Supermart Grocery Sales - Retail Analytics Dataset

This project is designed for data analysts to practice exploratory data analysis and visualization using Python. It involves a dataset containing grocery sales information from customers in Tamil Nadu, India. The key steps to complete this project are outlined below in an easy-to-follow format:

✅ Project Summary

This project is designed to help beginners and intermediate learners practice **data analysis** and **machine learning** using a grocery store dataset. It shows how you can explore sales data, clean it, and build a model to predict sales.

✅ Tools Used

- **Python** for coding.
- Libraries like **pandas**, **numpy**, **matplotlib**, **seaborn**, and **sklearn**.
- You can also use **Excel** or **SQL** to explore the data.

✅ Dataset Description

- It contains data about orders placed in a grocery delivery app.
- The data includes:
    - Order ID
    - Customer name
    - Product category and sub-category
    - City and region
    - Order date
    - Sales, discount, and profit amounts
- The dataset is fictional but reflects realistic scenarios from Tamil Nadu, India.

1. **Import Libraries**:
   Use libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn for data handling, visualization, and modeling.

2. **Load the Dataset**:
   Load the CSV file into a DataFrame and inspect its structure.

3. **Data Preprocessing**:
   - Handle missing values and duplicates.
   - Convert date columns to datetime format and extract day, month, and year.
   - Encode categorical variables using Label Encoding.

4. **Exploratory Data Analysis (EDA)**:
   - Visualize sales distribution by category.
   - Plot sales trends over time.
   - Analyze feature correlations with a heatmap.

5. **Feature Engineering & Model Building**:
   - Select relevant features to predict sales.
   - Split the data into training and test sets.
   - Apply scaling techniques like StandardScaler.

6. **Train a Machine Learning Model**:
   - Use Linear Regression to predict sales.
   - Train the model and make predictions.

7. **Evaluate the Model**:
   - Calculate Mean Squared Error (MSE) and R-squared (R^2) to assess performance.

8. **Visualize Results**:
   - Plot actual vs predicted sales to see how well the model performs.

9. **Conclude and Explore Further**:
   - The model offers a good fit.
   - Experiment with advanced models like Random Forest or XGBoost.
   - Enhance features and deploy the model in dashboards.

This project provides a practical and beginner-friendly experience in data analysis and machine learning.

Conclusion

- The model works well but can be improved.
- The final goal is to use the model for real-time sales predictions.

In [1]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (2.3.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (2.3.1)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (4.58.5)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.3.1)
Requirement already satisfied: packaging>=20.0 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\kkjeg\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [7]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\kkjeg\appdata\local\progr
ams\python\python313\lib\site-packages (1.7.2)
Requirement already satisfied: numpy>=1.22.0 in c:\users\kkjeg\appdata\local\prog
rams\python\python313\lib\site-packages (from scikit-learn) (2.3.1)
Requirement already satisfied: scipy>=1.8.0 in c:\users\kkjeg\appdata\local\progr
ams\python\python313\lib\site-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in c:\users\kkjeg\appdata\local\prog
rams\python\python313\lib\site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\kkjeg\appdata\loc
al\programs\python\python313\lib\site-packages (from scikit-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [8]:
```
# Cell 2: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


%matplotlib inline
sns.set(style="whitegrid")
```

In [10]:
```
# Cell 3: Load the Dataset
df=pd.read_csv('Supermart Grocery Sales - Retail Analytics Dataset.csv')
df.head()
```

Out[10]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.12
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.18
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.21
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.25
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.26

In [26]:
```
# Cell 4: Basic Data Exploration
df.info()
```

```
df.describe(include='all')
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4042 entries, 0 to 9991
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Order ID       4042 non-null   object
 1   Customer Name  4042 non-null   object
 2   Category       4042 non-null   object
 3   Sub Category   4042 non-null   object
 4   City           4042 non-null   object
 5   Order Date     4042 non-null   datetime64[ns]
 6   Region         4042 non-null   object
 7   Sales          4042 non-null   int64
 8   Discount       4042 non-null   float64
 9   Profit         4042 non-null   float64
 10  State          4042 non-null   object
dtypes: datetime64[ns](1), float64(2), int64(1), object(7)
memory usage: 378.9+ KB
```

Out[26]:
```
Order ID         0
Customer Name    0
Category         0
Sub Category     0
City             0
Order Date       0
Region           0
Sales            0
Discount         0
Profit           0
State            0
dtype: int64
```

In [27]:
```
# Cell 5: Drop Missing Values and Duplicates (Data Cleaning)
df.dropna(inplace=True)

# Drop duplicate entries
df.drop_duplicates(inplace=True)

print("Data shape after cleaning:", df.shape)
```

```
Data shape after cleaning: (4042, 11)
```

In [28]:
```
# Cell 6: Parse 'Order Date' to datetime and Extract Date Features

df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')

# Extract day, month year
df['Order Day'] = df['Order Date'].dt.day
df['Order Month'] = df['Order Date'].dt.month
df['Order Year'] = df['Order Date'].dt.year

df['Month'] = df['Order Date'].dt.strftime('%B')

df.head()
```

Out[28]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
0	OD1	Harish	Oil & Masala	Masalas	Vellore	2017-11-08	North	1254	0.12
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	2017-11-08	South	749	0.18
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	2017-06-12	West	2360	0.21
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	2016-10-11	South	896	0.25
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	2016-10-11	South	2355	0.26

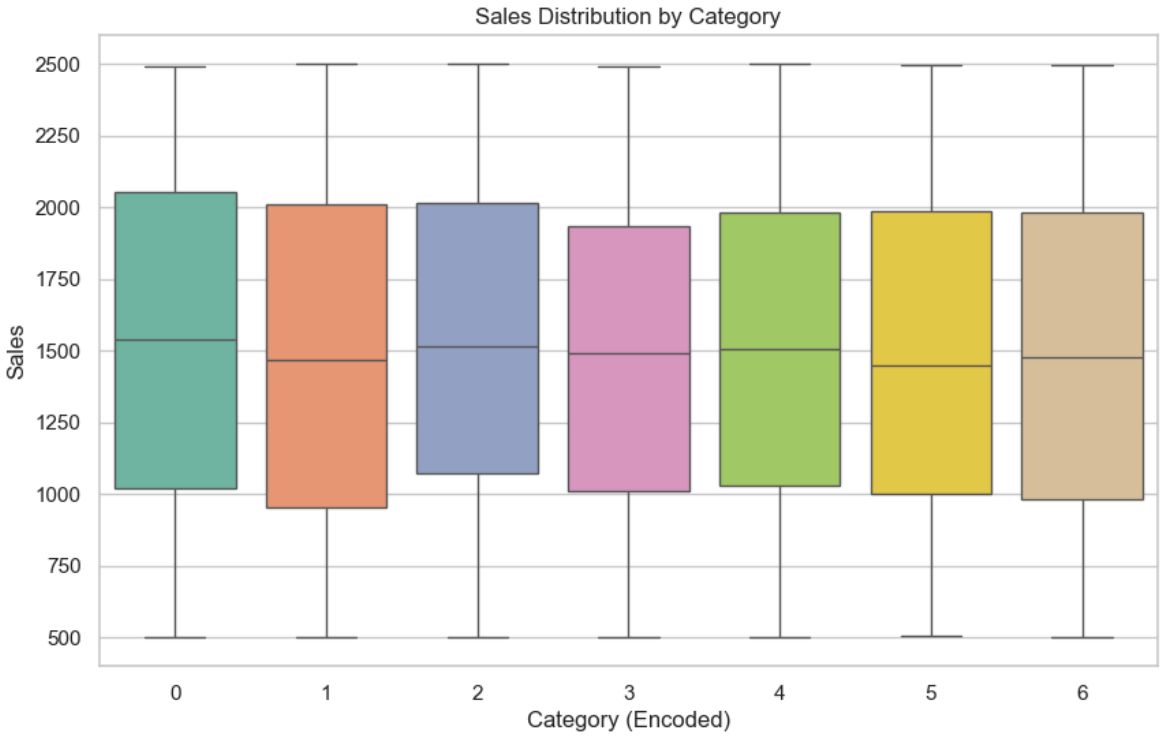
In [29]:
```
# Cell 7: Encode Categorical Variables
le = LabelEncoder()
for col in ['Category', 'Sub Category', 'City', 'Region', 'State', 'Month']:
    df[col] = le.fit_transform(df[col])

df.head()
```

