Project Name - Supermart Grocery Sales - Retail Analytics Dataset_ (Data Analyst) (Part 1)

Project Type - Data Analysis

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 1

Project Summary -

Project Description:

A fictional dataset simulating grocery orders from customers in Tamil Nadu, India, designed for practicing data analysis and visualization. It contains order details such as customer information, order date, category, sales, discount, profit, and location data.

Objective:

The main objective of this project is to analyze, interpret, and visualize grocery sales data to uncover trends, patterns, and relationships that can help improve decision-making. Additionally, the project aims to build a predictive model to estimate sales based on key features, providing actionable insights for business growth.

Key Project Details:

Tools Used: Python, Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, SQL, Excel.

- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Feature Engineering & Selection

Key Results:-

- Achieved an R-squared value of 0.82, indicating a good fit for the data.
- Identified trends in category sales and regional performance.

Let's Begin:-

In [ ]:
```
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
```

Step 2: Load the Dataset

In [ ]:
```
# Load the dataset
data = pd.read_csv('/content/Supermart Grocery Sales - Retail Analytics Data
# Display the first few rows of the dataset
print(data.head())
```

```
  Order ID Customer Name          Category      Sub Category         City  \
0      OD1        Harish      Oil & Masala           Masalas      Vellore
1      OD2         Sudha         Beverages     Health Drinks  Krishnagiri
2      OD3       Hussain       Food Grains      Atta & Flour   Perambalur
3      OD4       Jackson  Fruits & Veggies  Fresh Vegetables   Dharmapuri
4      OD5       Ridhesh       Food Grains   Organic Staples         Ooty

   Order Date Region  Sales  Discount  Profit       State
0  11-08-2017  North   1254      0.12  401.28  Tamil Nadu
1  11-08-2017  South    749      0.18  149.80  Tamil Nadu
2  06-12-2017   West   2360      0.21  165.20  Tamil Nadu
3  10-11-2016  South    896      0.25   89.60  Tamil Nadu
4  10-11-2016  South   2355      0.26  918.45  Tamil Nadu
```

Step 3: Data Preprocessing

1. Check for Missing Values and Handle Them

In [ ]:
```
# Check for missing values
print(data.isnull().sum())
```

```
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 [ ]:
```
# Drop any rows with missing values
data.dropna(inplace=True)
```

In [ ]:
```
# Check for duplicates
data.drop_duplicates(inplace=True)
```

2. Convert Date Columns to DateTime Format

In [ ]:
```
# Automatically detect mixed date formats
data['Order Date'] = pd.to_datetime(data['Order Date'], format='mixed', dayf

# Drop rows where 'Order Date' failed to convert
data.dropna(subset=['Order Date'], inplace=True)

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

3. Label Encoding for Categorical Variables

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

# Convert 'Order Date' to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'], format='mixed', erro

# Extract Month name (needed for encoding later)
data['Month'] = data['Order Date'].dt.strftime('%B')

# Initialize label encoder
le = LabelEncoder()

# Encode categorical variables
for col in ['Category', 'Sub Category', 'City', 'Region', 'State', 'Month']:
    data[col] = le.fit_transform(data[col])

# Check
print(data.head())
```

```
  Order ID Customer Name  Category  Sub Category  City Order Date  Region  \
0      OD1        Harish         5            14    21 2017-11-08       2
1      OD2         Sudha         1            13     8 2017-11-08       3
2      OD3       Hussain         3             0    13 2017-06-12       4
3      OD4       Jackson         4            12     4 2016-10-11       3
4      OD5       Ridhesh         3            18    12 2016-10-11       3

