EXPLORATORY DATA ANALYSIS ON SALES DATA SET

Purpose:

- Analyze sales data to identify trends, top-selling products, and revenue metrics for business decision-making.

Description:

- In this project, I will dive into a large sales dataset to extract valuable insights. I will explore sales trends over time, identify the best-selling products, calculate revenue metrics such as total sales and profit margins, and create visualizations to present my findings effectively. This project showcases my ability to manipulate and derive insights from large datasets, enabling me to make data-driven recommendations for optimizing sales strategies.

In [49]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [8]:

```
sales = pd.read_csv("Sales Data.csv",index_col=0)
```

In [11]:

```
sales.shape
```

Out[11]:

```
(185950, 10)
```

- So our new DataFrame has 185950 records split across 10 different columns. We can examine the contents of the resultant DataFrame using the head() command, which grabs the first five rows:

In [9]:

```
sales.head()
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City	0
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City	7
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City	18
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco	15
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta	12

In [30]:

```
print(sales.isnull().sum())
```

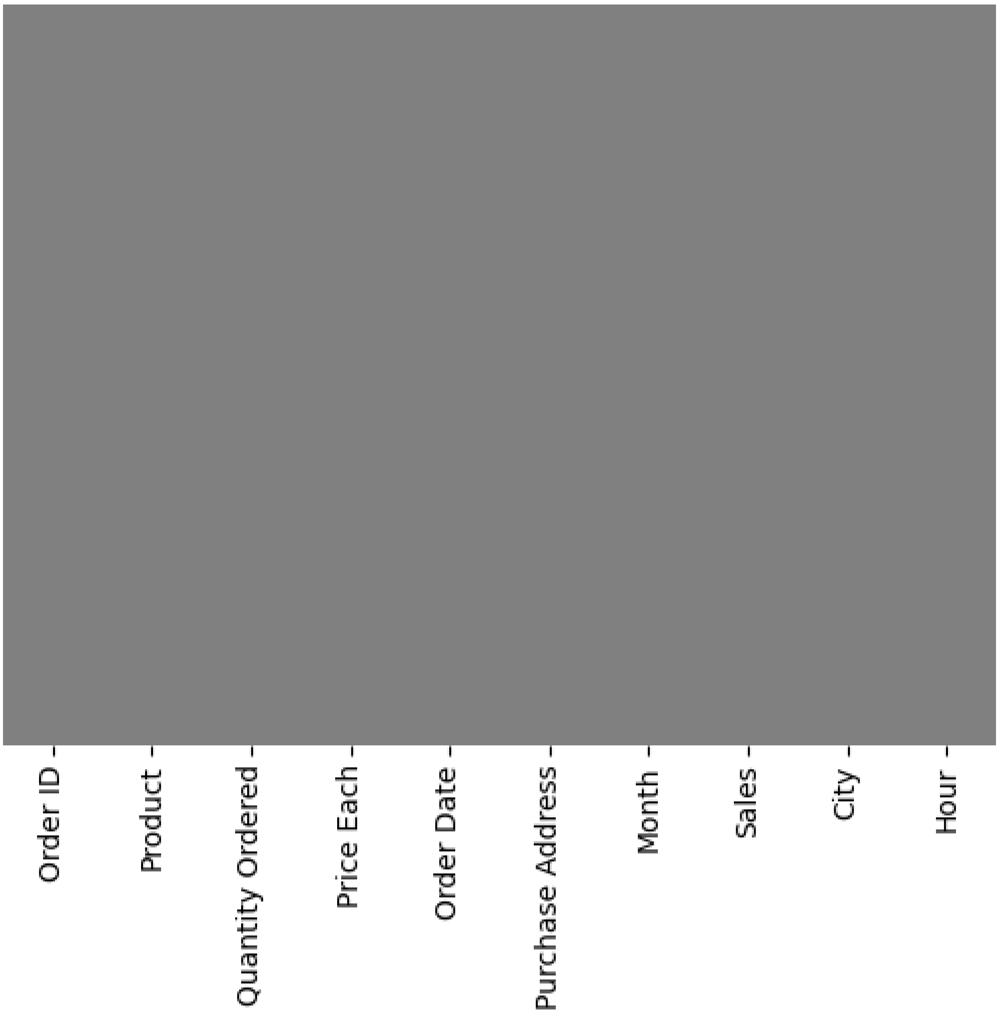
```