Out[29]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit
0	OD1	Harish	5	14	21	2017-11-08	2	1254	0.12	401.28
1	OD2	Sudha	1	13	8	2017-11-08	3	749	0.18	149.80
2	OD3	Hussain	3	0	13	2017-06-12	4	2360	0.21	165.20
3	OD4	Jackson	4	12	4	2016-10-11	3	896	0.25	89.60
4	OD5	Ridhesh	3	18	12	2016-10-11	3	2355	0.26	918.45

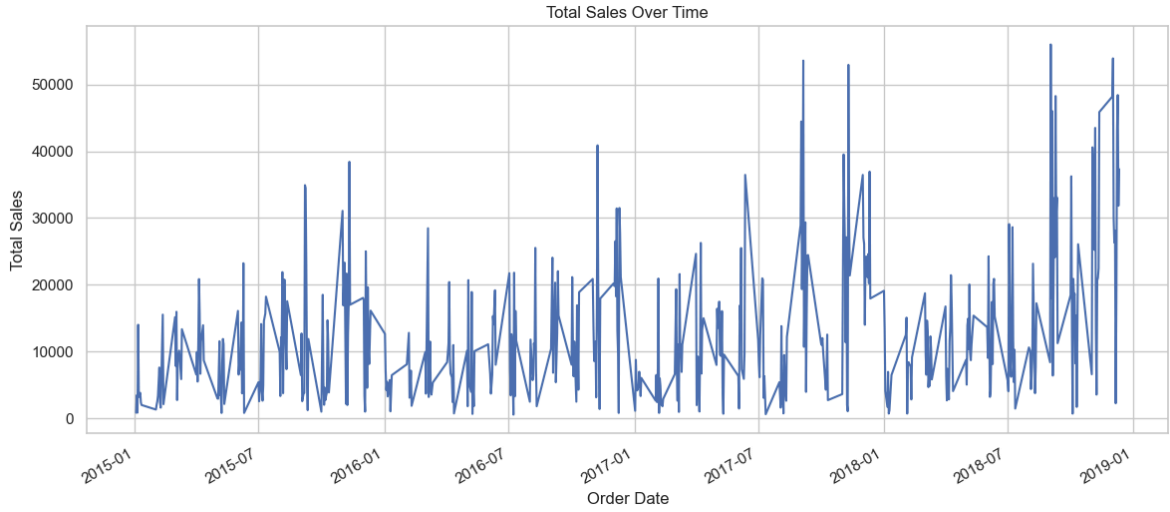
In [31]:
```
# Cell 8: Exploratory Data Analysis - Sales by Category

plt.figure(figsize=(10, 6))
sns.boxplot(x='Category', y='Sales', data=df, palette='Set2')
plt.title('Sales Distribution by Category')
plt.xlabel('Category (Encoded)')
plt.ylabel('Sales')
plt.show()
# For reference, print encoding
print("Category Encodings:", dict(enumerate(le.classes_)))
```

```
C:\Users\kkjeg\AppData\Local\Temp\ipykernel_2064\3807852725.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(x='Category', y='Sales', data=df, palette='Set2')
```

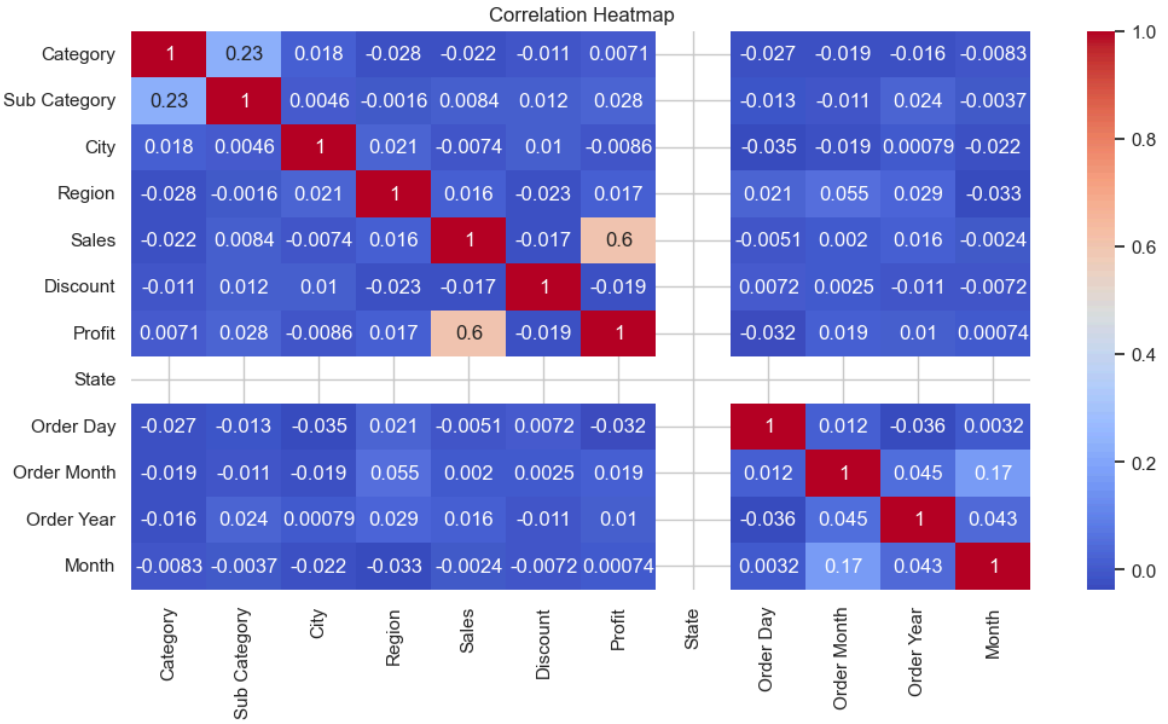



```
Category Encodings: {0: 'April', 1: 'August', 2: 'December', 3: 'February', 4: 'J
anuary', 5: 'July', 6: 'June', 7: 'March', 8: 'May', 9: 'November', 10: 'Octobe
r', 11: 'September'}
```

