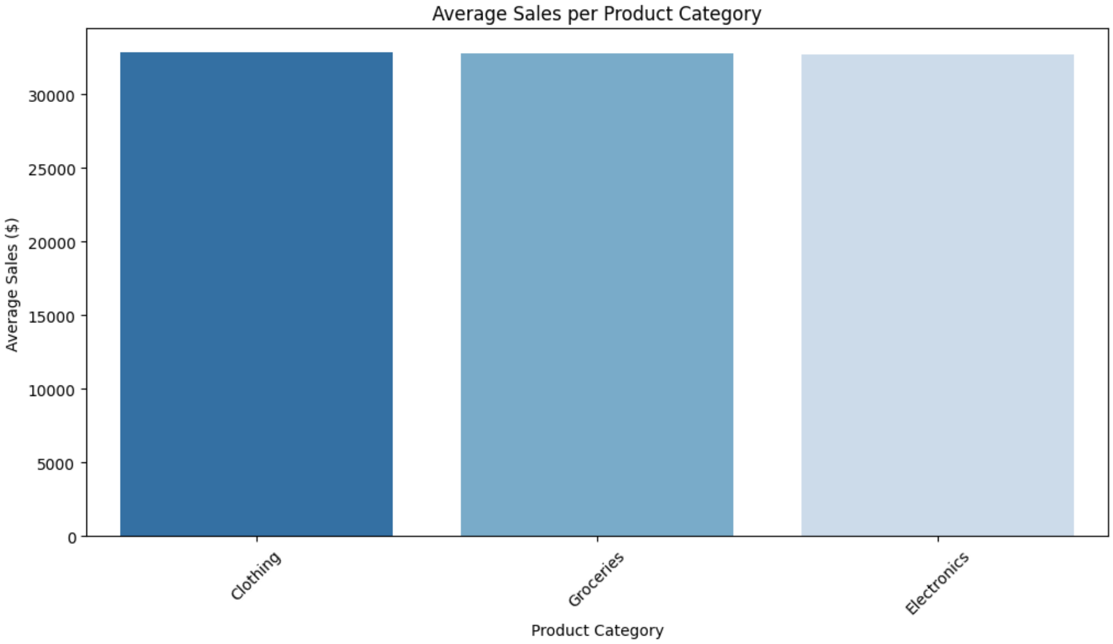
```
#Lab22_3c
# Product Category Performance - Pie Chart
category_sales = df.groupby('product_category')['total_sales'].sum().
 ↪reset_index()
plt.figure(figsize=(10, 6))
plt.pie(category_sales['total_sales'],␣
 ↪labels=category_sales['product_category'], autopct='%1.1f%%', colors=sns.
 ↪color_palette('pastel'))
plt.title('Sales Distribution by Product Category')
plt.show()

# Product Category Performance - Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(x='product_category', y='total_sales', data=category_sales,␣
 ↪palette='coolwarm')
plt.xlabel('Product Category')
plt.ylabel('Total Sales ($)')
plt.title('Total Sales by Product Category')
plt.xticks(rotation=45)
plt.show()

# Average Sales per Product Category
avg_sales_by_category = df.groupby('product_category')['total_sales'].mean().
 ↪reset_index().sort_values(by='total_sales', ascending=False)
plt.figure(figsize=(12, 6))
sns.barplot(x='product_category', y='total_sales', data=avg_sales_by_category,␣
 ↪palette='Blues_r')
plt.xlabel('Product Category')
plt.ylabel('Average Sales ($)')
plt.title('Average Sales per Product Category')
plt.xticks(rotation=45)
plt.show()
```



```
/var/folders/9m/f5jl8xnd4ls2_3ykntk1_rpm0000gn/T/ipykernel_10622/1355883582.py:1
1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='product_category', y='total_sales', data=category_sales,
palette='coolwarm')
```