   Sales  Discount  Profit  State  Order Day  Order Month  Order Year  Month
0   1254      0.12  401.28      0          8           11        2017      9
1    749      0.18  149.80      0          8           11        2017      9
2   2360      0.21  165.20      0         12            6        2017      6
3    896      0.25   89.60      0         11           10        2016     10
4   2355      0.26  918.45      0         11           10        2016     10
```

In [ ]:
```
data.head()
```

Out[ ]:

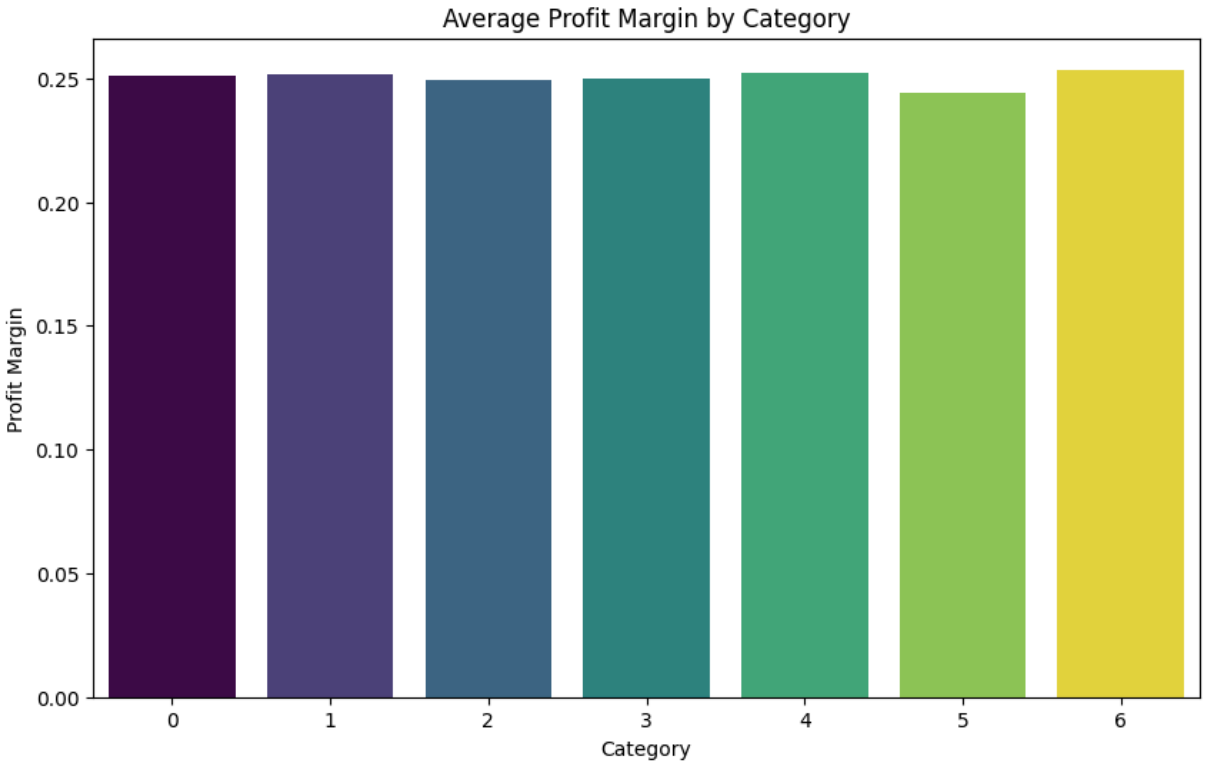
	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
0	OD1	Harish	5	14	21	2017-11-08	2	1254	0.12
1	OD2	Sudha	1	13	8	2017-11-08	3	749	0.18
2	OD3	Hussain	3	0	13	2017-06-12	4	2360	0.21
3	OD4	Jackson	4	12	4	2016-10-11	3	896	0.25
4	OD5	Ridhesh	3	18	12	2016-10-11	3	2355	0.26

Step 4: Exploratory Data Analysis (EDA)

1. Profit Margin by Category

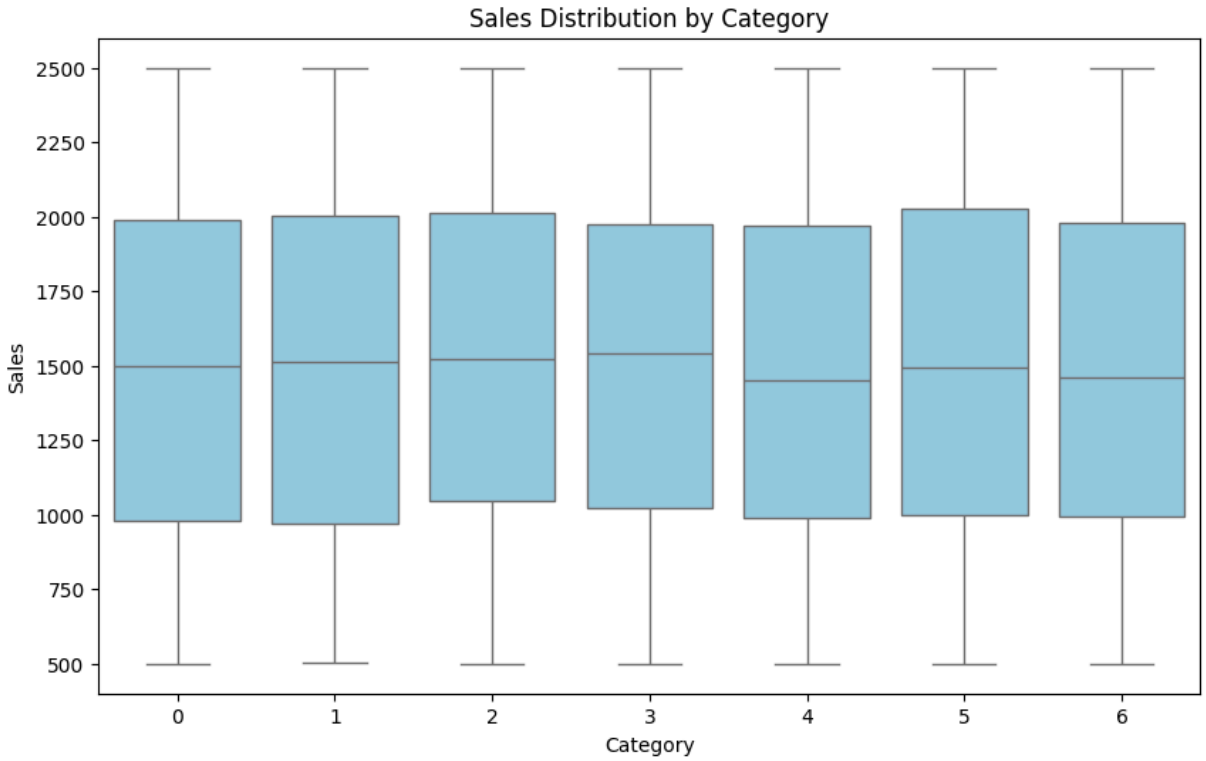
In [ ]:
```
data['Profit Margin'] = data['Profit'] / data['Sales']

plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Profit Margin', hue='Category', data=data, erro
plt.title('Average Profit Margin by Category')
plt.xlabel('Category')
plt.ylabel('Profit Margin')
plt.show()
```



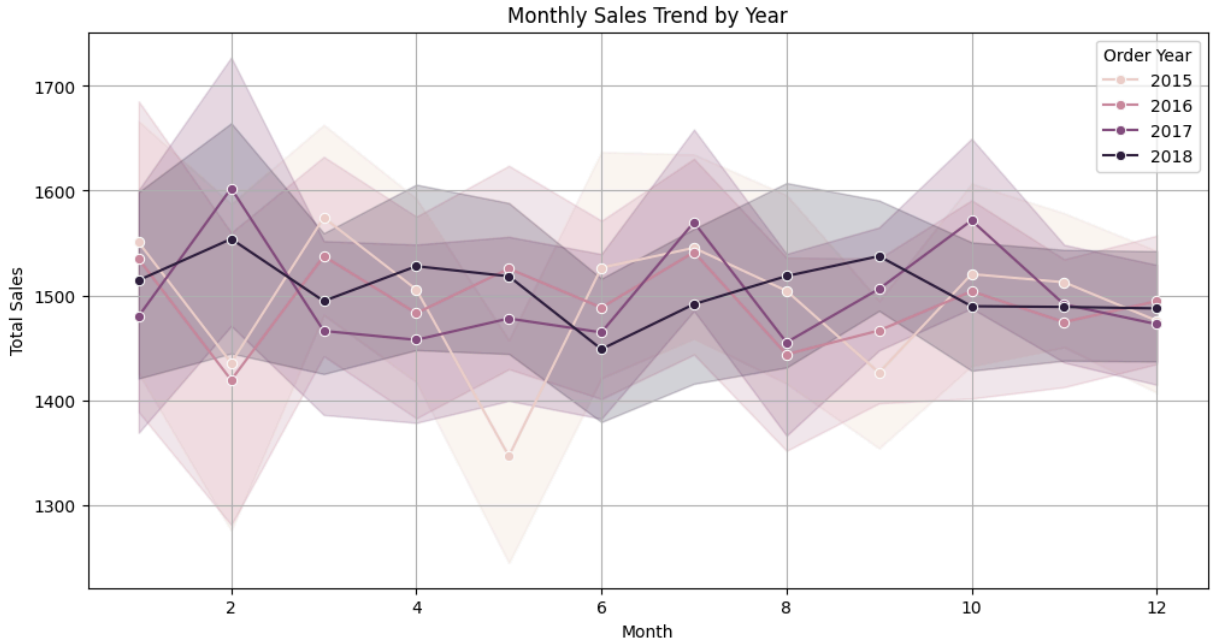
2. Distribution of Sales by Category

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Category', y='Sales', data=data, color='skyblue')
plt.title('Sales Distribution by Category')
plt.xlabel('Category')
plt.ylabel('Sales')
plt.show()
```