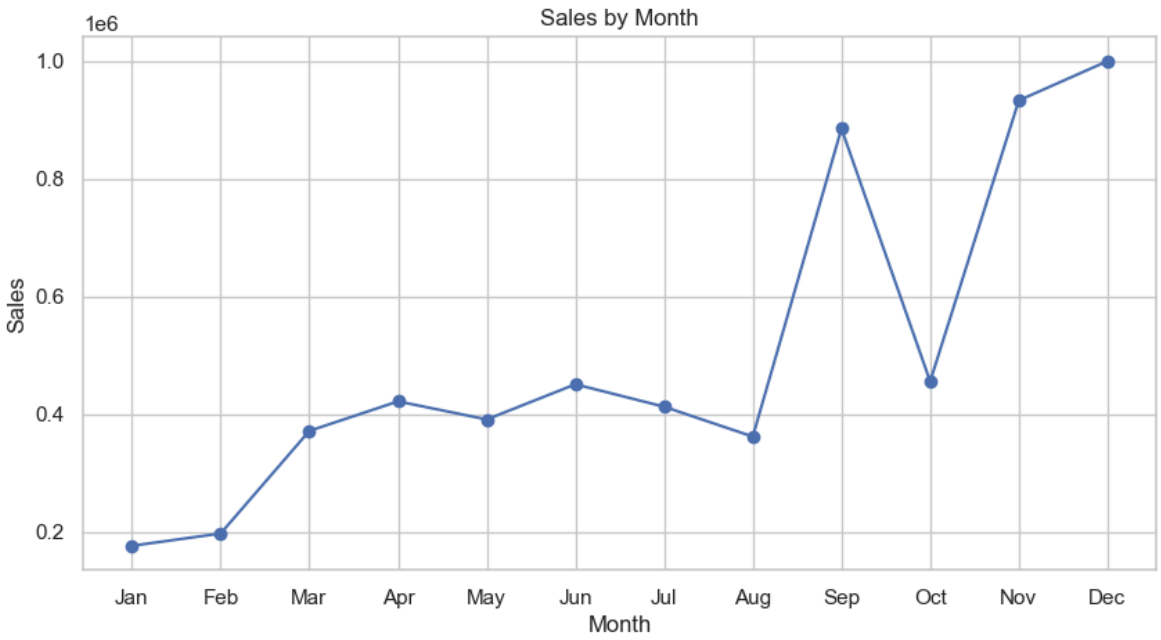
In [32]:
```
#Cell 9: Sales Trends Over Time

plt.figure(figsize=(14, 6))
df.groupby('Order Date')['Sales'].sum().plot()
plt.title('Total Sales Over Time')
plt.xlabel('Order Date')
plt.ylabel('Total Sales')
plt.show()
```



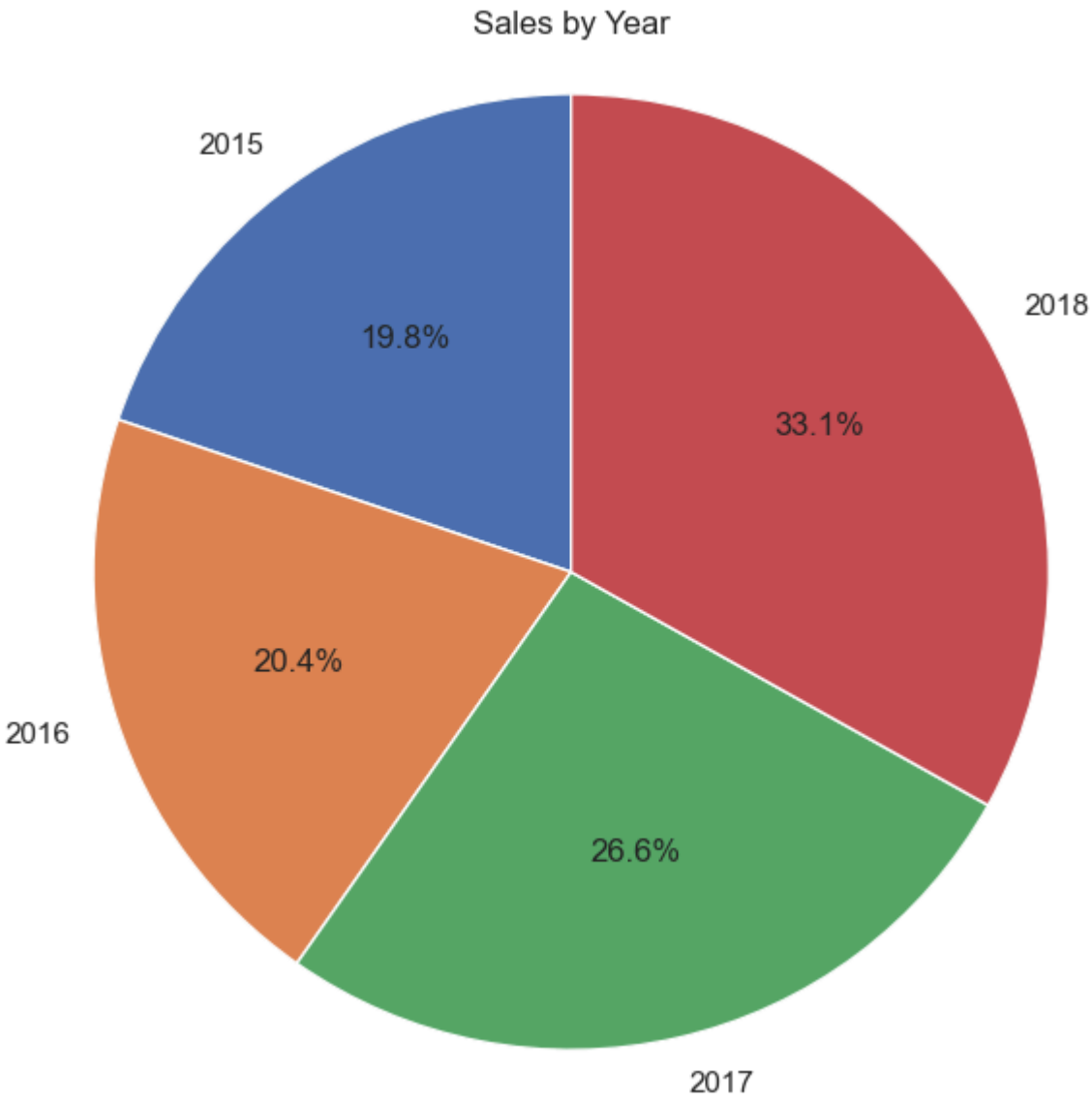
In [33]:
```
# Cell 10: Correlation Analysis

numeric_df = df.select_dtypes(include=[np.number])

plt.figure(figsize=(12, 6))
corr_matrix = numeric_df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



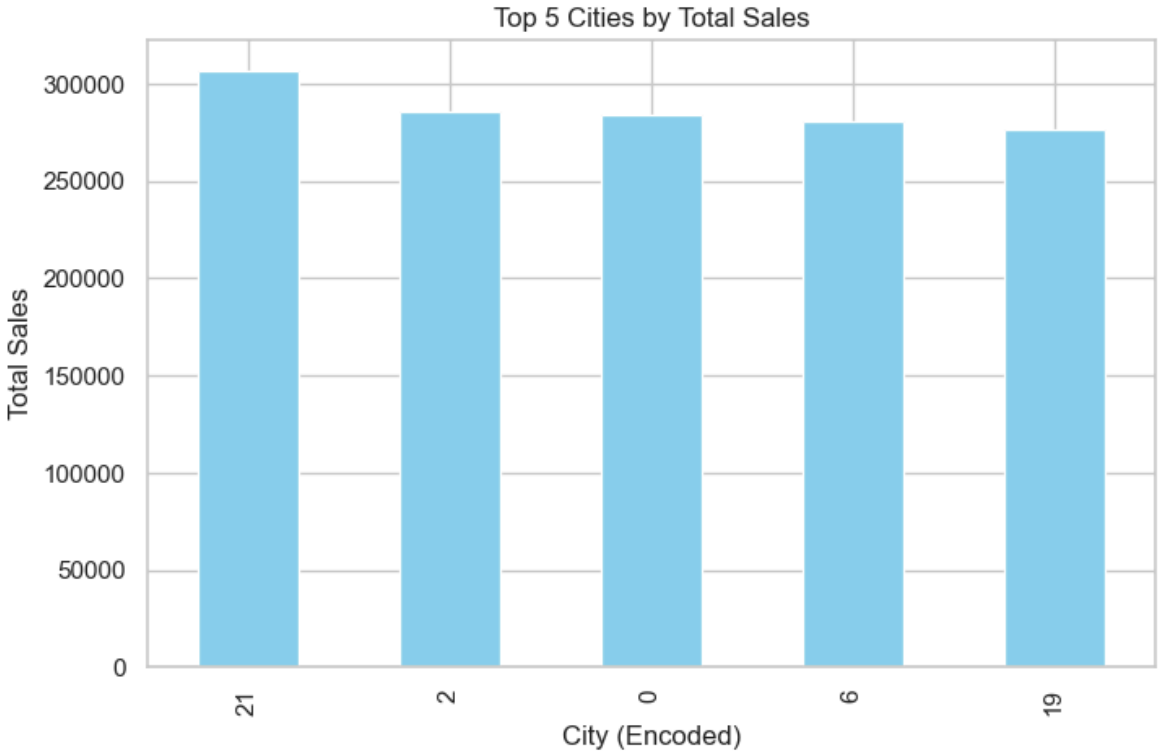
In [34]:
```
# Cell 11: Monthly Sales Pattern

