Sales_Exploratory Data Analysis Project

In [2]:
```
import pandas as pd # Import the pandas library and os module
import os
```

In [16]:
```
file_path = r"C:\\Project\\Data\\Sales_Data\\" # Define the file path for the sales data directory using a raw string

# Print the file path
print(file_path)
```

```
C:\\Project\\Data\\Sales_Data\\
```

In [18]:
```
# Create an empty DataFrame to store the sales data
sales_df = pd.DataFrame()

# List all files in the specified directory
files = os.listdir(file_path)

# Loop through each file in the directory
for file in files:
    # Read the CSV file and create a DataFrame
    df = pd.read_csv(file_path + file)
    #print(df)

    # Concatenate the current DataFrame with the overall sales DataFrame
    sales_df = pd.concat((sales_df, df))
    #print(files_df.shape)
```

In [19]:
```
sales_df.head()
```

Out[19]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

In [20]:
```
sales_df.shape # Here I am checking shape of data
```

Out[20]:
```
(186850, 6)
```

Q 1: find the month in which the most sales was done ?

Data Cleaning Activities

In [8]:
```
# Code to convert
# Quantity Ordered to 'int'
# Price Each to float
```

In [21]:
```
# I am checking null values/ NAN values
# Here I am dropping NAN using dropna method of 'Quantity Ordered'
sales_df.dropna(how="any", subset = ["Quantity Ordered"], inplace = True)

sales_df['Quantity Ordered'].isnull().sum()
```

Out[21]:
```
0
```

In [22]:
```
# Here I am dropping NAN using drona method
sales_df.dropna(how='any',subset =['Quantity Ordered'], inplace = True)
# To check the null values
sales_df['Quantity Ordered'].isnull().sum()
```

Out[22]:
```
0
```

In [23]:
```
sales_df.isnull().sum() # Check the count of null values in each column of the DataFrame
```

Out[23]:
```
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
dtype: int64
```

In [24]:
```
sales_df.head()
```

Out[24]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

In [25]:
```
sales_df.dtypes # Here I am checking the data types of the columns
```

Out[25]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

In [26]:
```
sales_df.dropna(how='any',subset =['Quantity Ordered'], inplace = True) # Drop rows with missing values in the 'Quantity Ordered' column
sales_df['Quantity Ordered'].isnull().sum() # Check the count of null values in the 'Quantity Ordered' column after dropping rows
```

Out[26]:
```
0
```

In [27]:
```
# From the above error- invalid literal for int() with base 10: 'Quantity Ordered'
# First Think I need to check is there 'Quantity Ordered' in the column of "Quantity Ordered"
sales_df.loc[(sales_df['Quantity Ordered'] == "Quantity Ordered"),:]
```

Out[27]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
519	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2878	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2893	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
...	...	...	...	...	...	...
10000	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
10387	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
11399	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
11468	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
11574	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

355 rows × 6 columns

In [29]:
```
sales_df = sales_df.loc[~(sales_df['Quantity Ordered'] == "Quantity Ordered"), :] # Here I have drop 'Quatity ordered from sales column'
```

In [30]:
```
sales_df
```

Out[30]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

185950 rows × 6 columns

In [31]:
```
sales_df['Quantity Ordered']= sales_df['Quantity Ordered'].astype(int) # Convert the 'Quantity Ordered' column to integer type
```

In [32]:
```
sales_df['Price Each'] = sales_df['Price Each'].astype(float) # Convert the "Price Each" column to float type
```

In [33]:
```
sales_df['Total_sell'] = sales_df['Quantity Ordered'] * sales_df['Price Each'] # Multipy 'Quantity Ordered' and 'Price Each' to get 'Total Sell'
```

In [34]:
```
sales_df
```

Out[34]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Total_sell
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	11.99
...	...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	8.97
11682	259354	iPhone	1	700.00	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	700.00
11683	259355	iPhone	1	700.00	09/23/19 07:39	220 12th St, San Francisco, CA 94016	700.00
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	379.99
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	11.95

185950 rows × 7 columns

In [36]:
```
# Convert the 'Order Date' column to datetime format
sales_df['Order Date']= pd.to_datetime(sales_df['Order Date'])
```

In [37]:
```
# Extract the month from the 'Order Date' and create a new 'Month' column
sales_df['Month']= sales_df['Order Date'].dt.month
```

In [38]:
```
sales_df.head()
```

Out[38]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Total_sell	Month
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	23.90	4
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	99.99	4
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	600.00	4
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	11.99	4
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	11.99	4

In [52]:
```
# Group by month and calculate the total sales
sale_by_month = sales_df.groupby(["Month"])["Total_sell"].sum()
```

In [55]:
```
# Find the month with the highest total sales
best_month = sale_by_month.idxmax()
```