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
Month               0
Sales               0
City                0
Hour                0
dtype: int64
```

In [23]:

```
sns.heatmap(sales.isnull(), yticklabels=False, cbar=False, cmap='viridis', linewidt
```

Out[23]:

```
<Axes: >
```

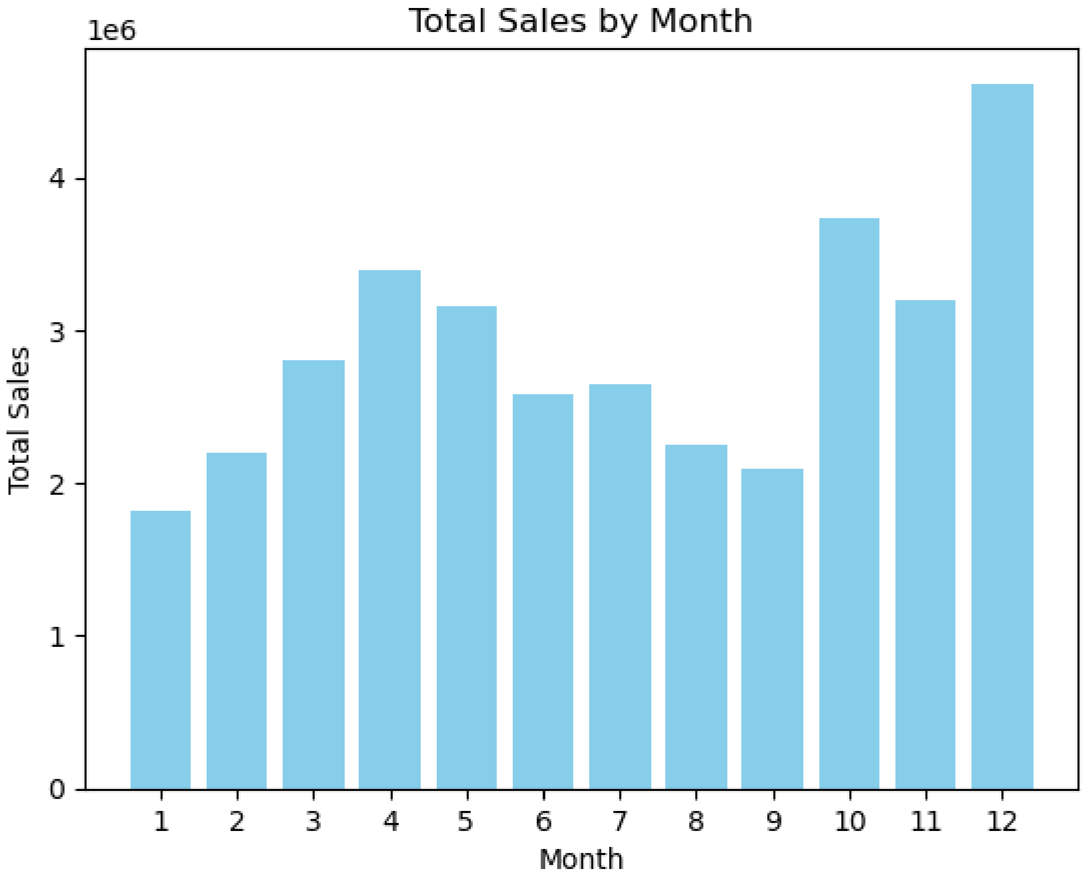


- I can identify by using isnull and heat map that there is no null values in sales data sets.

1. Total Sales by Month:

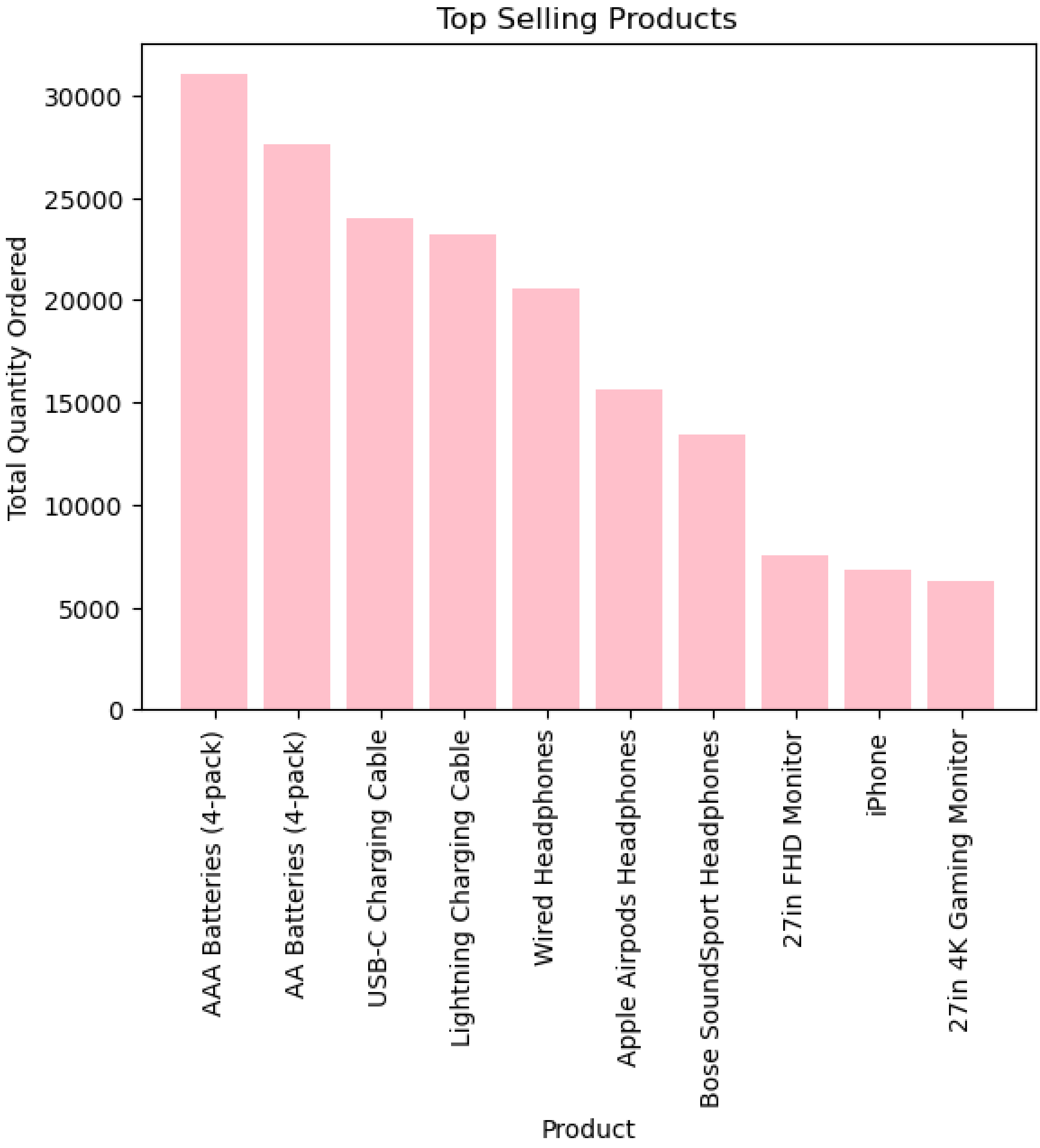
In [59]:

```
sales = data.groupby('Month')['Sales'].sum()
plt.bar(sales.index, sales.values,color='skyblue')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('Total Sales by Month')
plt.xticks(sales.index)
plt.show()
```