monthly_sales = df.groupby('Order Month')['Sales'].sum().reindex(range(1,13), fi

plt.figure(figsize=(10, 5))
monthly_sales.plot(kind='line', marker='o')
plt.xticks(np.arange(1,13),
           [ 'Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','
plt.title("Sales by Month")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.grid(True)
plt.show()
```



In [35]:
```
# Cell 12: Yearly Sales (Pie Chart)

yearly_sales = df.groupby('Order Year')['Sales'].sum()
plt.figure(figsize=(7, 7))
plt.pie(yearly_sales, labels=yearly_sales.index, autopct='%1.1f%%', startangle=9
plt.title('Sales by Year')
plt.axis('equal')
plt.show()
```



In [36]:
```
# Cell 13: Top 5 Cities by Total Sales

city_sales = df.groupby('City')['Sales'].sum()
top_cities = city_sales.sort_values(ascending=False).head(5)

plt.figure(figsize=(8,5))
top_cities.plot(kind='bar', color='skyblue')
plt.title('Top 5 Cities by Total Sales')
plt.xlabel('City (Encoded)')
plt.ylabel('Total Sales')
plt.show()

# Print city encoding for interpretation
print("City Encodings:", dict(enumerate(le.classes_)))
```




```
City Encodings: {0: 'April', 1: 'August', 2: 'December', 3: 'February', 4: 'Janua
ry', 5: 'July', 6: 'June', 7: 'March', 8: 'May', 9: 'November', 10: 'October', 1
1: 'September'}
```

In [37]:
```
# Cell 14: Feature Selection for Modeling

feature_columns = ['Category', 'Sub Category', 'City', 'Region', 'State', 'Order

X = df[feature_columns]
y = df['Sales']

X.head()
```

Out[37]:

	Category	Sub Category	City	Region	State	Order Month	Discount	Profit
0	5	14	21	2	0	11	0.12	401.28
1	1	13	8	3	0	11	0.18	149.80
2	3	0	13	4	0	6	0.21	165.20
3	4	12	4	3	0	10	0.25	89.60
4	3	18	12	3	0	10	0.26	918.45

In [38]:
```
# Cell 15: Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(3233, 8) (809, 8) (3233,) (809,)
```

In [40]:
```
# Cell 16: Feature Scaling
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

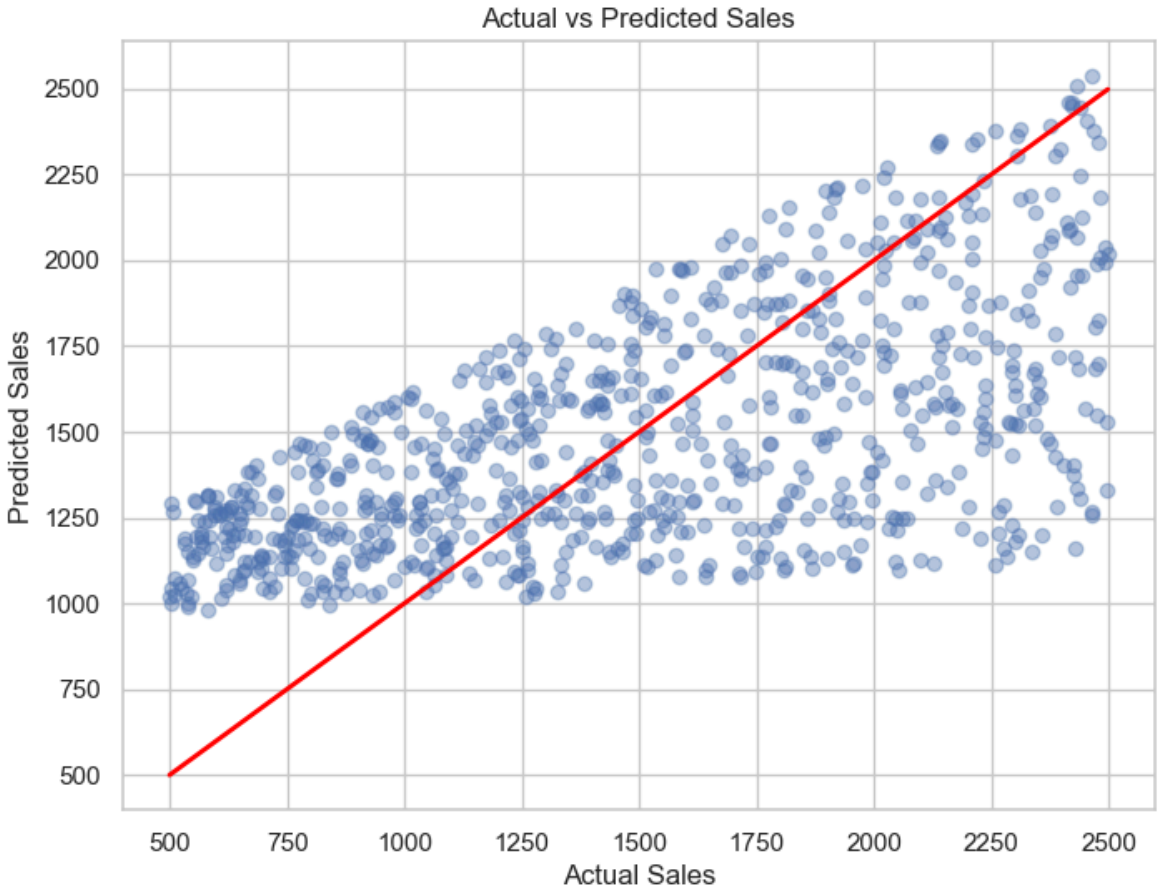
In [41]:
```
# Cell 17: Model Training - Linear Regression

from sklearn.impute import SimpleImputer

# Impute missing values BEFORE scaling
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_test_scaled = scaler.transform(X_test_imputed)
```