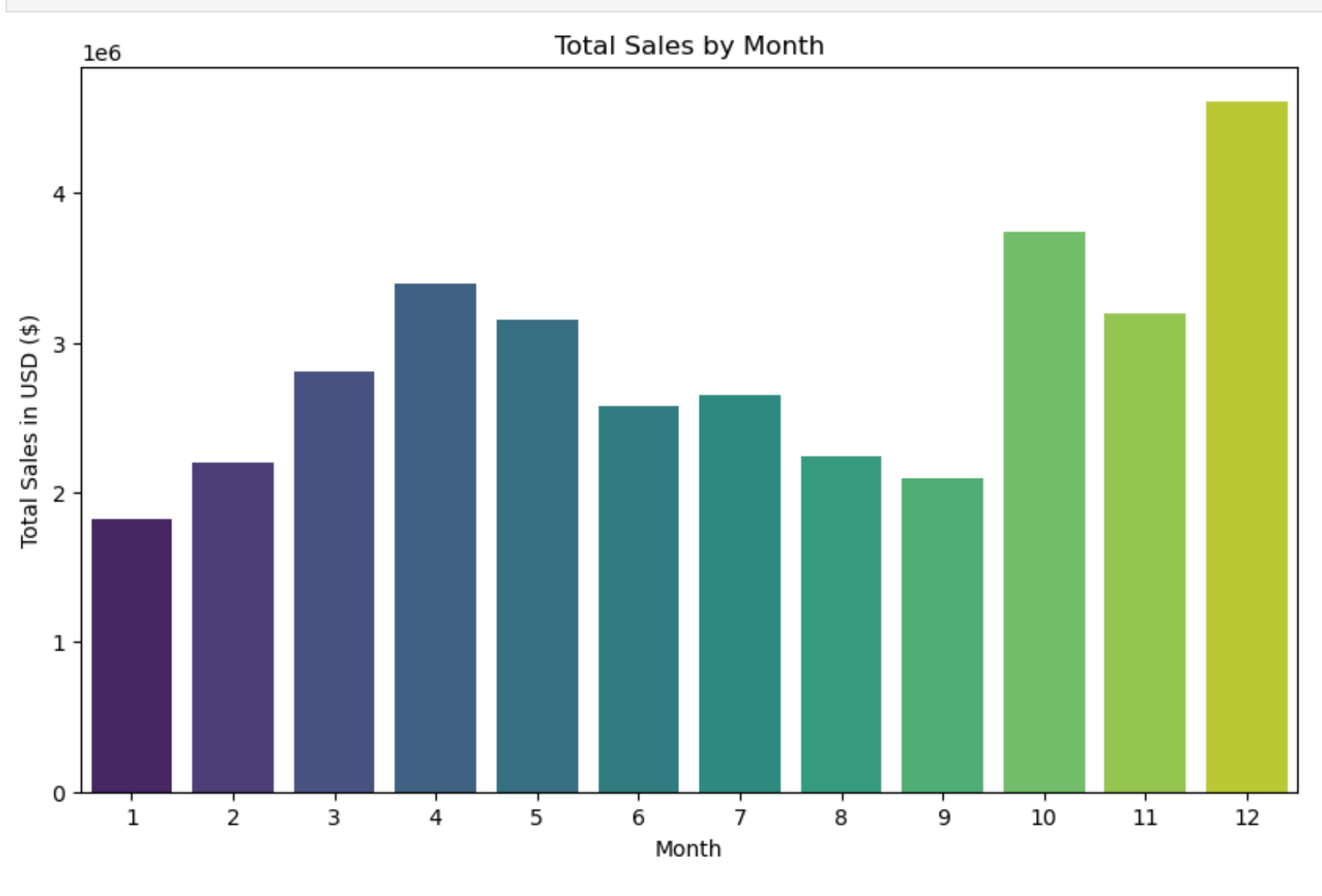
In [56]:
```
# Print the result
print(f"The month with the most sales is: {best_month}")
```

```
The month with the most sales is: 12
```

In [51]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [53]:
```
# Extract months and sales amounts
months = sale_by_month.index
sales_amt = list(sale_by_month)
```

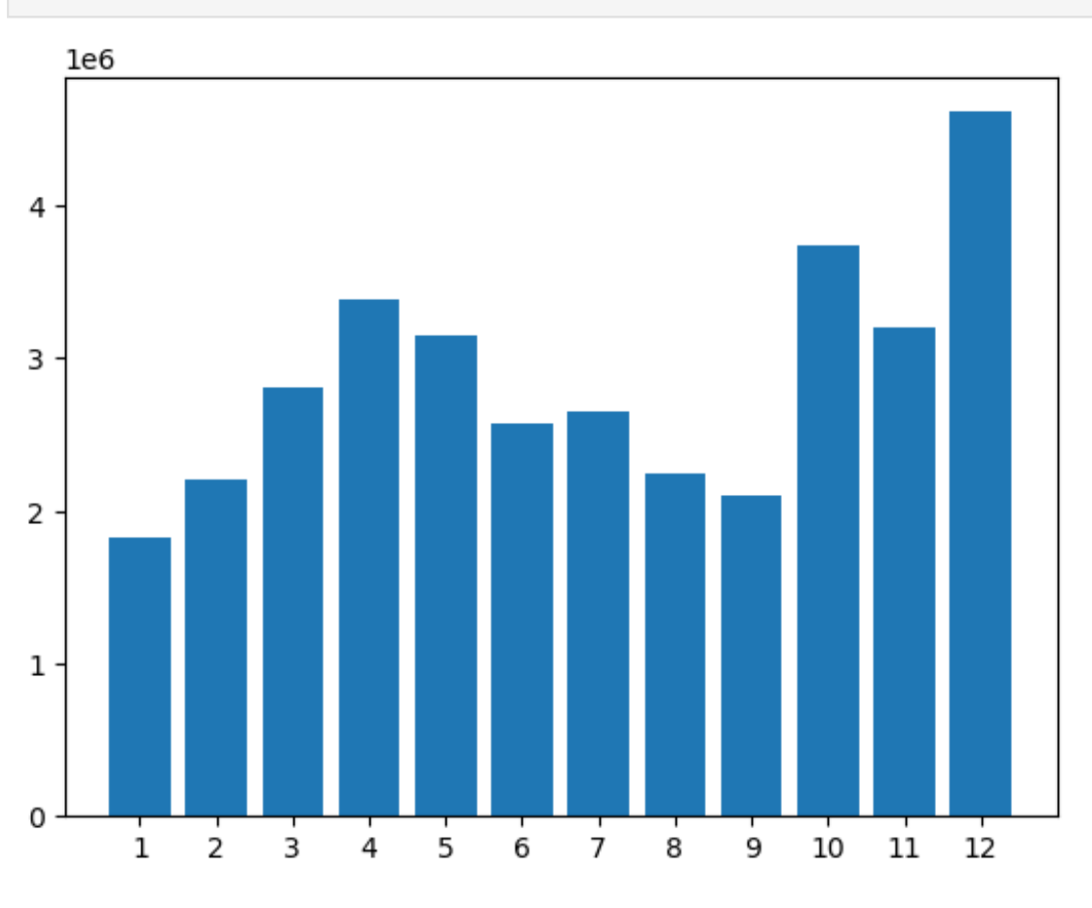
In [54]:
```
# Plot the bar graph using Seaborn and Matplotlib
plt.figure(figsize=(10, 6))
sns.barplot(x=months, y=sales_amt, palette="viridis")
plt.title('Total Sales by Month')
plt.xlabel('Month')
plt.ylabel('Total Sales in USD ($)')
plt.show()
```



