Analyzing Adidas sales Data with Python

Problem Statement The objective of this assignment is to analyze the Adidas sales database and identify key insights to help improve sales performance and optimize business strategies. By examining the sales data, we aim to understand factors influencing sales, identify trends, and uncover opportunities for growth. The analysis will be conducted using Advanced Python visualizations and filters to provide an interactive and insightful dashboard.

Import Library

In [1]:

```
import pandas as pd
```

In [2]:

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

```
C:\Users\Syed Arif\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWar
ning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of Sc
iPy (detected version 1.25.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Uploading Csv fle

In [3]:

```
df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\AdidasSalesdata.csv")
```

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

In [4]:
```
df.head()
```

Out[4]:

	Retailer	Retailer ID	Invoice Date	Region	State	City	Gender Type	Product Category	Price per Unit	U
0	Foot Locker	1185732	Tuesday, October 26, 2021	Northeast	Pennsylvania	Philadelphia	Men	Apparel	55	
1	Foot Locker	1185732	Wednesday, October 27, 2021	Northeast	Pennsylvania	Philadelphia	Women	Apparel	45	
2	Foot Locker	1185732	Thursday, October 28, 2021	Northeast	Pennsylvania	Philadelphia	Men	Street Footwear	45	
3	Foot Locker	1185732	Friday, October 29, 2021	Northeast	Pennsylvania	Philadelphia	Men	Athletic Footwear	45	
4	Foot Locker	1185732	Saturday, October 30, 2021	Northeast	Pennsylvania	Philadelphia	Women	Street Footwear	35	

.tail()

tail is used to show rows by Descending order

In [5]:
```
df.tail()
```

Out[5]:

	Retailer	Retailer ID	Invoice Date	Region	State	City	Gender Type	Product Category	Price per Unit	Units Sold	To Sa
9643	West Gear	1128299	Saturday, March 14, 2020	West	Nevada	Las Vegas	Women	Apparel	56	170	9
9644	West Gear	1128299	Sunday, March 15, 2020	West	Nevada	Las Vegas	Men	Street Footwear	20	149	2
9645	West Gear	1128299	Monday, March 16, 2020	West	Nevada	Las Vegas	Men	Athletic Footwear	31	145	4
9646	West Gear	1128299	Tuesday, March 17, 2020	West	Nevada	Las Vegas	Women	Street Footwear	26	128	3
9647	West Gear	1128299	Wednesday, March 18, 2020	West	Nevada	Las Vegas	Women	Athletic Footwear	26	96	2

.shape

It show the total no of rows & Column in the dataset

In [6]:
```
df.shape
```

Out[6]:
```
(9648, 14)
```

.Columns

It show the no of each Column

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Retailer', 'Retailer ID', 'Invoice Date', 'Region', 'State', 'City',
       'Gender Type', 'Product Category', 'Price per Unit', 'Units Sold',
       'Total Sales', 'Operating Profit', 'Operating Margin', 'Sales Metho
d'],
      dtype='object')
```