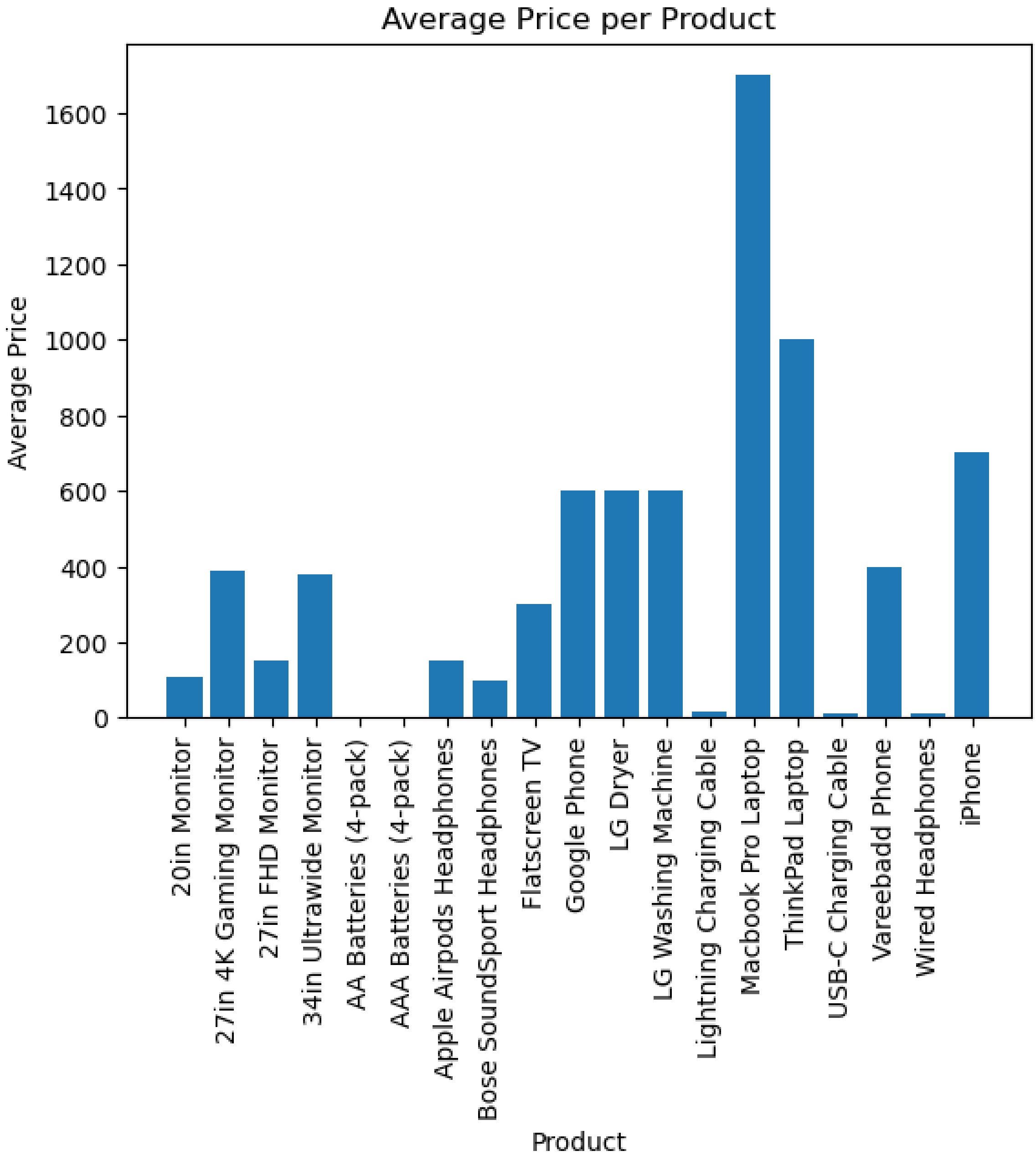
2.Top Selling Products:

In [41]:
```
top_products = data.groupby('Product')['Quantity Ordered'].sum().nlargest(10)
plt.bar(top_products.index, top_products.values,color='pink')
plt.xlabel('Product')
plt.ylabel('Total Quantity Ordered')
plt.title('Top Selling Products')
plt.xticks(rotation=90)
plt.show()
```



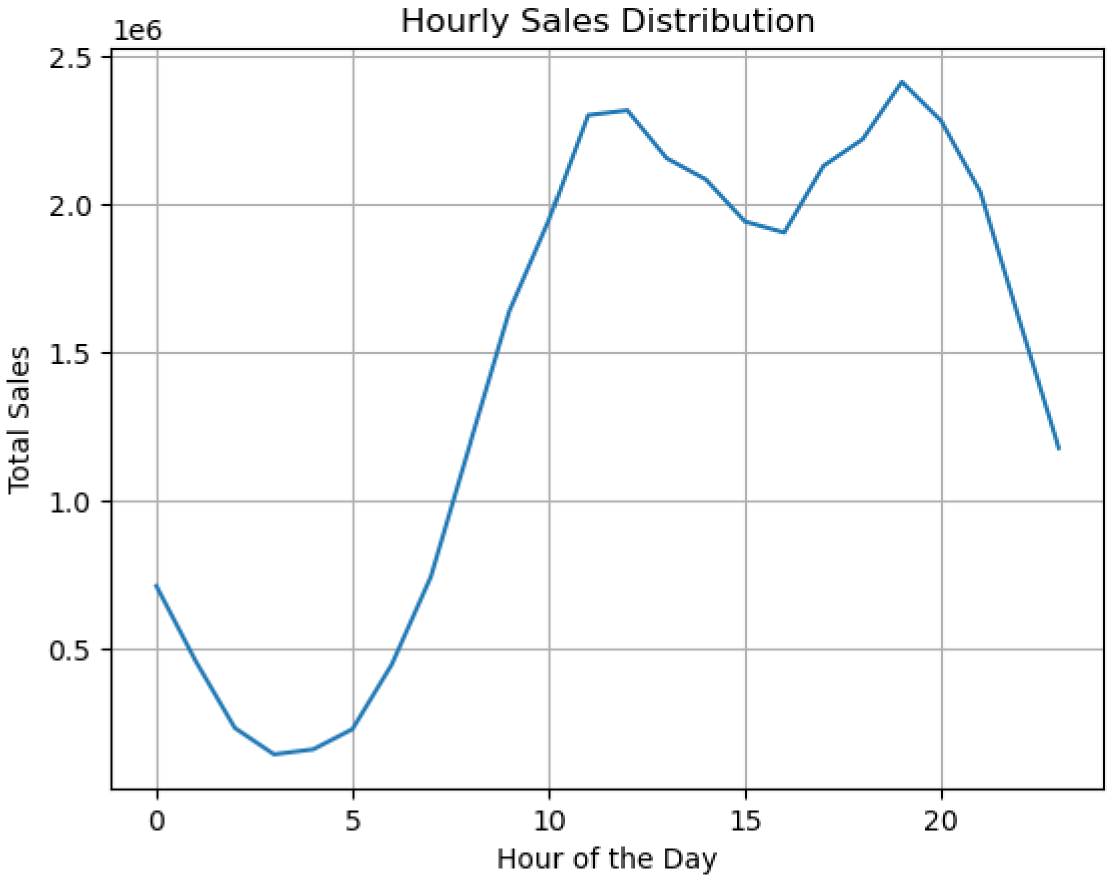
3. Average Price per Product:

In [45]:
```
avg_price_per_product = data.groupby('Product')['Price Each'].mean()
plt.bar(avg_price_per_product.index, avg_price_per_product.values)
plt.xlabel('Product')
plt.ylabel('Average Price')
plt.title('Average Price per Product')
plt.xticks(rotation=90)
plt.show()
```



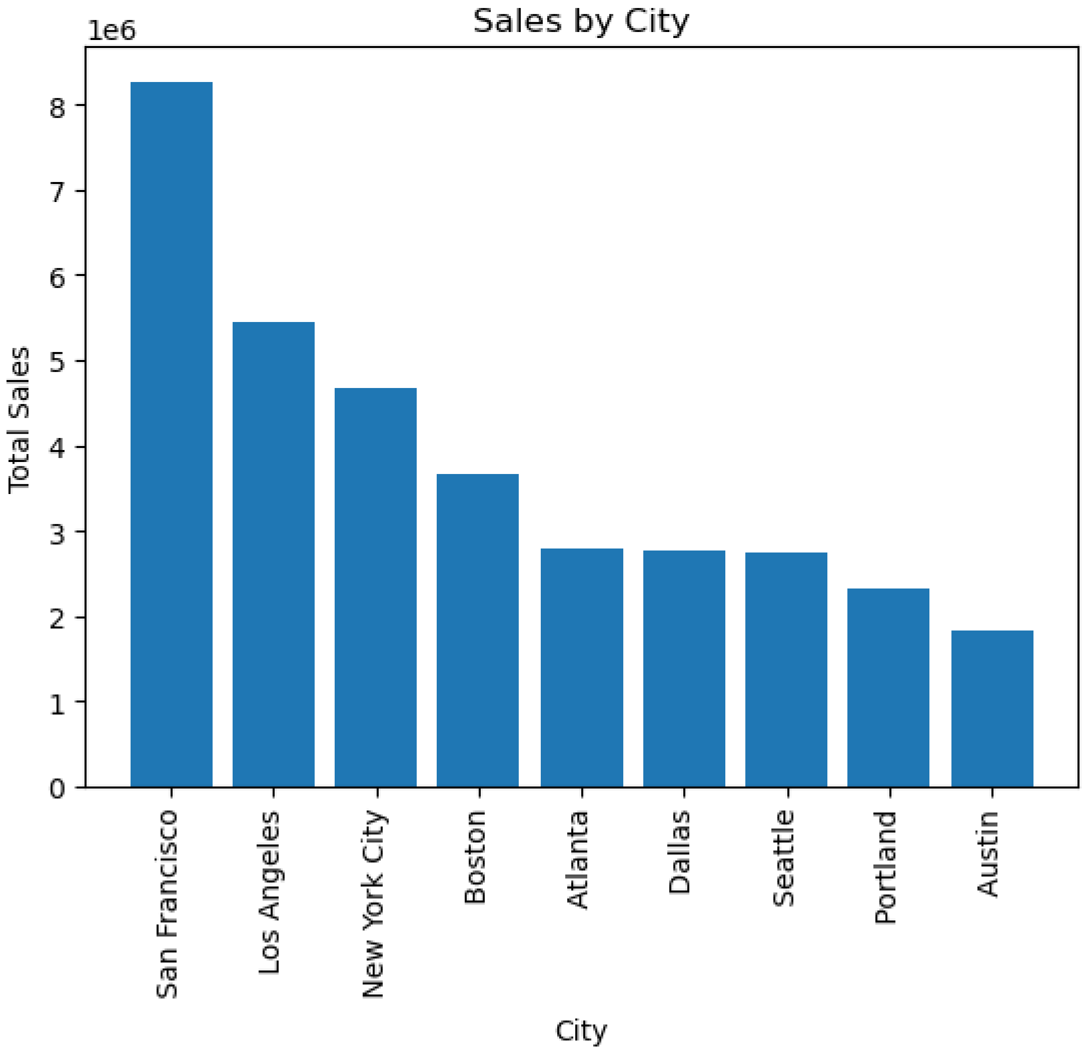