Ans The month with the most sales is: 12

In [46]:
```
plt.bar(months, sales_amt)

plt.xticks(months)
plt.show()
```



Q2: Top five product which have maximum sales

In [57]:
```
sales_df['Product']= sales_df['Product'].str.lower().str.strip()
```

In [58]:
```
sales_df
```

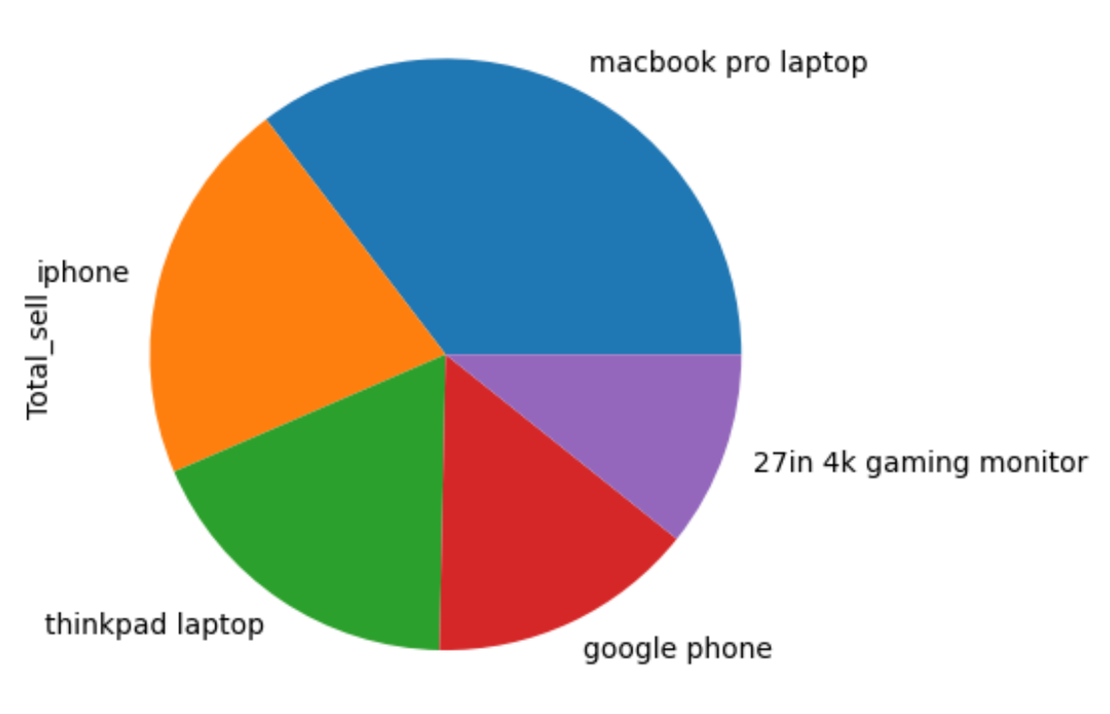
Out[58]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Total_sell	Month
0	176558	usb-c charging cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	23.90	4
2	176559	bose soundsport headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	99.99	4
3	176560	google phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	600.00	4
4	176560	wired headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	11.99	4
5	176561	wired headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	11.99	4
...	...	...	...	...	...	...	...	...
11681	259353	aaa batteries (4-pack)	3	2.99	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	8.97	9
11682	259354	iphone	1	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	700.00	9
11683	259355	iphone	1	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	700.00	9
11684	259356	34in ultrawide monitor	1	379.99	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	379.99	9
11685	259357	usb-c charging cable	1	11.95	2019-09-30 00:18:00	250 Meadow St, San Francisco, CA 94016	11.95	9