.dtypes

This Attribute show the data type of each column

In [8]:
```
df.dtypes
```

Out[8]:
```
Retailer              object
Retailer ID            int64
Invoice Date          object
Region                object
State                 object
City                  object
Gender Type           object
Product Category      object
Price per Unit         int64
Units Sold             int64
Total Sales            int64
Operating Profit     float64
Operating Margin     float64
Sales Method          object
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [9]:
```
df["City"].unique()
```

Out[9]:
```
array(['Philadelphia', 'Providence', 'New York', 'Wilmington',
       'Manchester', 'Hartford', 'Charleston', 'Baltimore', 'Boston',
       'Portland', 'Burlington', 'Newark', 'Albany', 'Columbus',
       'Detroit', 'Fargo', 'Sioux Falls', 'St. Louis', 'Des Moines',
       'Indianapolis', 'Milwaukee', 'Chicago', 'Minneapolis', 'Omaha',
       'Wichita', 'Richmond', 'Atlanta', 'Orlando', 'Miami', 'Louisville',
       'Charlotte', 'Salt Lake City', 'Anchorage', 'Cheyenne',
       'Los Angeles', 'Seattle', 'Dallas', 'Knoxville', 'Birmingham',
       'Jackson', 'Billings', 'New Orleans', 'Houston', 'Oklahoma City',
       'Little Rock', 'San Francisco', 'Boise', 'Honolulu', 'Albuquerque',
       'Phoenix', 'Denver', 'Las Vegas'], dtype=object)
```

.nuique()

It will show the total no of unque value from whole data frame

In [10]:
```
df.nunique()
```

Out[10]:
```
Retailer                6
Retailer ID             4
Invoice Date          724
Region                  5
State                  50
City                   52
Gender Type             2
Product Category        3
Price per Unit         94
Units Sold            361
Total Sales          3138
Operating Profit     5618
Operating Margin       66
Sales Method            3
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [11]:
```
df.describe()
```

Out[11]:

	Retailer ID	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin
count	9.648000e+03	9648.000000	9648.000000	9648.000000	9648.000000	9648.000000
mean	1.173850e+06	45.216625	256.930037	93273.437500	34425.244761	0.422991
std	2.636038e+04	14.705397	214.252030	141916.016727	54193.113713	0.097197
min	1.128299e+06	7.000000	0.000000	0.000000	0.000000	0.100000
25%	1.185732e+06	35.000000	106.000000	4254.500000	1921.752500	0.350000
50%	1.185732e+06	45.000000	176.000000	9576.000000	4371.420000	0.410000
75%	1.185732e+06	55.000000	350.000000	150000.000000	52062.500000	0.490000
max	1.197831e+06	110.000000	1275.000000	825000.000000	390000.000000	0.800000

.value_counts

It Shows all the unique values with their count

In [12]:
```
df["City"].value_counts()
```

Out[12]:
```
Portland          360
Charleston        288
Philadelphia      216
New Orleans       216
Orlando           216
Salt Lake City    216
Los Angeles       216
Dallas            216
Knoxville         216
Birmingham        216
Jackson           216
Houston           216
Richmond          216
Oklahoma City     216
Little Rock       216
San Francisco     216
Boise             216
Albuquerque       216
Phoenix           216
Providence        216
Atlanta           216
Las Vegas         216
New York          216
Manchester        216
Hartford          216
Boston            216
Burlington        216
Detroit           144
Denver            144
Wilmington        144
Honolulu          144
Baltimore         144
Newark            144
Billings          144
Albany            144
Columbus          144
Chicago           144
Minneapolis       144
Seattle           144
Sioux Falls       144
Cheyenne          144
Anchorage         144
St. Louis         144
Charlotte         144
Louisville        144
Miami             144
Des Moines        144
Indianapolis      144
Milwaukee         144
Wichita           144
Omaha             144
Fargo             144
Name: City, dtype: int64
```

.isnull()

It shows the how many null values

In [13]:

```
df.isnull()
```

Out[13]:

	Retailer	Retailer ID	Invoice Date	Region	State	City	Gender Type	Product Category	Price per Unit	Units Sold	Total Sales	Op
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
9643	False	False	False	False	False	False	False	False	False	False	False	
9644	False	False	False	False	False	False	False	False	False	False	False	
9645	False	False	False	False	False	False	False	False	False	False	False	
9646	False	False	False	False	False	False	False	False	False	False	False	
9647	False	False	False	False	False	False	False	False	False	False	False	

9648 rows × 14 columns

1. Calculate and Visualize Overall Sales, Profit, Average Price per Unit, and Total Units Sold

In [14]:

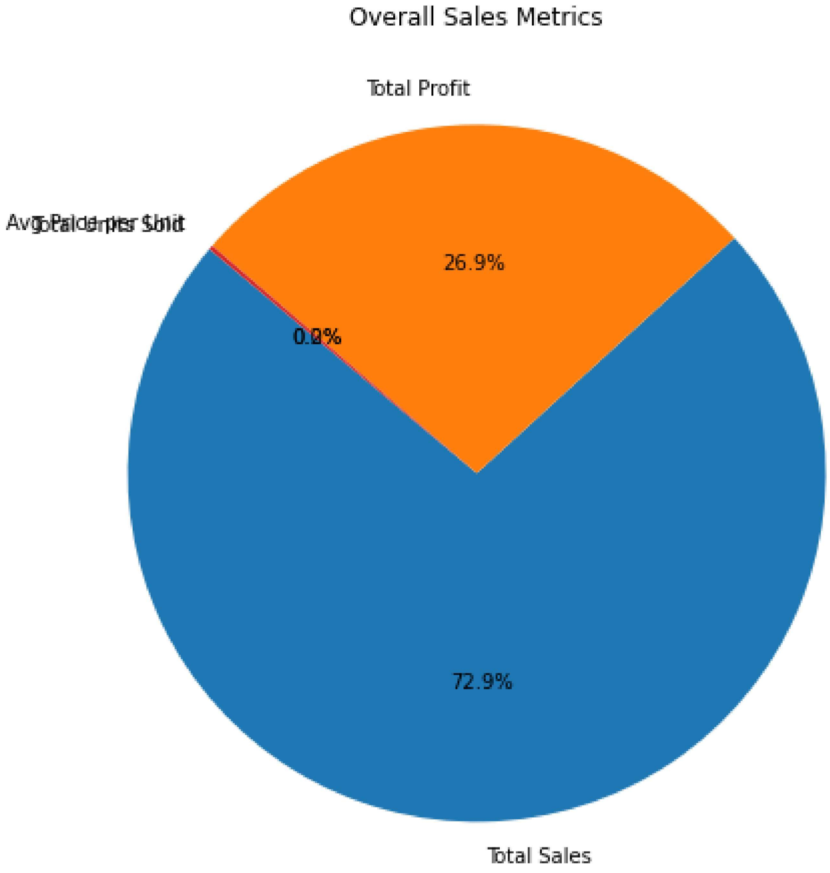
```
total_sale = df["Total Sales"].sum()
total_profit = df["Operating Profit"].sum()
avg_price  = df["Price per Unit"].mean()
total_unit_sold = df["Units Sold"].sum()
```

In [15]:

```
print ("Total Sale of Adidas" ,total_sale )
print ("Total Profit of Adidas" , total_profit )
print ("Average price of Adidas" , avg_price )
print ("Total unit sold of Adidas" , total_unit_sold )
```

```
Total Sale of Adidas 899902125
Total Profit of Adidas 332134761.45000005
Average price of Adidas 45.21662520729685
Total unit sold of Adidas 2478861
```