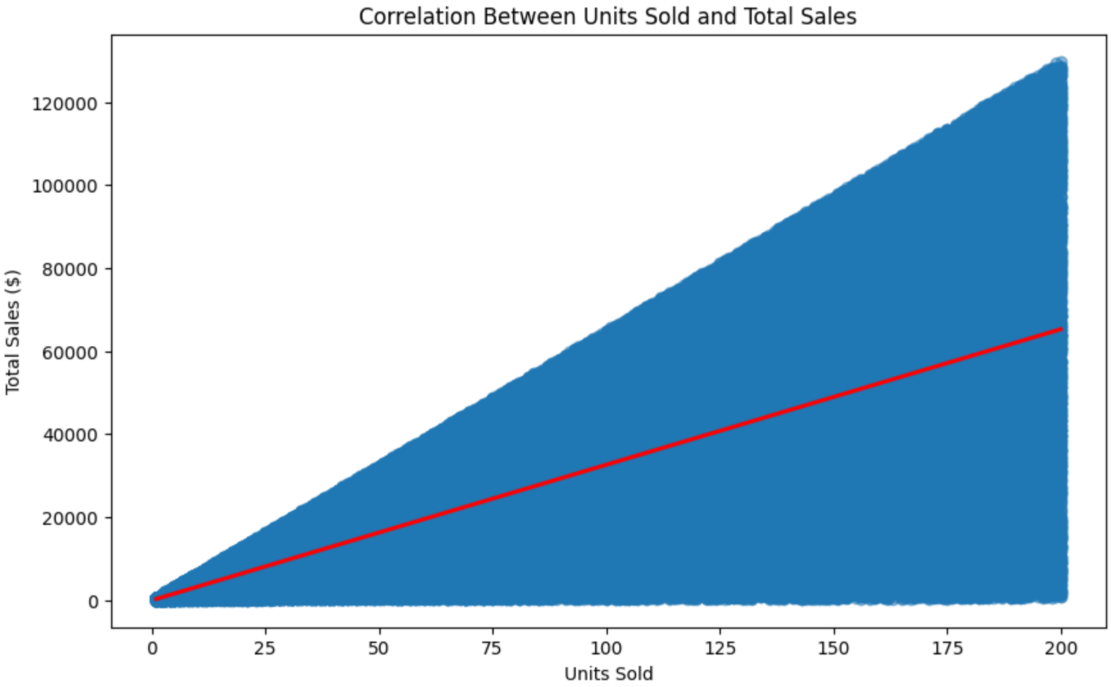


```
/var/folders/9m/f5jl8xnd4ls2_3ykntk1_rpm0000gn/T/ipykernel_10622/1355883582.py:2
1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='product_category', y='total_sales', data=avg_sales_by_category,
palette='Blues_r')
```



[47]:
```python
import scipy.stats as stats
```

[48]:
```python
#Lab22_3d
# Sales Correlation Analysis - Scatter Plot with Regression Line
plt.figure(figsize=(10, 6))
sns.regplot(x='units_sold', y='total_sales', data=df, scatter_kws={'alpha':0.
  ↪5}, line_kws={'color':'red'})
plt.xlabel('Units Sold')
plt.ylabel('Total Sales ($)')
plt.title('Correlation Between Units Sold and Total Sales')
plt.show()

# Calculate correlation coefficient
correlation, p_value = stats.pearsonr(df['units_sold'], df['total_sales'])
print(f'Correlation Coefficient: {correlation:.2f}, P-value: {p_value:.5f}')
```