185950 rows × 8 columns

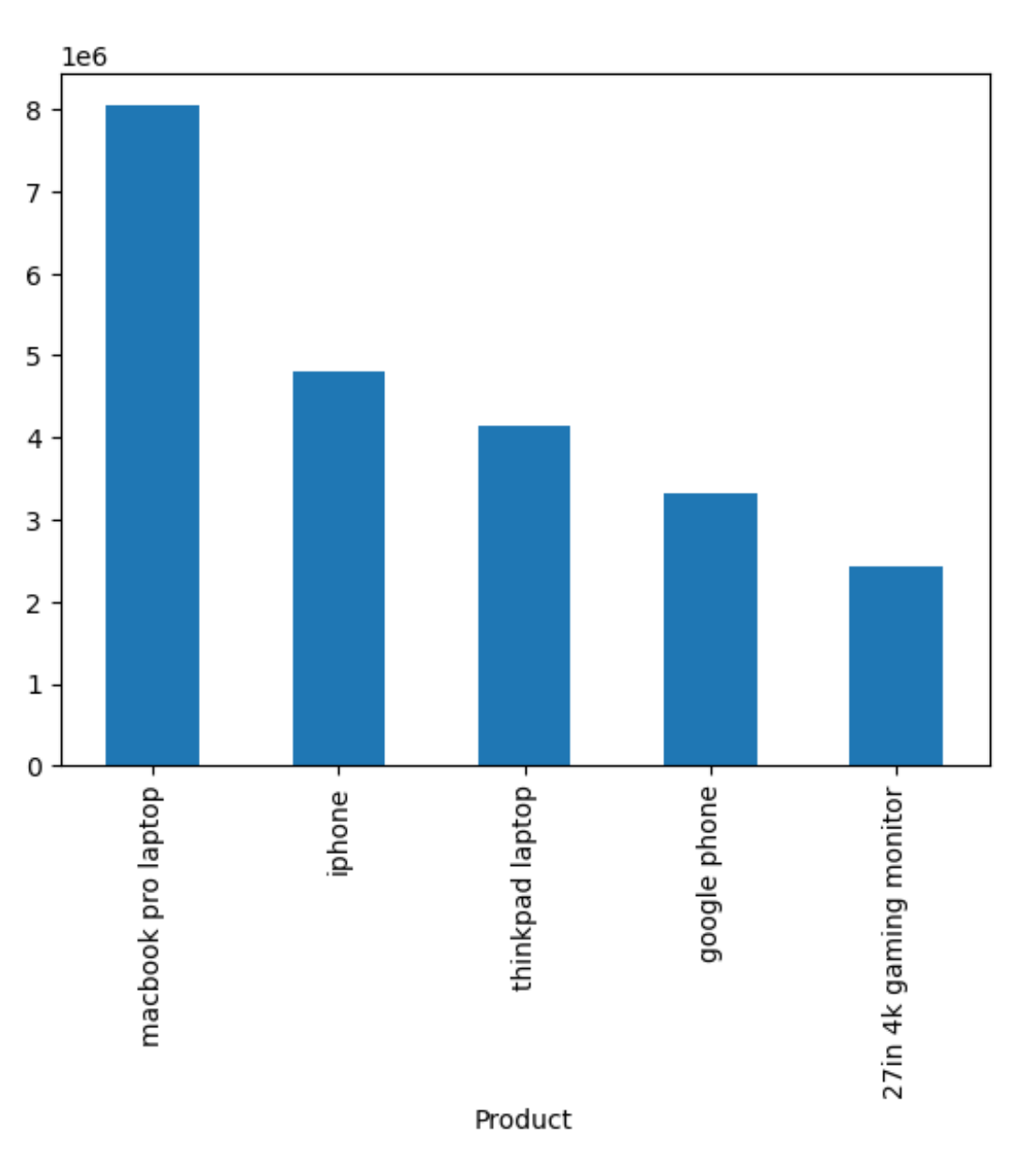
In [59]:
```
sales_df.groupby(['Product'])['Total_sell'].sum().sort_values(ascending= False).head().plot(kind='pie')
```

Out[59]:
```
<AxesSubplot:ylabel='Total_sell'>
```



In [60]:
```
sales_df.groupby(['Product'])['Total_sell'].sum().sort_values(ascending = False).head().plot(kind='bar')
```

Out[60]:
```
<AxesSubplot:xlabel='Product'>
```



Q 3: Which city has maximum sales?