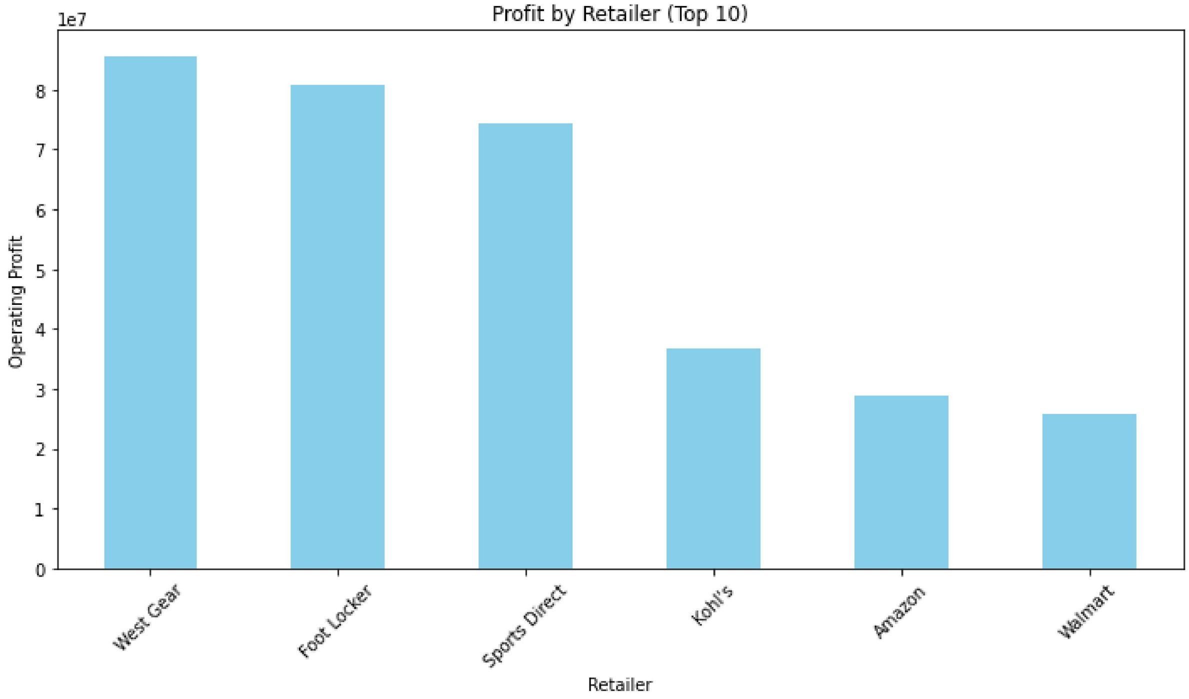
```
In [16]: # Create a pie chart to visualize overall sales composition
         labels = ['Total Sales', 'Total Profit', 'Avg Price per Unit', 'Total Units So
         values = [total_sale, total_profit, avg_price, total_unit_sold]
         plt.figure(figsize=(8, 8))
         plt.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)
         plt.title('Overall Sales Metrics')
         plt.show()
```



2. Profit by Retailer

```
In [17]: profit_by_retailer = df.groupby('Retailer')['Operating Profit'].sum().sort_valu
         top_retailers = profit_by_retailer.head(10)
```

In [18]:
```
# Create a bar chart to visualize profit by retailer
plt.figure(figsize=(10, 6))
top_retailers.plot(kind='bar', color='skyblue')
plt.title('Profit by Retailer (Top 10)')
plt.xlabel('Retailer')
plt.ylabel('Operating Profit')
plt.xticks(rotation=45)
plt.tight_layout()
```