In [42]:
```
model = LinearRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

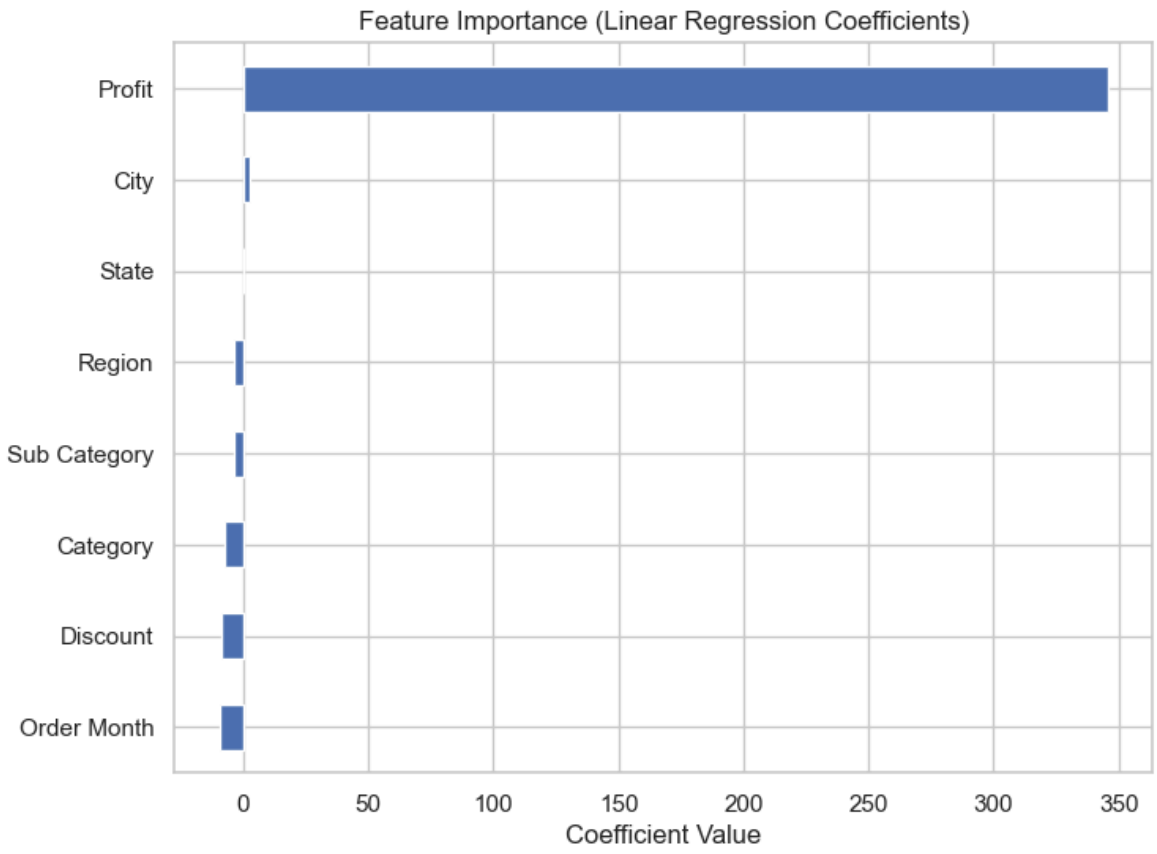
In [43]:
```
# Cell 18: Model Evaluation

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
print(f"R^2 Score: {r2:.2f}")
```

```
Mean Squared Error: 211366.56
R^2 Score: 0.37
```

In [44]:
```
# Cell 19: Plot Actual vs Predicted Sales

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.4)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', li
plt.title('Actual vs Predicted Sales')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.grid(True)
plt.show()
```



In [45]:
```
# Cell 20: Feature Importance (Linear Regression Coefficients)

feature_importance = pd.Series(model.coef_, index=feature_columns)
feature_importance.sort_values().plot(kind='barh', figsize=(8,6))
plt.title("Feature Importance (Linear Regression Coefficients)")
plt.xlabel("Coefficient Value")
plt.show()
```

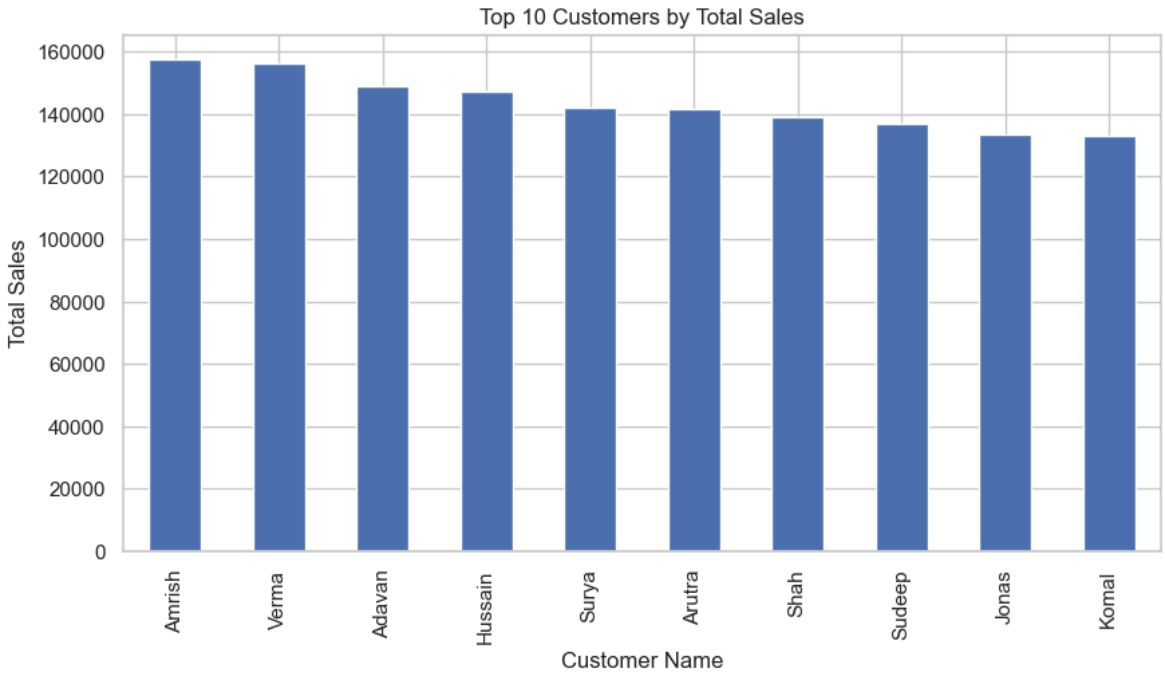


In [46]:
```
print("Number of unique customers:", df['Customer Name'].nunique())
```