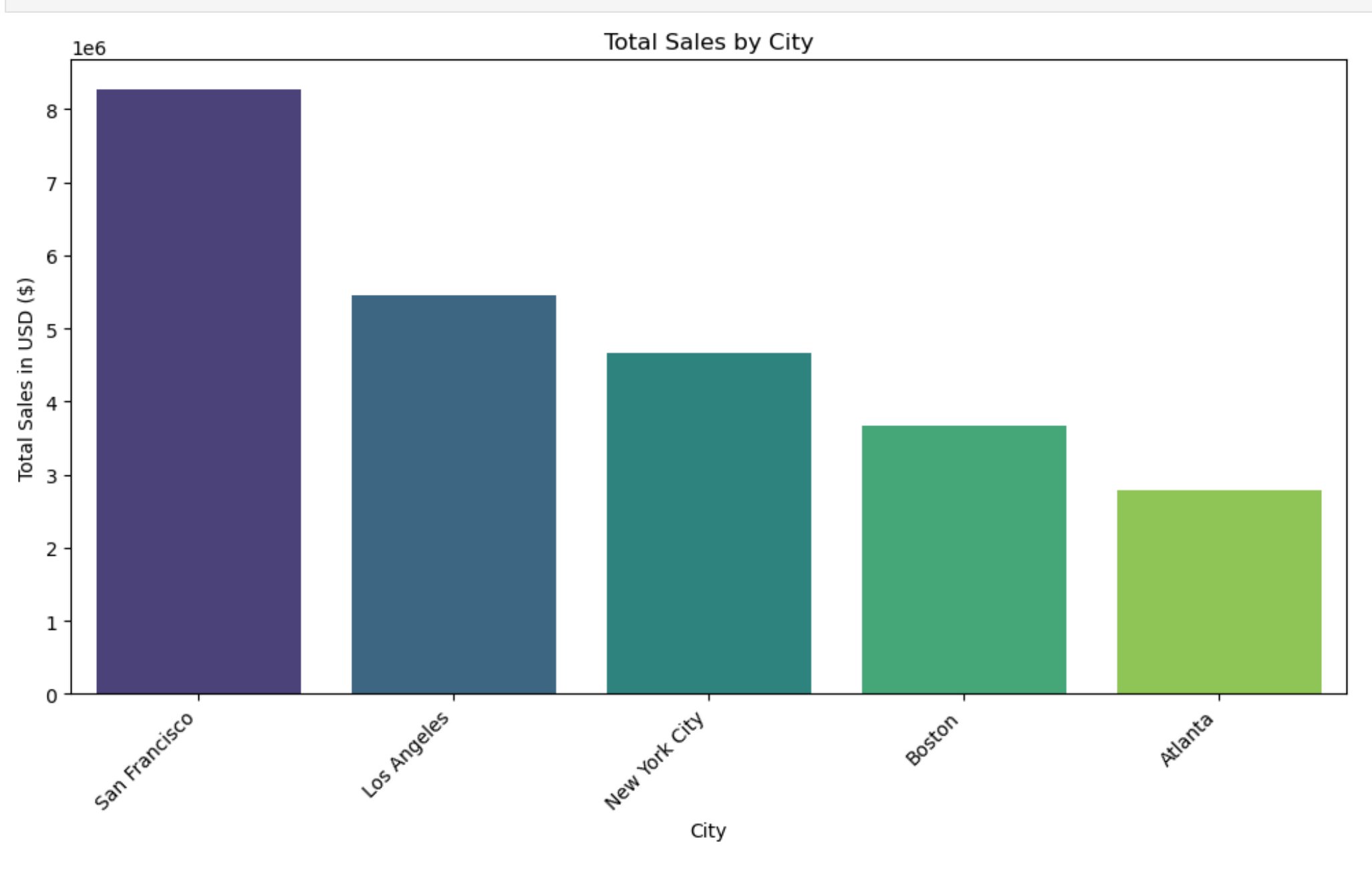
In [82]:
```
# Function to extract city from address
def get_city(address):
    return address.split(',')[1].strip()
```

In [83]:
```
# Apply the function to 'Purchase Address' and create 'city' column
sales_df['city'] = sales_df['Purchase Address'].apply(get_city)
```

In [84]:
```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [85]:
```
# Group by 'city' and calculate the total sales
results = sales_df.groupby(['city'])['Total_sell'].sum().sort_values(ascending=False).head()
```

In [86]:
```
# Plot the bar graph using Seaborn and Matplotlib
plt.figure(figsize=(12, 6))
sns.barplot(x=results.index, y=results.values, palette="viridis")
plt.title('Total Sales by City')
plt.xlabel('City')
plt.ylabel('Total Sales in USD ($)')