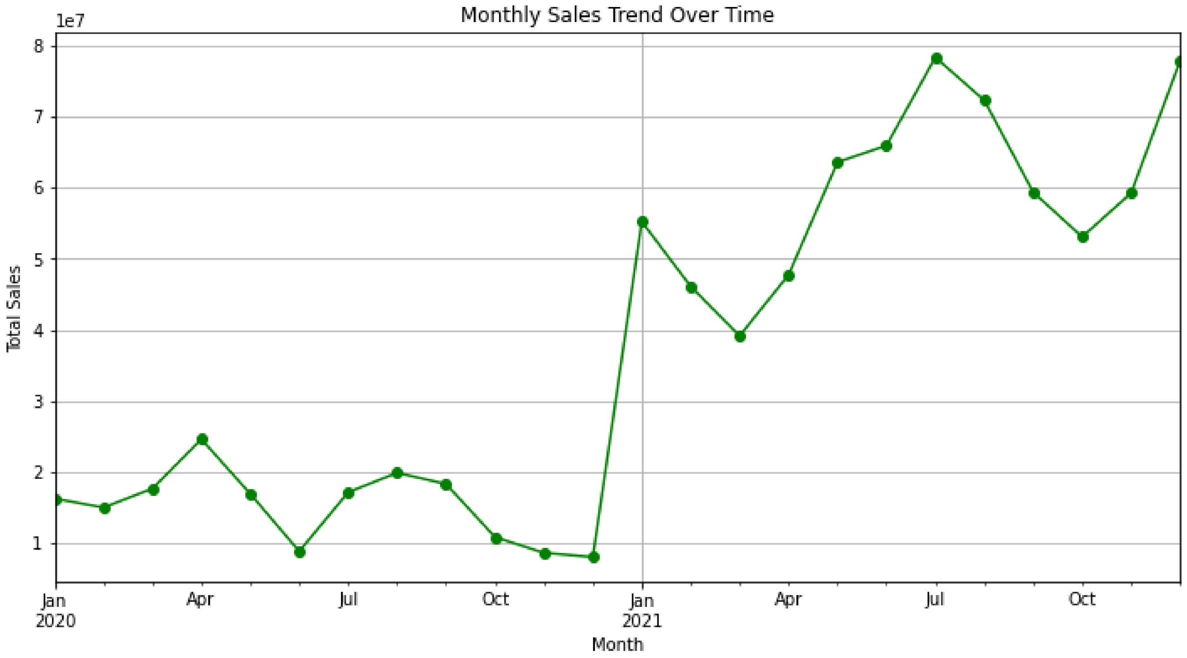


3. Sales Trend Over Time (Monthly)

In [19]:
```
df['Invoice Date'] = pd.to_datetime(df['Invoice Date'])
monthly_sales = df.resample('M', on='Invoice Date')['Total Sales'].sum()
```

Create a line chart to visualize sales trend over time

In [20]:
```
plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='line', marker='o', color='green')
plt.title('Monthly Sales Trend Over Time')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid()
```

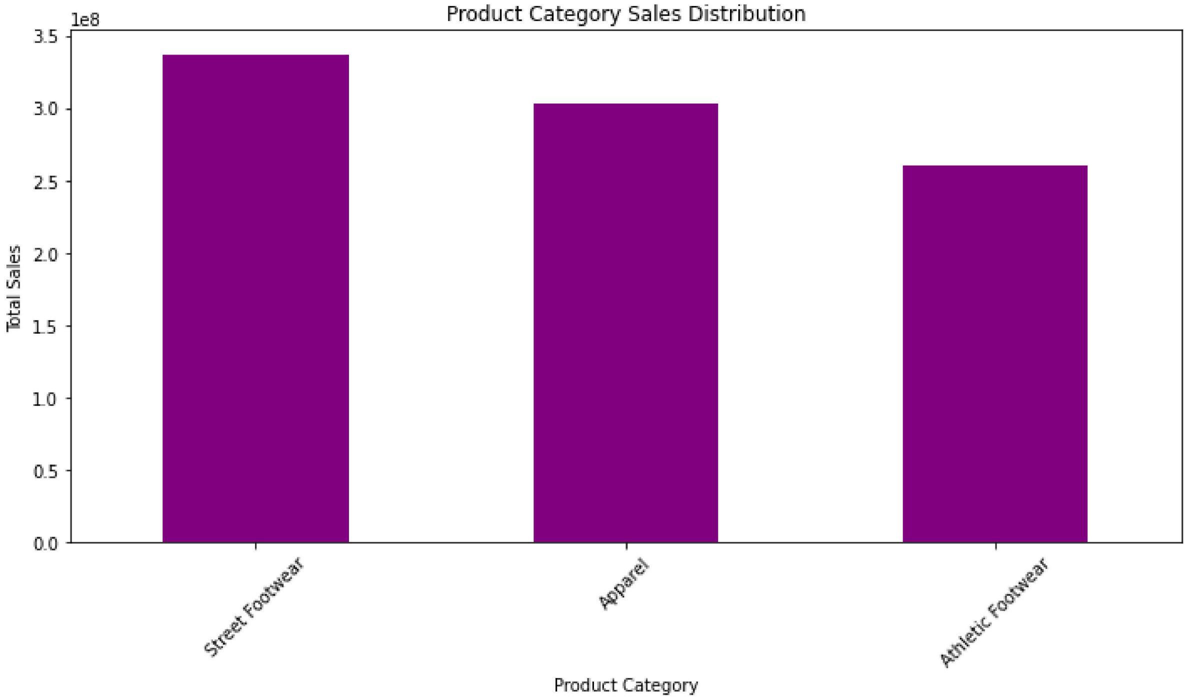


4. Product Category Sales Distribution

In [21]:
```
category_sales = df.groupby('Product Category')['Total Sales'].sum().sort_valu
```