```
Number of unique customers: 50
```

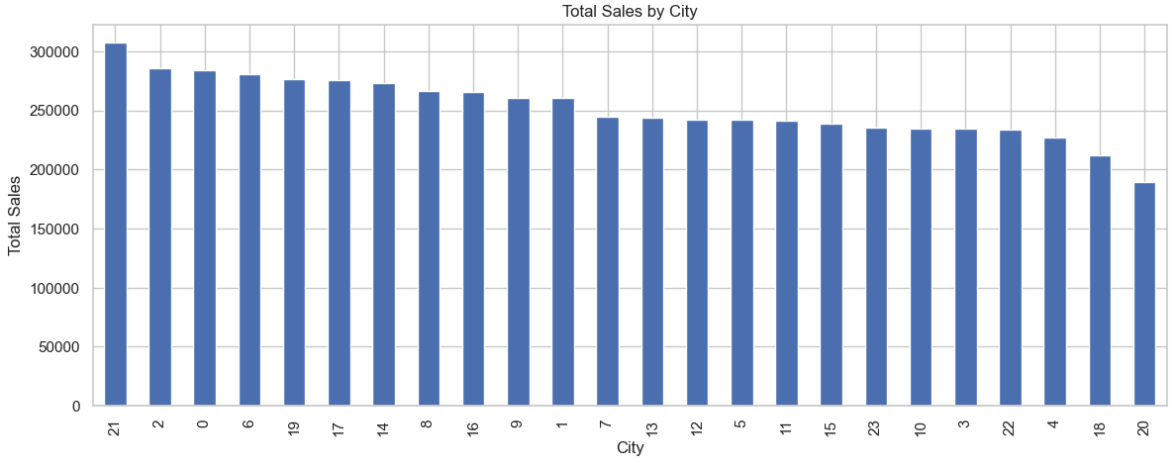
In [47]:
```
#Top customers by total sales

top_customers = df.groupby("Customer Name")["Sales"].sum().sort_values(ascending
print(top_customers)
top_customers.plot(kind='bar', figsize=(10,5), title="Top 10 Customers by Total
plt.ylabel("Total Sales")
plt.show()
```

```
Customer Name
Amrish     157400
Verma      156248
Adavan     148837
Hussain    147228
Surya      142099
Arutra     141448
Shah       138827
Sudeep     136704
Jonas      133593
Komal      133173
Name: Sales, dtype: int64
```

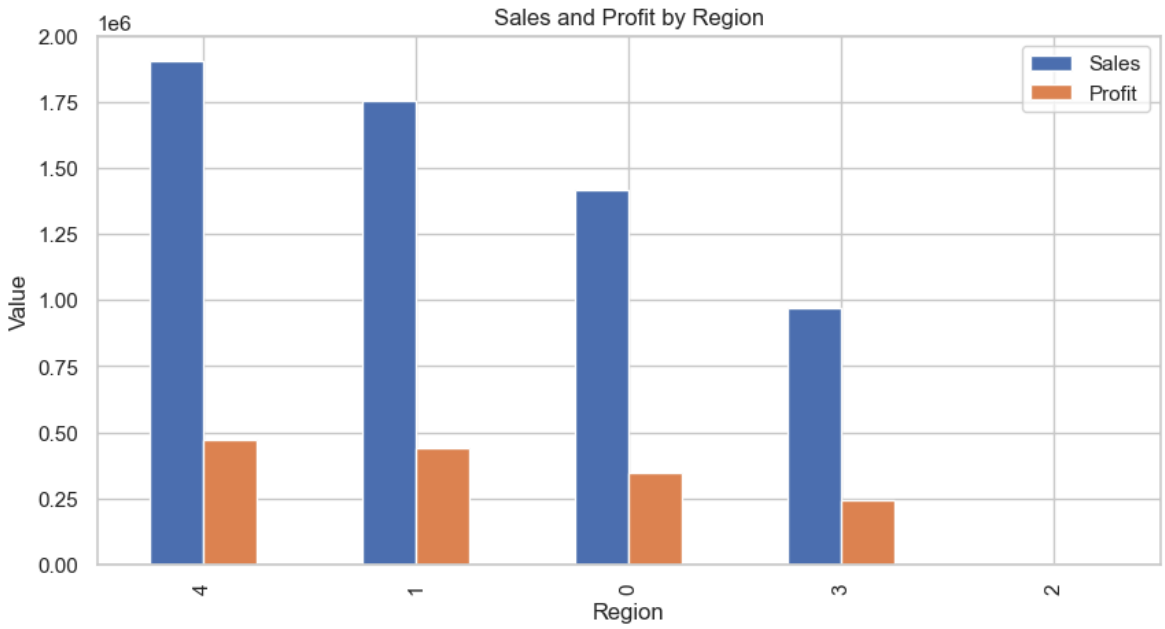


In [48]:
```
#City-wise Analysis
#Number of unique cities and sales per city
print("Number of unique cities:", df['City'].nunique())
city_sales = df.groupby('City')['Sales'].sum().sort_values(ascending=False)
city_sales.plot(kind='bar', figsize=(14,5), title='Total Sales by City')
plt.ylabel("Total Sales")
plt.show()
```