4. Hourly Sales Distribution:

In [46]:
```
hourly_sales = data.groupby('Hour')['Sales'].sum()
sns.lineplot(x=hourly_sales.index, y=hourly_sales.values)
plt.xlabel('Hour of the Day')
plt.ylabel('Total Sales')
plt.title('Hourly Sales Distribution')
plt.grid()
plt.show()
```



5. Sales by City:

In [47]:
```
city_sales = data.groupby('City')['Sales'].sum().sort_values(ascending=False)
plt.bar(city_sales.index, city_sales.values)
plt.xlabel('City')
plt.ylabel('Total Sales')
plt.title('Sales by City')
plt.xticks(rotation=90)
plt.show()
```

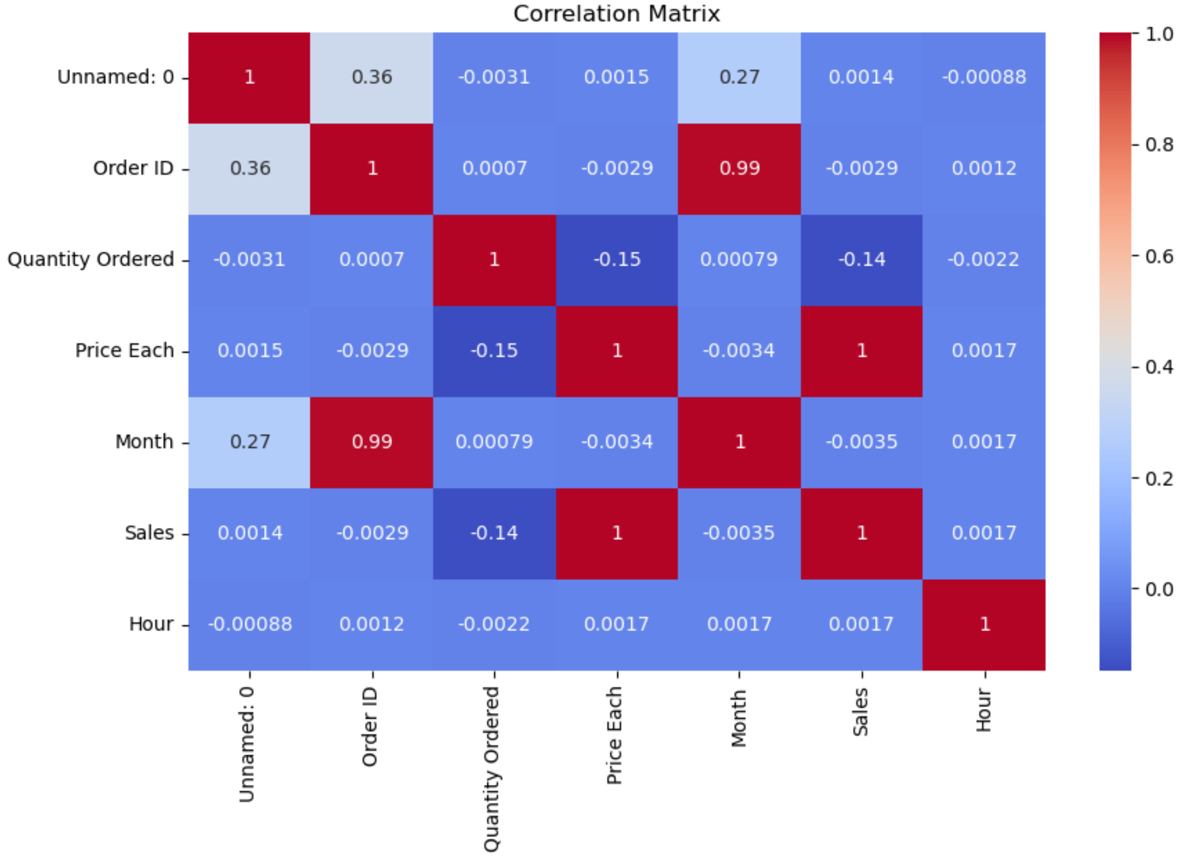


6.Correlation Analysis:

- To check for correlations between variables, I can create a correlation matrix and visualize it using a heatmap.