Create a bar chart to visualize product category sales distribution

In [22]:
```
plt.figure(figsize=(10, 6))
category_sales.plot(kind='bar', color='purple')
plt.title('Product Category Sales Distribution')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
```



5. Units Sold by Product Category and Gender Type

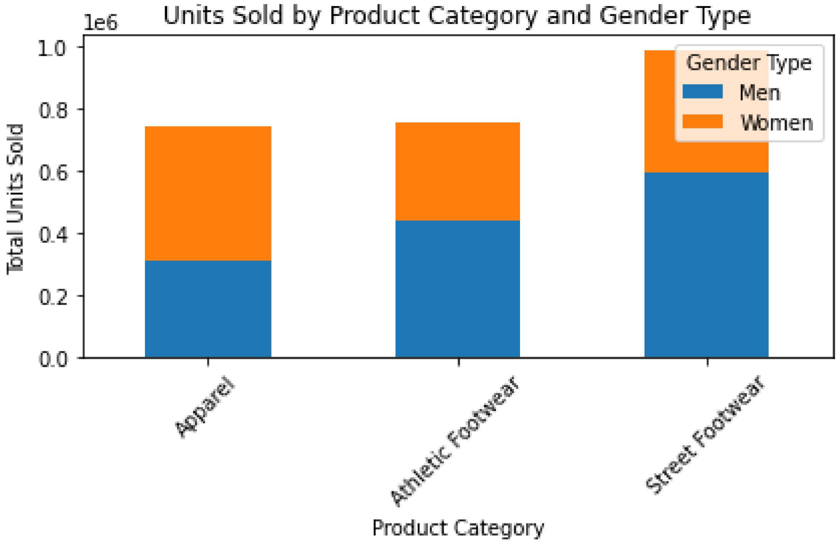
In [23]:
```
units_sold_by_category_gender = df.groupby(['Product Category', 'Gender Type']
```

Create a stacked bar chart to visualize units sold by product category and gender type

In [24]:

```
plt.figure(figsize=(12, 6))
units_sold_by_category_gender.plot(kind='bar', stacked=True)
plt.title('Units Sold by Product Category and Gender Type')
plt.xlabel('Product Category')
plt.ylabel('Total Units Sold')
plt.xticks(rotation=45)
plt.tight_layout()
```

```
<Figure size 864x432 with 0 Axes>
```