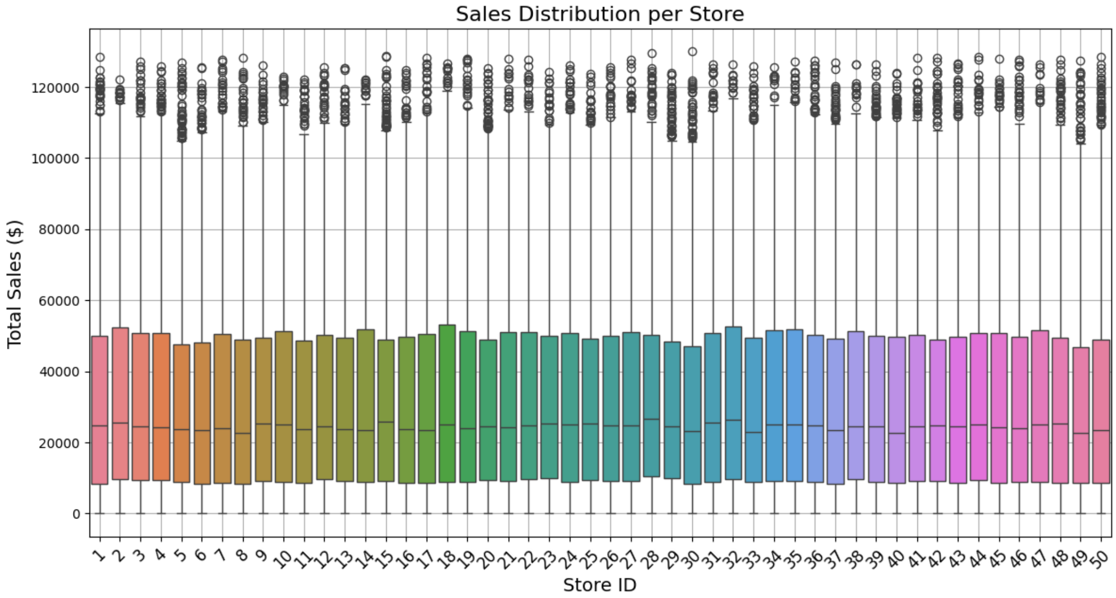


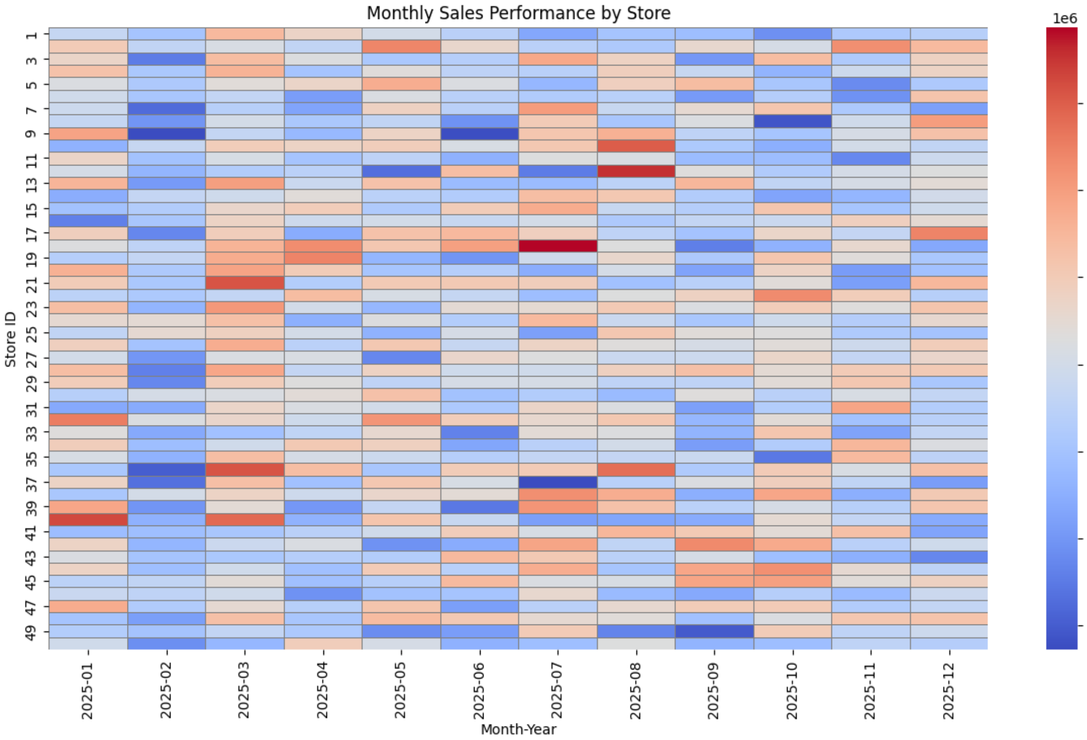
```
Correlation Coefficient: 0.66, P-value: 0.00000
```

[50]:

```python
#Lab22_3e
import matplotlib.pyplot as plt
import seaborn as sns

# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Extract year and month for grouping
df['year_month'] = df['date'].dt.to_period('M')

# Aggregate total sales per store per month
store_sales = df.groupby(['store_id', 'year_month'])['total_sales'].sum().reset_index()

# Convert year_month to string format for better plotting
store_sales['year_month'] = store_sales['year_month'].astype(str)

# Plot boxplot for sales distribution by store
plt.figure(figsize=(14, 7))
sns.boxplot(x='store_id', y='total_sales', data=df, palette='husl')  # Use a vibrant color palette
plt.xlabel('Store ID', fontsize=14)
plt.ylabel('Total Sales ($)', fontsize=14)
plt.title('Sales Distribution per Store', fontsize=16)
plt.xticks(rotation=45, fontsize=12)
plt.grid(True)
plt.show()

# Pivot the data to create a matrix of sales per store per month
sales_pivot = store_sales.pivot(index='store_id', columns='year_month',␣
 ↪values='total_sales')

# Plot heatmap
plt.figure(figsize=(15, 8))
sns.heatmap(sales_pivot, cmap='coolwarm', linewidths=0.5, linecolor='gray',␣
 ↪annot=False)
plt.xlabel('Month-Year')
plt.ylabel('Store ID')
plt.title('Monthly Sales Performance by Store')
plt.xticks(rotation=90)
plt.show()
```

```
/var/folders/9m/f5jl8xnd4ls2_3ykntk1_rpm0000gn/T/ipykernel_10622/2281572989.py:1
9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='store_id', y='total_sales', data=df, palette='husl')  # Use a
vibrant color palette
```




Monthly Sales Performance by Store
Store ID
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
2025-12
Month-Year
1e6
7.0
6.5
6.0
5.5
5.0
4.5
4.0