plt.xticks(rotation=45, ha='right')
plt.show()
```



Q 3: Which State has maximum sales?

In [75]:
```
# Function to extract state from address
def get_state_from_address(address):
    # Sample address format: 333 8th St, Los Angeles, CA 90001
    state_zip = address.split(',')[-1].strip()
    state = state_zip.split(' ')[0].strip()
    return state
```

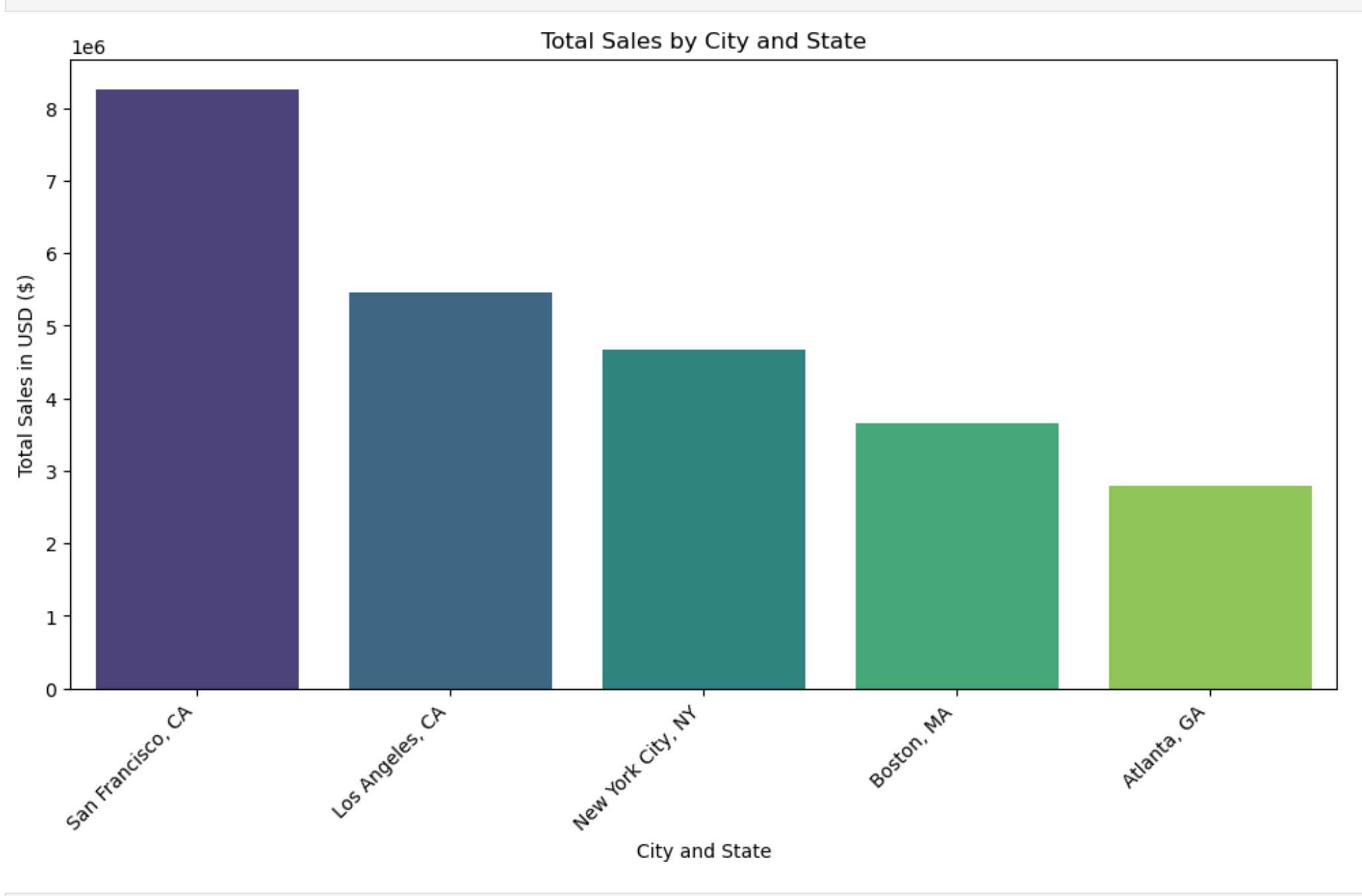
In [76]:
```
# Apply the function to 'Purchase Address' and create 'State_Address' column
sales_df['State_Address'] = sales_df['Purchase Address'].apply(get_state_from_address)
```

In [77]:
```
# Create 'city_state' column by combining 'city' and 'State_Address'
sales_df['city_state'] = sales_df['city'] + ', ' + sales_df['State_Address']
```

In [78]:
```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