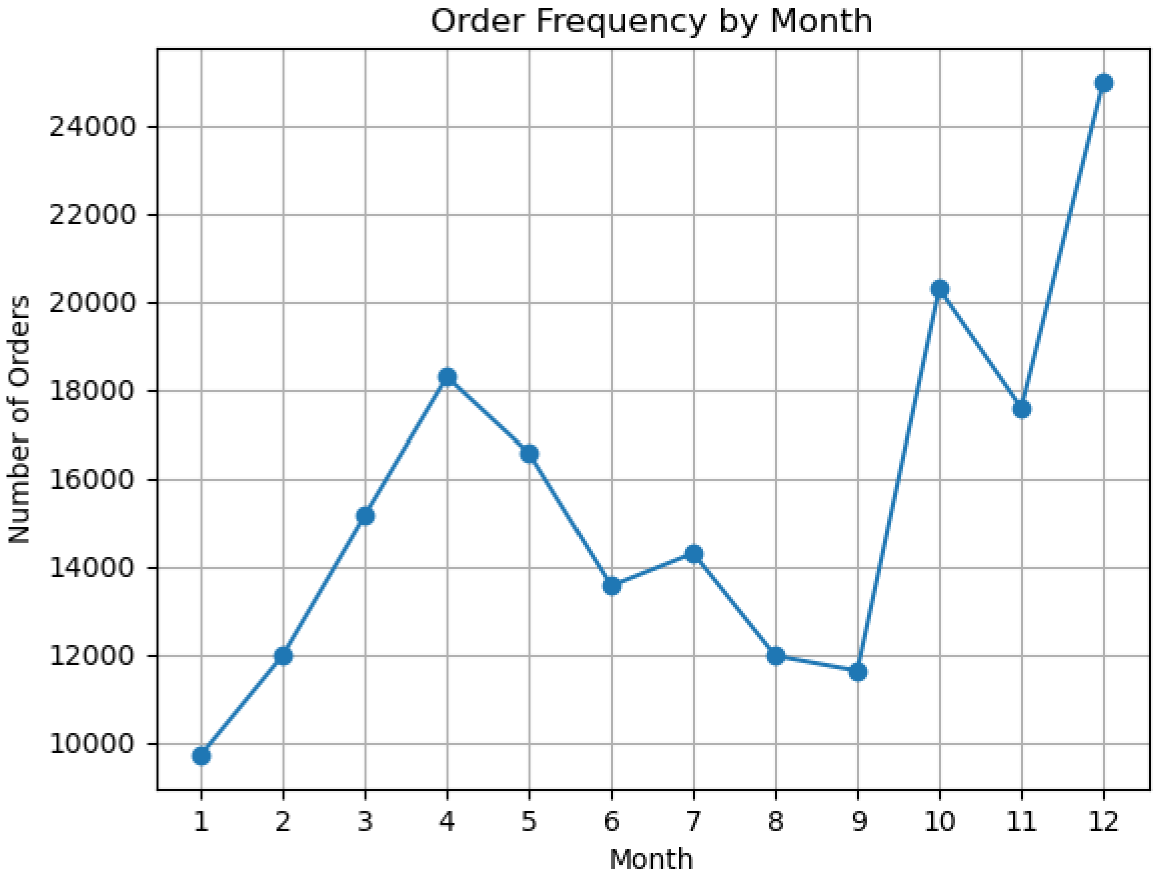
In [52]:
```
correlation_matrix = data.corr(numeric_only=True)
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
```

Out[52]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```



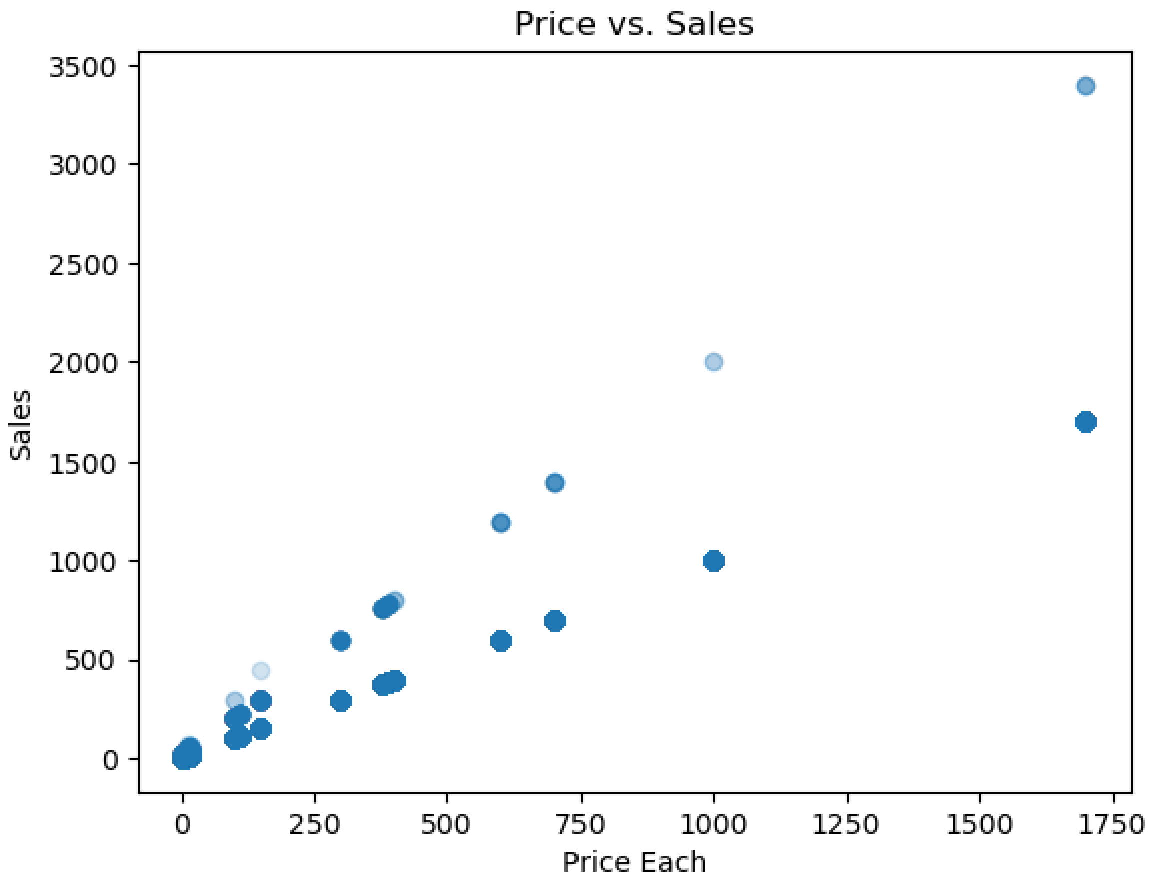
7. Order Frequency by Month:

In [54]:
```
order_frequency = data['Month'].value_counts().sort_index()
plt.plot(order_frequency.index, order_frequency.values, marker='o')
plt.xlabel('Month')
plt.ylabel('Number of Orders')
plt.title('Order Frequency by Month')
plt.xticks(order_frequency.index)
plt.grid()