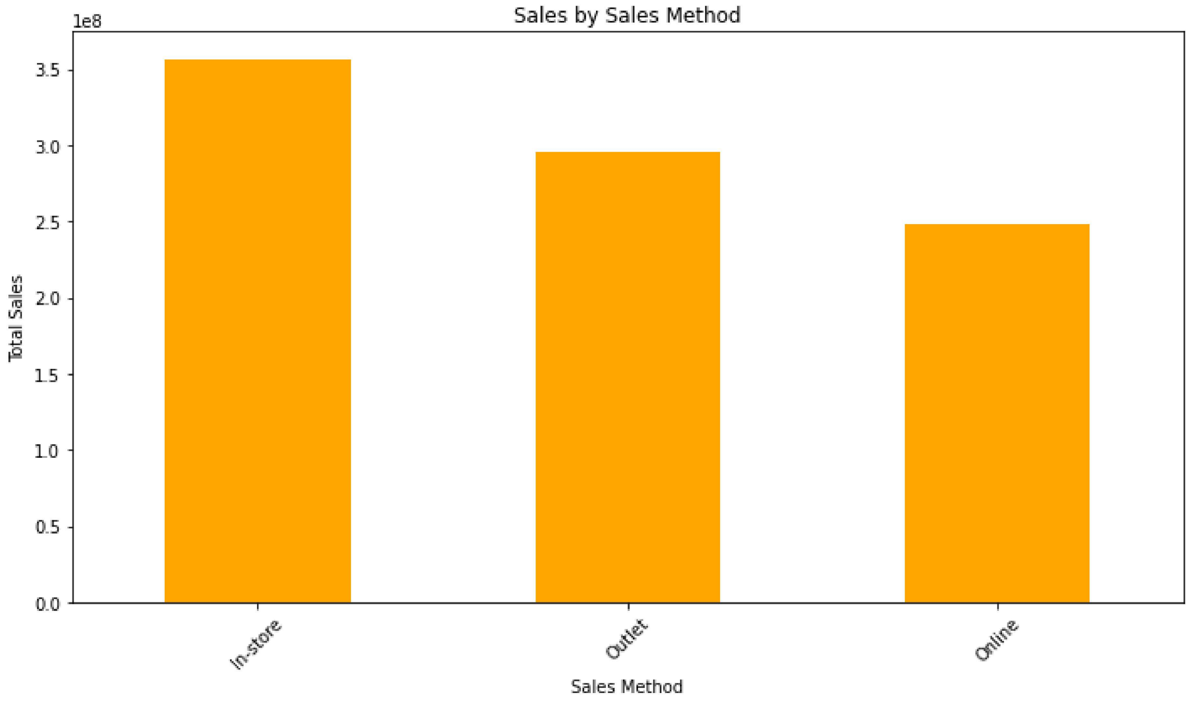
6. Effective Sales Methods

In [25]:

```
sales_by_method = df.groupby('Sales Method')['Total Sales'].sum().sort_values(
```

In [26]:

```
# Create a bar chart to visualize sales by method
plt.figure(figsize=(10, 6))
sales_by_method.plot(kind='bar', color='orange')
plt.title('Sales by Sales Method')
plt.xlabel('Sales Method')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
```