In [80]:
```
# Group by 'city_state' and calculate the total sales
results = sales_df.groupby(['city_state'])['Total_sell'].sum().sort_values(ascending=False).head()
```

In [81]:
```
# Plot the bar graph using Seaborn and Matplotlib
plt.figure(figsize=(12, 6))
sns.barplot(x=results.index, y=results.values, palette="viridis")
plt.title('Total Sales by City and State')
plt.xlabel('City and State')
plt.ylabel('Total Sales in USD ($)')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [ ]:
```
# Group by 'city_state' and calculate the total sales
results = sales_df.groupby(['city_state'])['Total_sell'].sum().sort_values(ascending=False).head()
```

In [ ]:

In [69]:
```
sales_df['State_Address']= sales_df['Purchase Address'].apply( get_state_from_address )
```

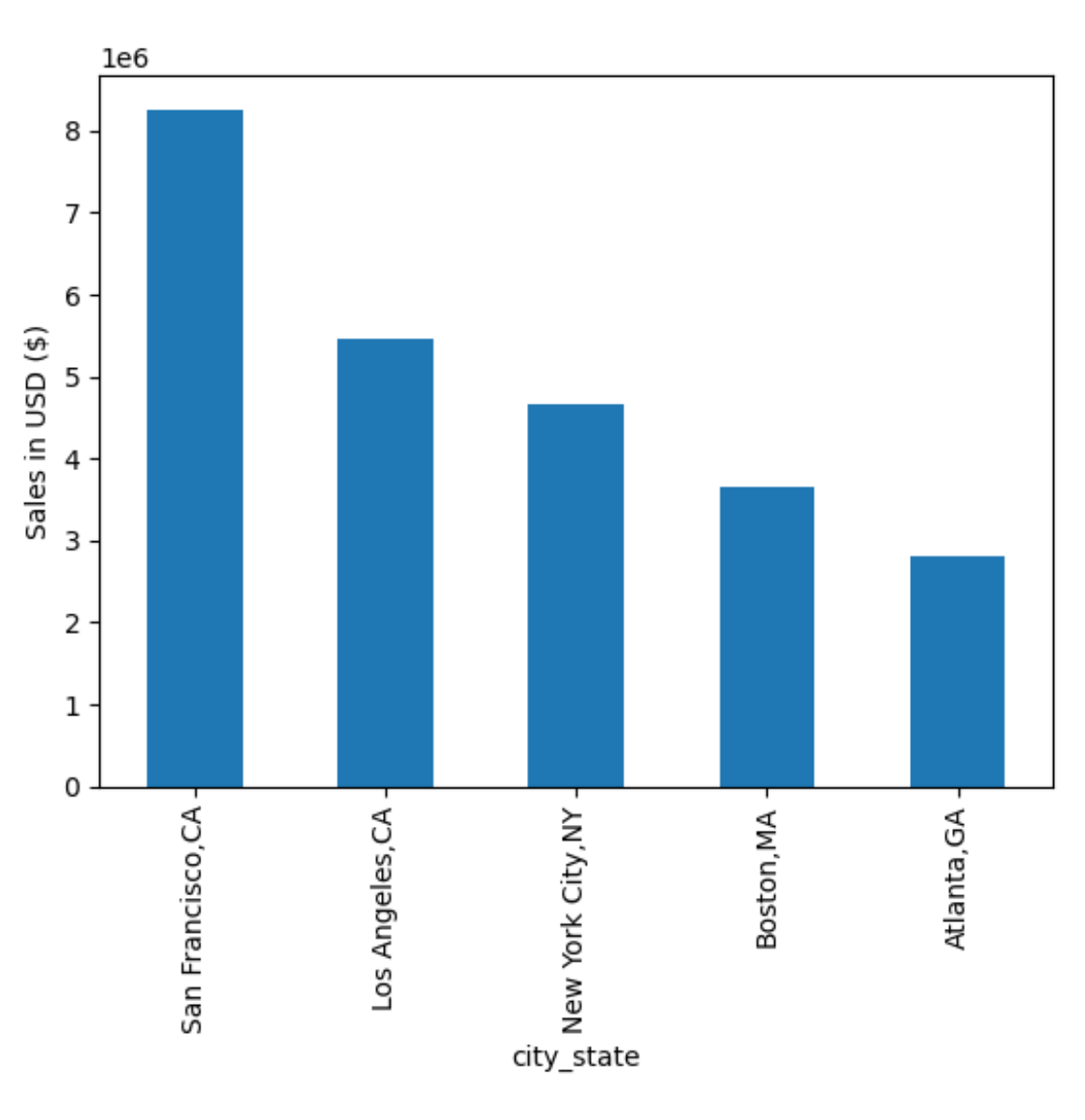
In [70]:
```
sales_df['city_state'] = sales_df['city'] +',' + sales_df['State_Address']
```

In [73]:
```
sales_df['city_state']
```

Out[73]:
```
0                Dallas,TX
2                Boston,MA
3           Los Angeles,CA
4           Los Angeles,CA
5           Los Angeles,CA
                 ...
11681       Los Angeles,CA
11682     San Francisco,CA
11683     San Francisco,CA
11684     San Francisco,CA
11685     San Francisco,CA
Name: city_state, Length: 185950, dtype: object
```

In [74]:
```
import matplotlib.pyplot as plt
import seaborn as sns
results = sales_df.groupby(['city_state'])['Total_sell'].sum().sort_values(ascending= False).head().plot(kind='bar')
plt.xlabel('city_state')
plt.ylabel('Sales in USD ($)')
```

Out[74]:
```
Text(0, 0.5, 'Sales in USD ($)')
```



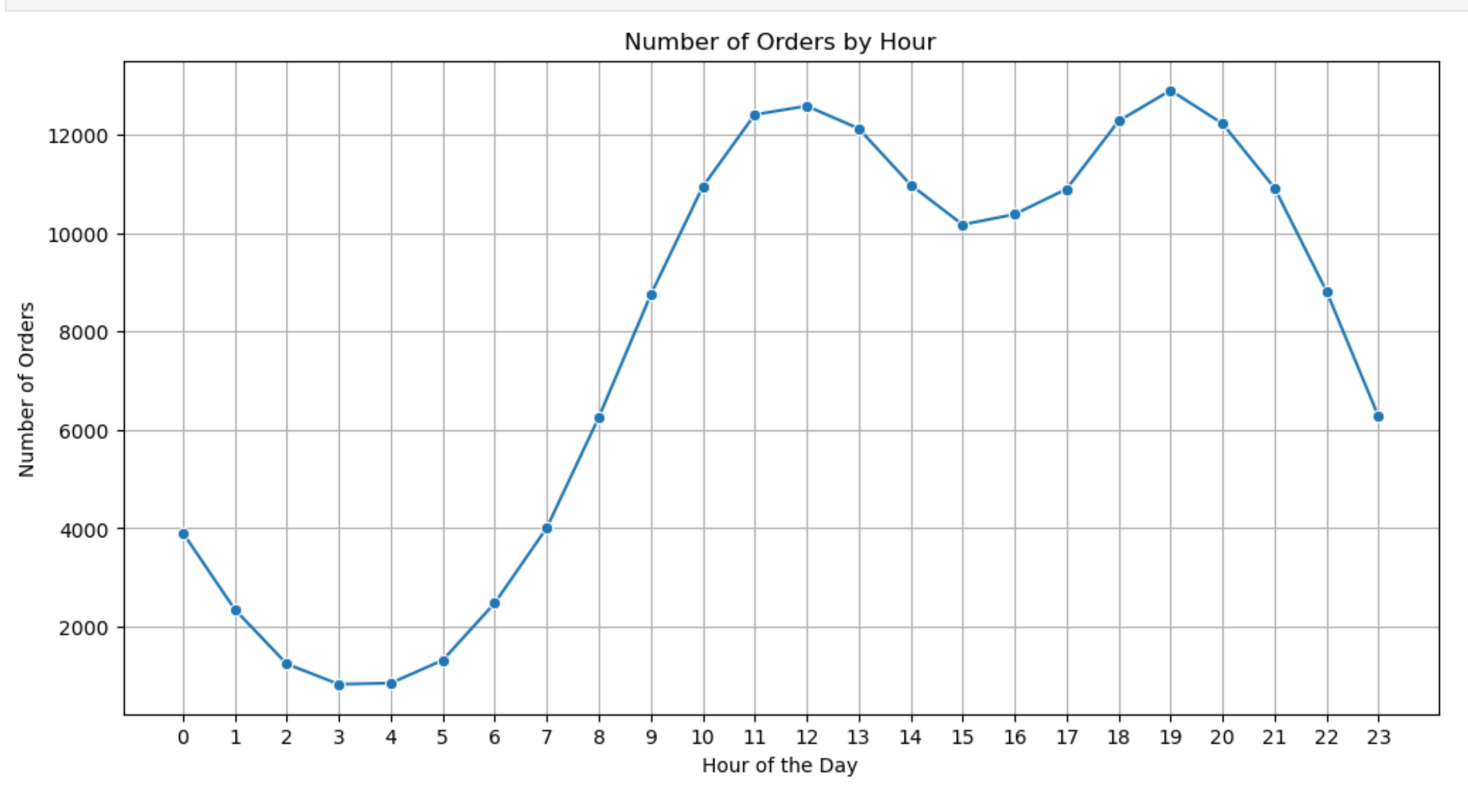