```
Number of unique cities: 24
```

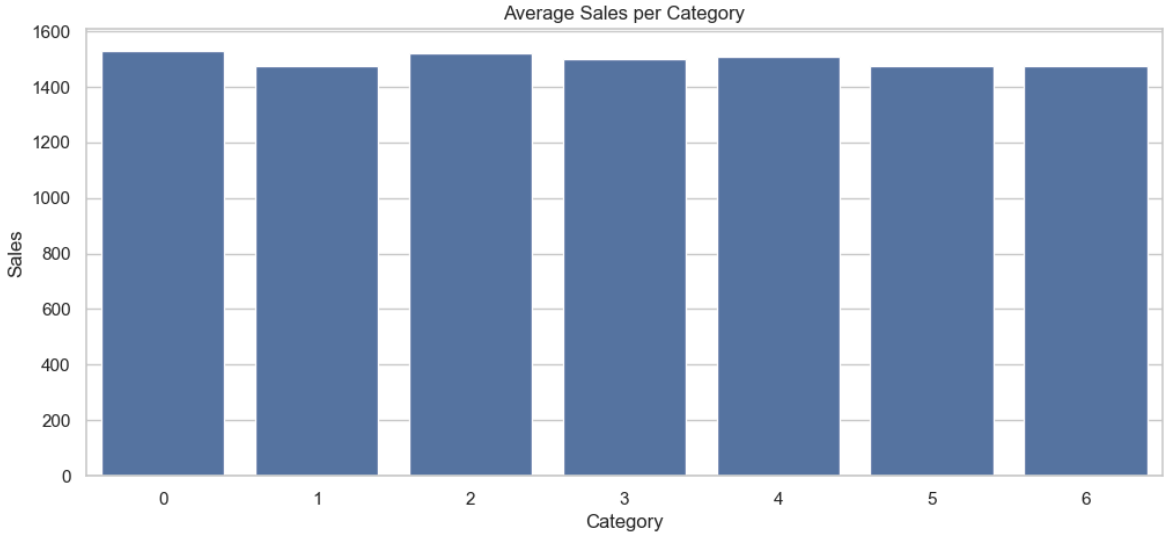


In [49]:
```
#Region/Customer Segmentation Analysis
#Sales and profit by region
region_sales = df.groupby('Region')[['Sales', 'Profit']].sum().sort_values('Sale
region_sales.plot(kind='bar', figsize=(10,5), title="Sales and Profit by Region"
plt.ylabel("Value")
plt.show()
```

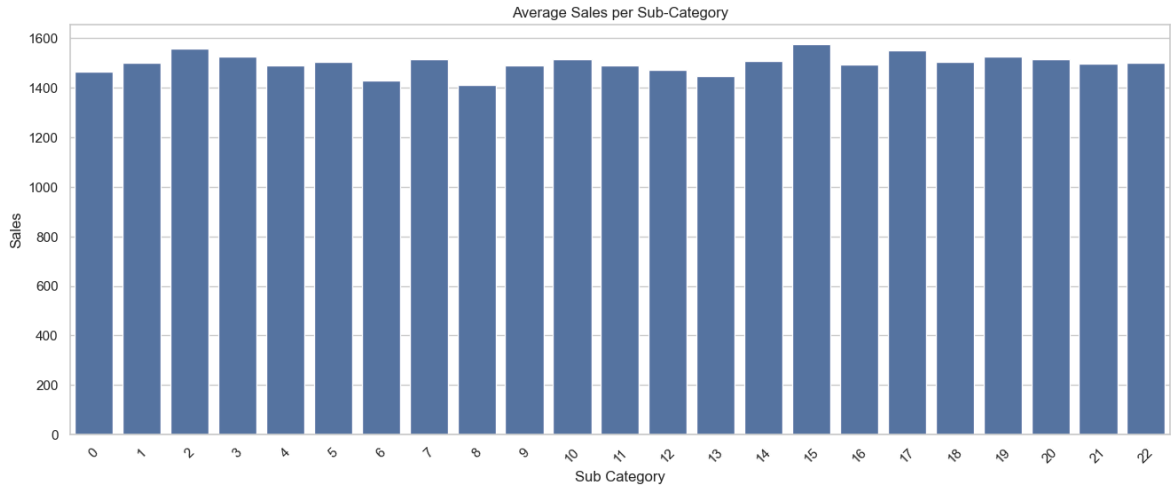


In [50]:
```
#Product/Category Analysis
#Average sales per category and sub-category
plt.figure(figsize=(12,5))
sns.barplot(x='Category', y='Sales', data=df, ci=None, estimator=np.mean)
plt.title('Average Sales per Category')
plt.show()

plt.figure(figsize=(16,6))
sns.barplot(x='Sub Category', y='Sales', data=df, ci=None, estimator=np.mean)
plt.title('Average Sales per Sub-Category')
plt.xticks(rotation=45)
plt.show()
```

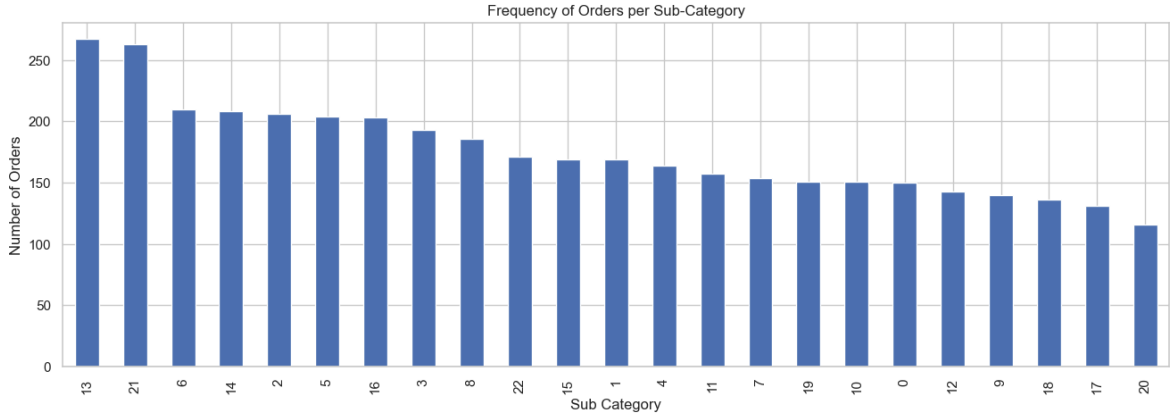
```
C:\Users\kkjeg\AppData\Local\Temp\ipykernel_2064\1323773504.py:4: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Category', y='Sales', data=df, ci=None, estimator=np.mean)
```



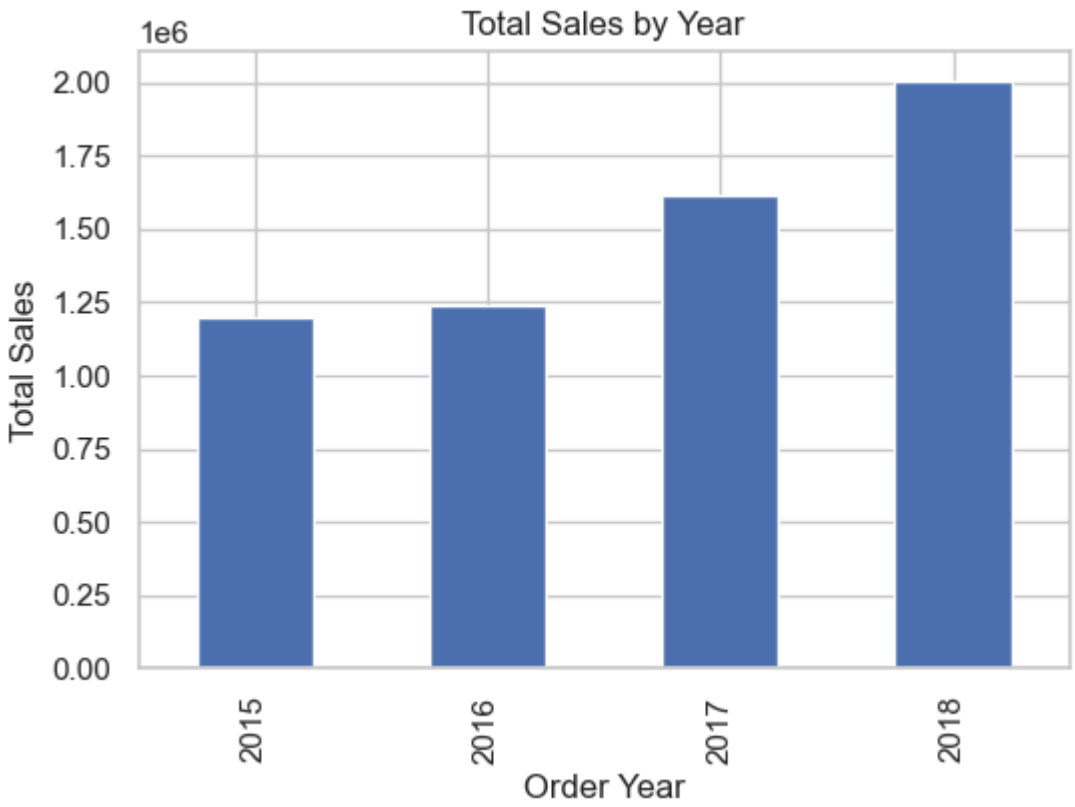