plt.show()
```



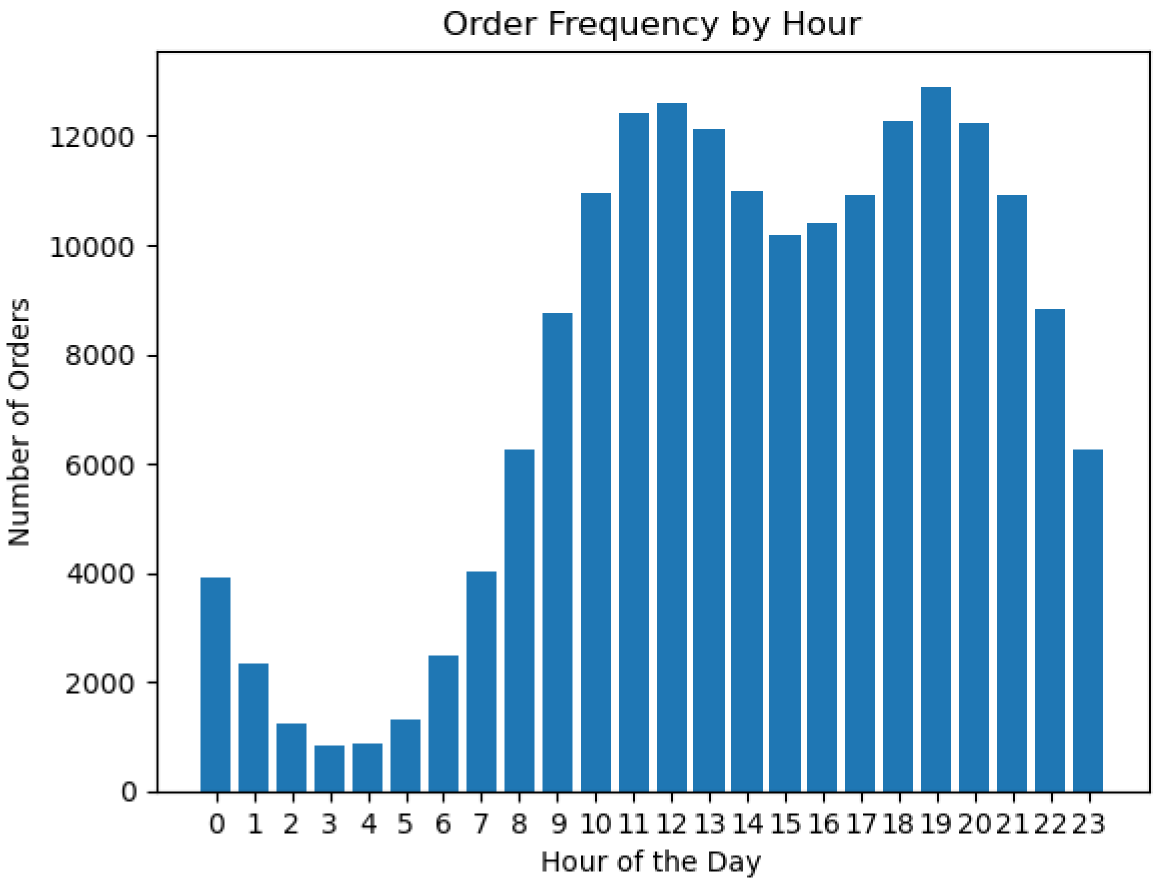
8.Price vs. Sales Scatter Plot:

In [55]:
```
plt.scatter(data['Price Each'], data['Sales'], alpha=0.2)
plt.xlabel('Price Each')
plt.ylabel('Sales')
plt.title('Price vs. Sales')
```



9.Order Frequency by Hour:

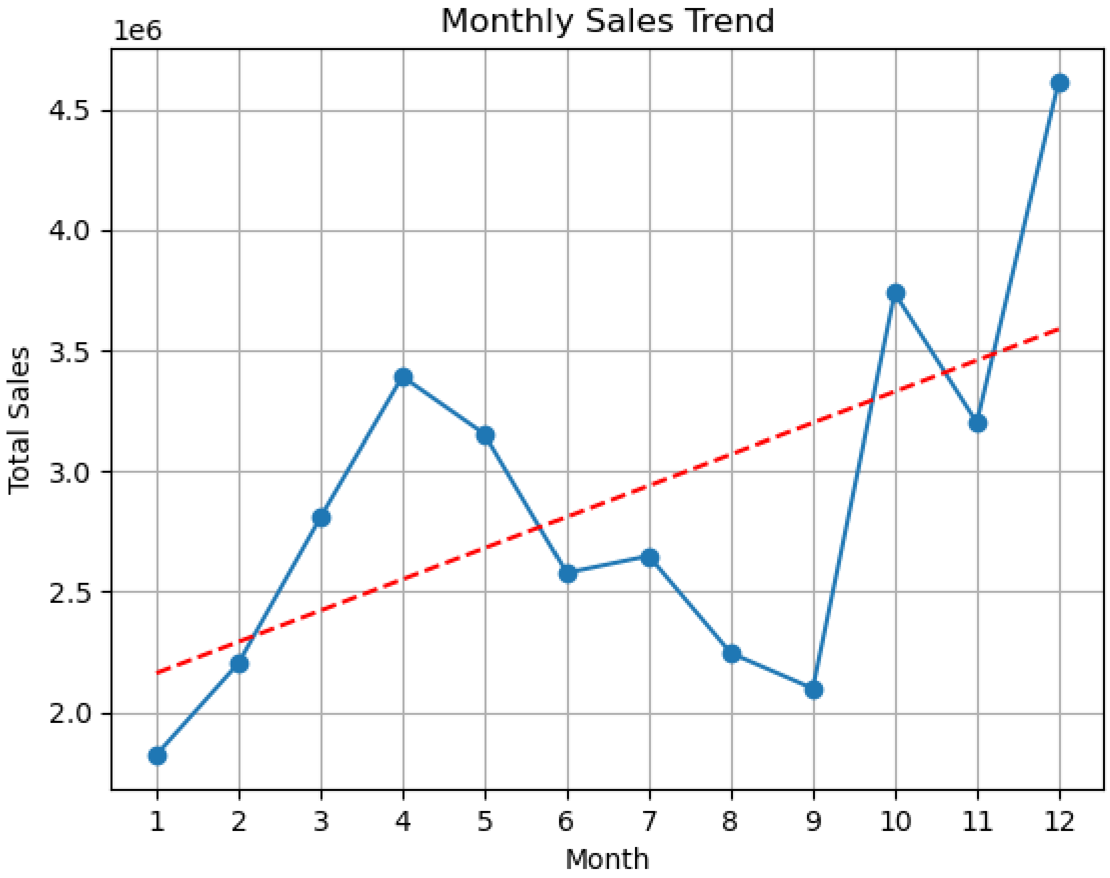
In [58]:
```
hourly_orders = data['Hour'].value_counts().sort_index()
plt.bar(hourly_orders.index, hourly_orders.values)
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Orders')
plt.title('Order Frequency by Hour')
plt.xticks(hourly_orders.index)
plt.show()
```



10. Monthly Sales Trend with a Trendline:

In [61]:

```
monthly_sales = data.groupby('Month')['Sales'].sum()
plt.plot(monthly_sales.index, monthly_sales.values, marker='o')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('Monthly Sales Trend')
plt.xticks(monthly_sales.index)
plt.grid()
# Add a trendline
z = np.polyfit(monthly_sales.index, monthly_sales.values, 1)
p = np.poly1d(z)
plt.plot(monthly_sales.index, p(monthly_sales.index), "r--")
plt.show()
```


Monthly Sales Trend
1e6
Total Sales
4.5
4.0
3.5
3.0
2.5
2.0
1
2
3
4
5
6
7
8
9
10
11
12
Month