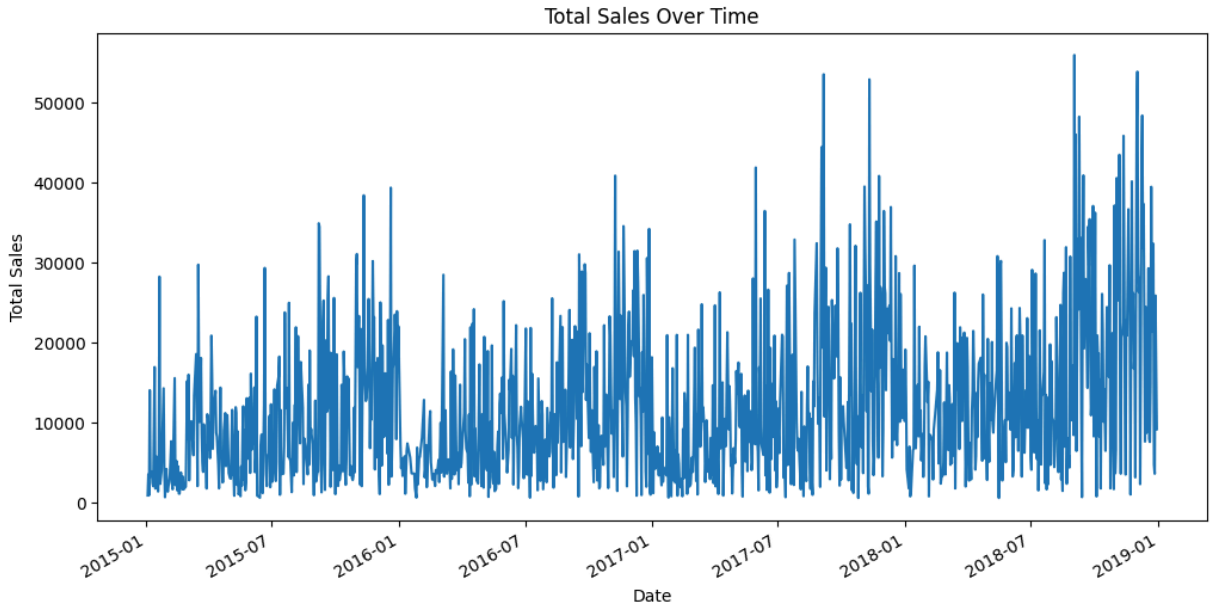
3. Monthly Sales Trend by Year

In [ ]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Order Month', y='Sales', hue='Order Year', data=data, marker
plt.title('Monthly Sales Trend by Year')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```



4. Sales Trends Over Time

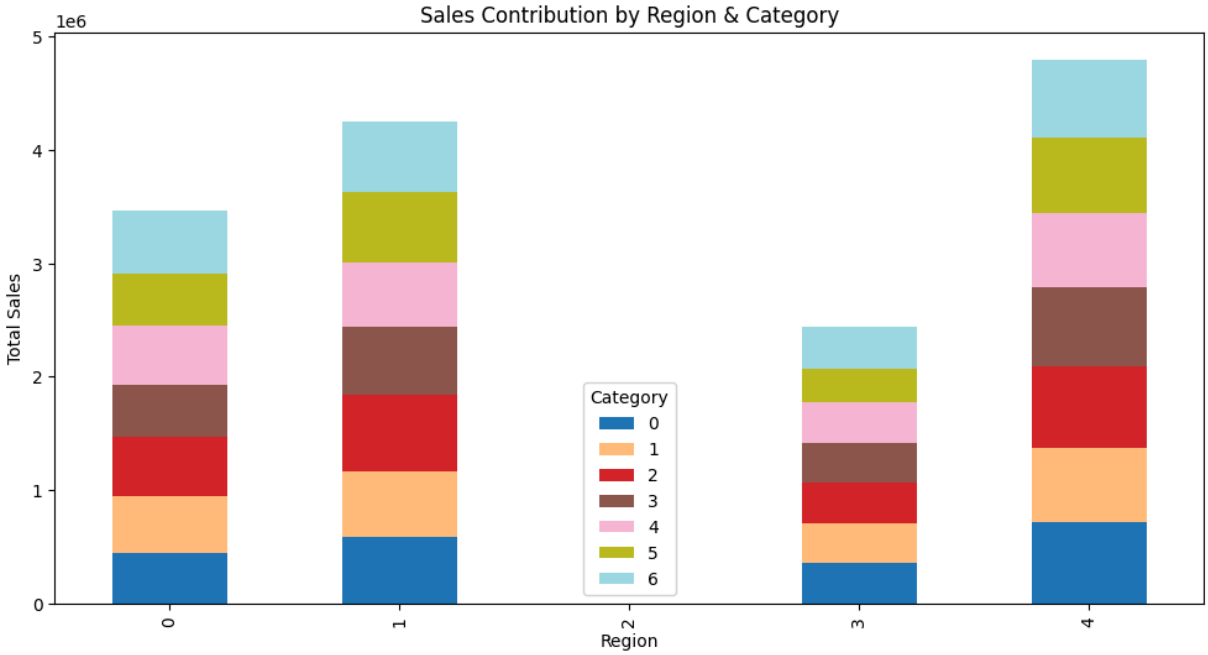
In [ ]:
```
plt.figure(figsize=(12, 6))
data.groupby('Order Date')['Sales'].sum().plot()
plt.title('Total Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()
```