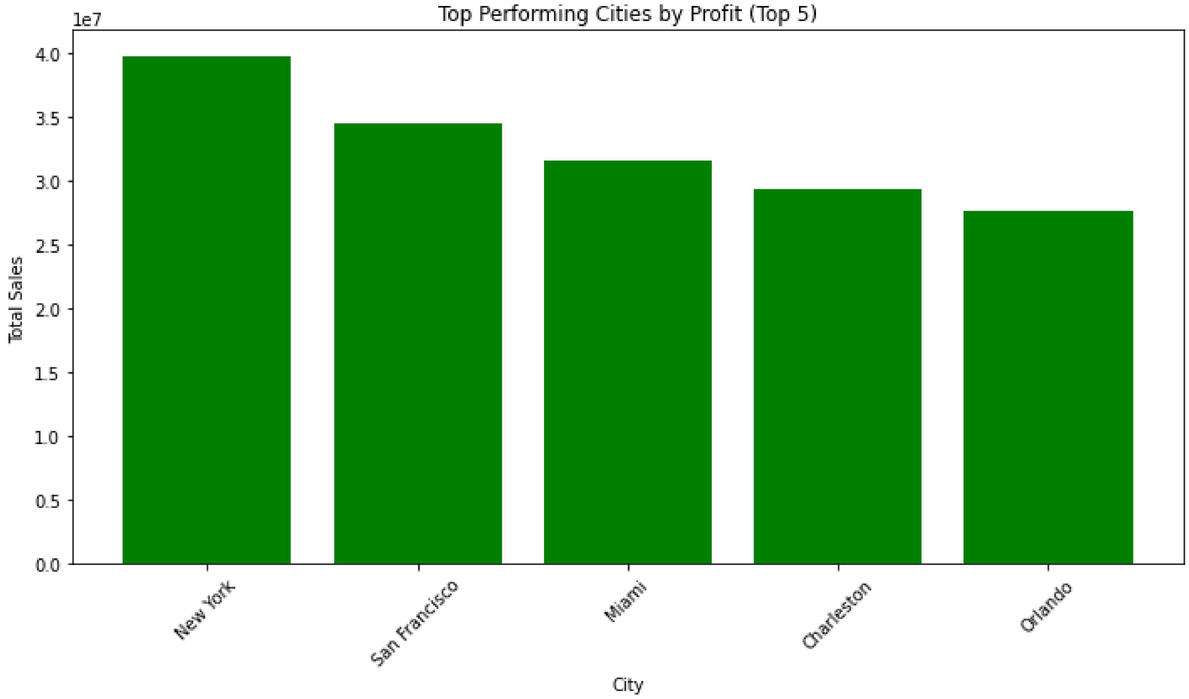
7. Regional Sales Analysis

In [27]:

```
region_sales = df.groupby(['Region', 'State', 'City'])['Total Sales'].sum().re
top_cities_by_profit = region_sales.sort_values('Total Sales', ascending=False
```

In [28]:

```
# Create a bar chart to visualize top performing cities by profit
plt.figure(figsize=(10, 6))
plt.bar(top_cities_by_profit['City'], top_cities_by_profit['Total Sales'], col
plt.title('Top Performing Cities by Profit (Top 5)')
plt.xlabel('City')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
```



9. Sales Trend Over Time (Yearly)

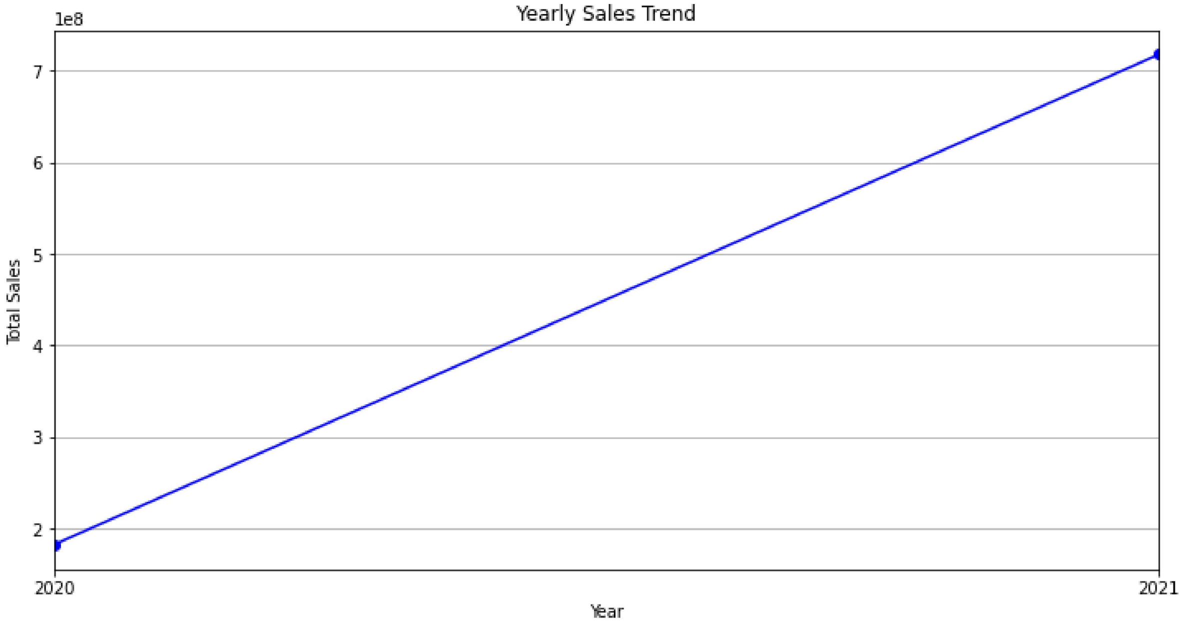
In [29]:

```
yearly_sales = df.resample('Y', on='Invoice Date')['Total Sales'].sum()
```

In [30]:

```
# Create a line chart to visualize yearly sales trend
plt.figure(figsize=(12, 6))
yearly_sales.plot(kind='line', marker='o', color='blue')
plt.title('Yearly Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.grid()

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


Yearly Sales Trend
1e8
Total Sales
Year