What time should we display advertisements to maximise the likelihood of customer's buying product

In [87]:
```
# Convert 'Order Date' to datetime if not done already
sales_df['Order Date'] = pd.to_datetime(sales_df['Order Date'])
```

In [88]:
```
# Extract the hour from the 'Order Date' and add a new 'Hour' column
sales_df['Hour'] = sales_df['Order Date'].dt.hour
```

In [89]:
```
# Group by hour and calculate the count of orders
orders_by_hour = sales_df.groupby(['Hour']).size()
```

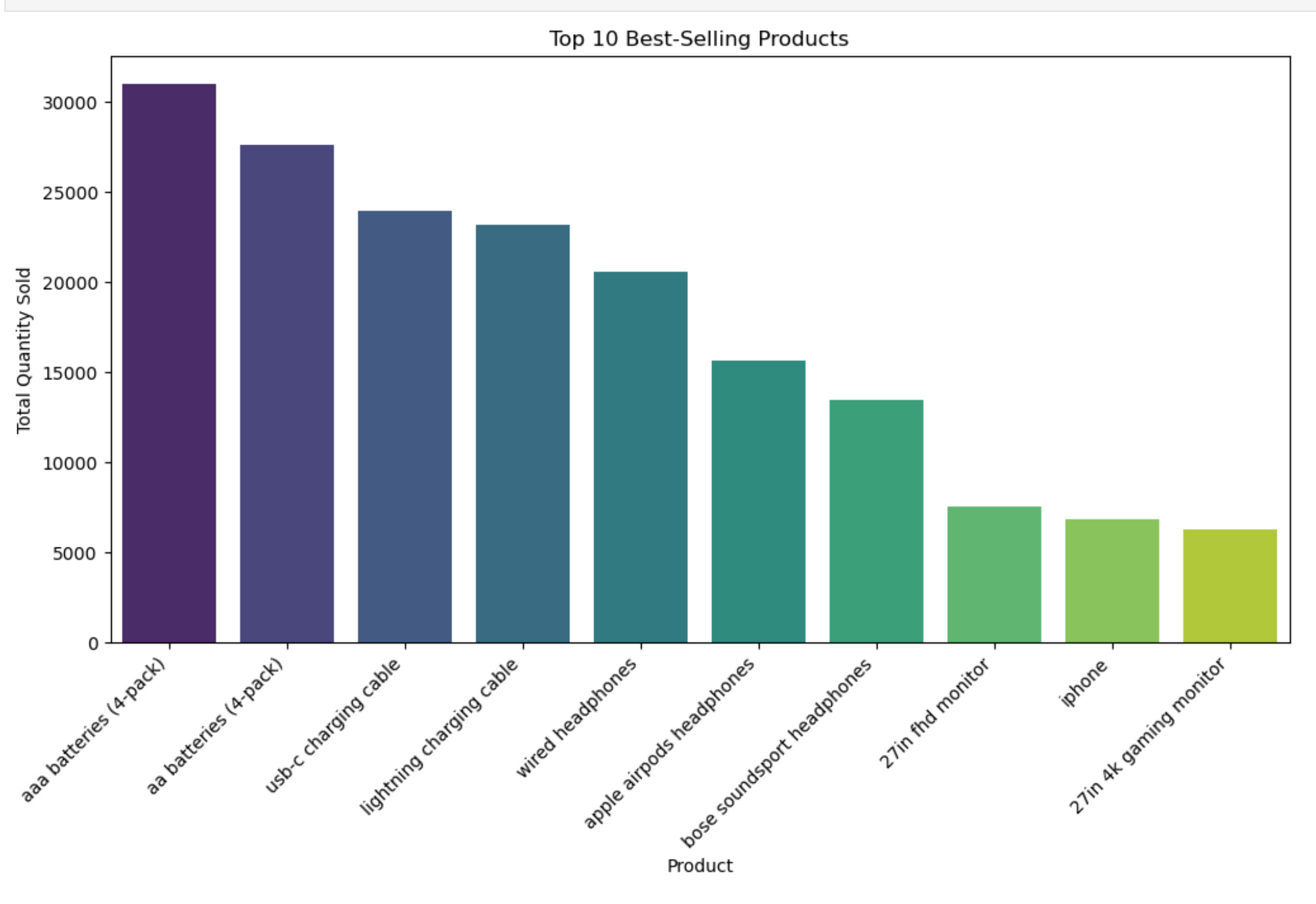
In [90]:
```
# Plot the number of orders by hour
plt.figure(figsize=(12, 6))
sns.lineplot(x=orders_by_hour.index, y=orders_by_hour.values, marker='o')
plt.title('Number of Orders by Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Orders')
plt.xticks(range(24))
plt.grid(True)
plt.show()
```



What product sold the most? Why do you think it sold the most?

In [91]:
```
# Group by product and calculate the total quantity sold
product_sales = sales_df.groupby('Product')['Quantity Ordered'].sum().sort_values(ascending=False)
```

In [92]:
```
# Plot the top-selling products
plt.figure(figsize=(12, 6))
sns.barplot(x=product_sales.head(10).index, y=product_sales.head(10).values, palette="viridis")
plt.title('Top 10 Best-Selling Products')
plt.xlabel('Product')
plt.ylabel('Total Quantity Sold')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [93]:
```
# Identify the product that sold the most
best_selling_product = product_sales.idxmax()
print(f"The product that sold the most is: {best_selling_product}")
```

```
The product that sold the most is: aaa batteries (4-pack)
```