```
C:\Users\kkjeg\AppData\Local\Temp\ipykernel_2064\1323773504.py:9: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='Sub Category', y='Sales', data=df, ci=None, estimator=np.mean)
```



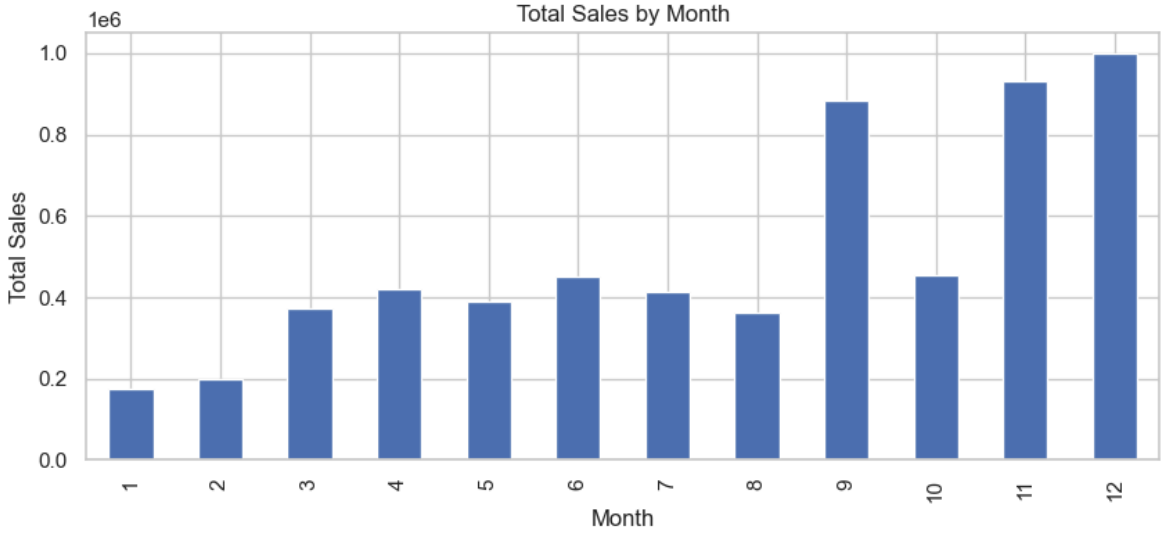
In [51]:
```
#frequencies of sub-categories sold
plt.figure(figsize=(16,5))
df['Sub Category'].value_counts().plot(kind='bar')
plt.title('Frequency of Orders per Sub-Category')
plt.ylabel('Number of Orders')
plt.show()
```



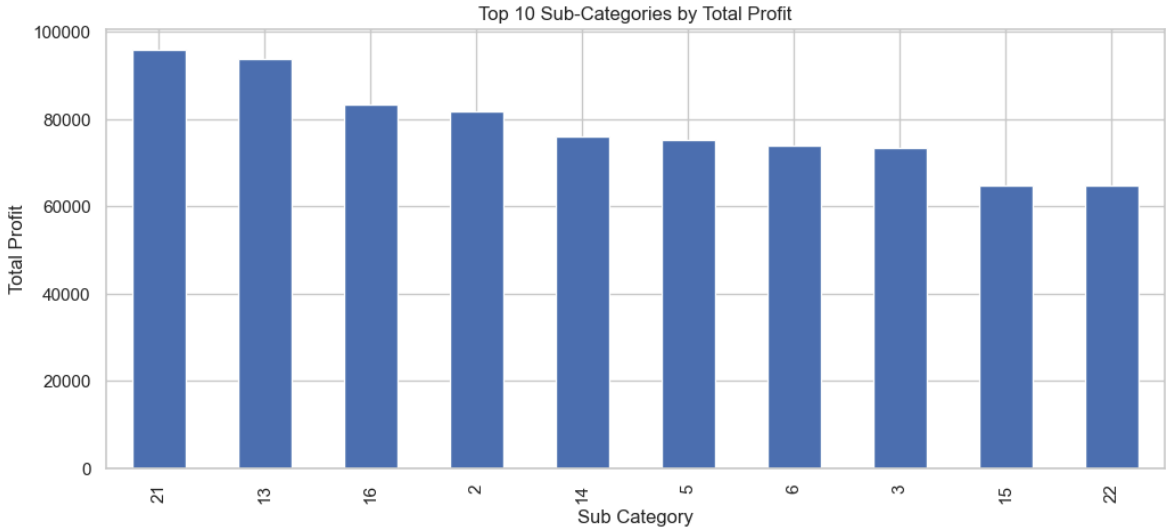
In [52]:
```
#Time-based Analysis (Hour Not Present, But You Can Analyze by Day/Month/Year)
#Sales trend by year
yearly_sales = df.groupby('Order Year')['Sales'].sum()
yearly_sales.plot(kind='bar', figsize=(6,4), title='Total Sales by Year')
plt.ylabel('Total Sales')
plt.show()
```



In [53]:
```
#Sales trend by Month
monthly_sales = df.groupby('Order Month')['Sales'].sum().sort_index()
monthly_sales.plot(kind='bar', figsize=(10,4), title='Total Sales by Month')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```



In [54]:
```
#Profit margin analysis
#top sub categories by total profit
subcat_profit = df.groupby('Sub Category')['Profit'].sum().sort_values(ascending
subcat_profit.head(10).plot(kind='bar', figsize=(12,5), title='Top 10 Sub-Catego
plt.ylabel('Total Profit')
plt.show()
```


Top 10 Sub-Categories by Total Profit
Total Profit
100000
80000
60000
40000
20000
0
21
13
16
2
14
5
6
3
15
22
Sub Category