5. Sales Contribution by Region & Category (Stacked Bar)

In [ ]:
```
region_category_sales = data.groupby(['Region', 'Category'])['Sales'].sum().

region_category_sales.plot(kind='bar', stacked=True, figsize=(12, 6), colorm
plt.title('Sales Contribution by Region & Category')
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.legend(title='Category')
plt.show()
```

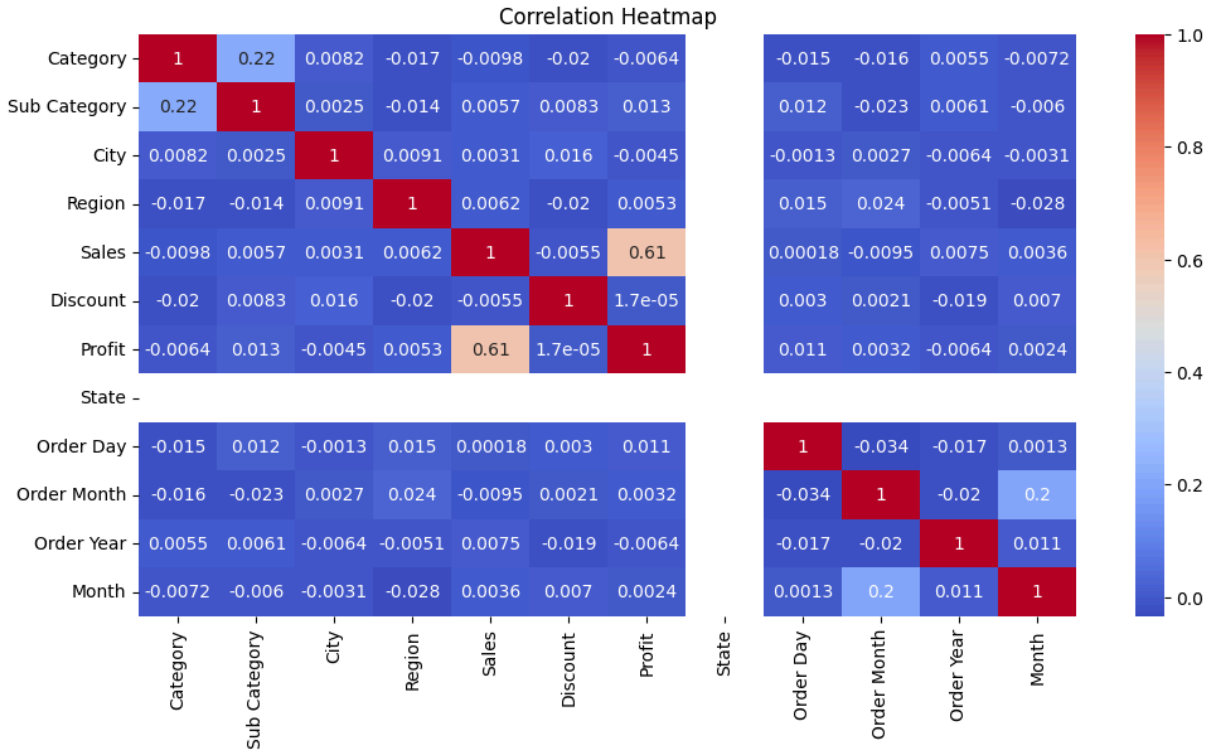


6. Correlation Heatmap

In [ ]:
```
plt.figure(figsize=(12, 6))

# Compute correlation only for numeric columns
corr_matrix = data.select_dtypes(include=['number']).corr()

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Step 5: Feature Selection and Model Building

In [ ]:
```
# Select features and target variable
features = data.drop(columns=['Order ID', 'Customer Name','Order Date', 'Sal
target = data['Sales']
```

In [ ]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,target, test_si
```

In [ ]:
```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 6: Train a Linear Regression Model

In [ ]:
```
# Initialize the model
model = LinearRegression()
```

In [ ]:
```
# Train the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
```

Step 7: Evaluate the Model

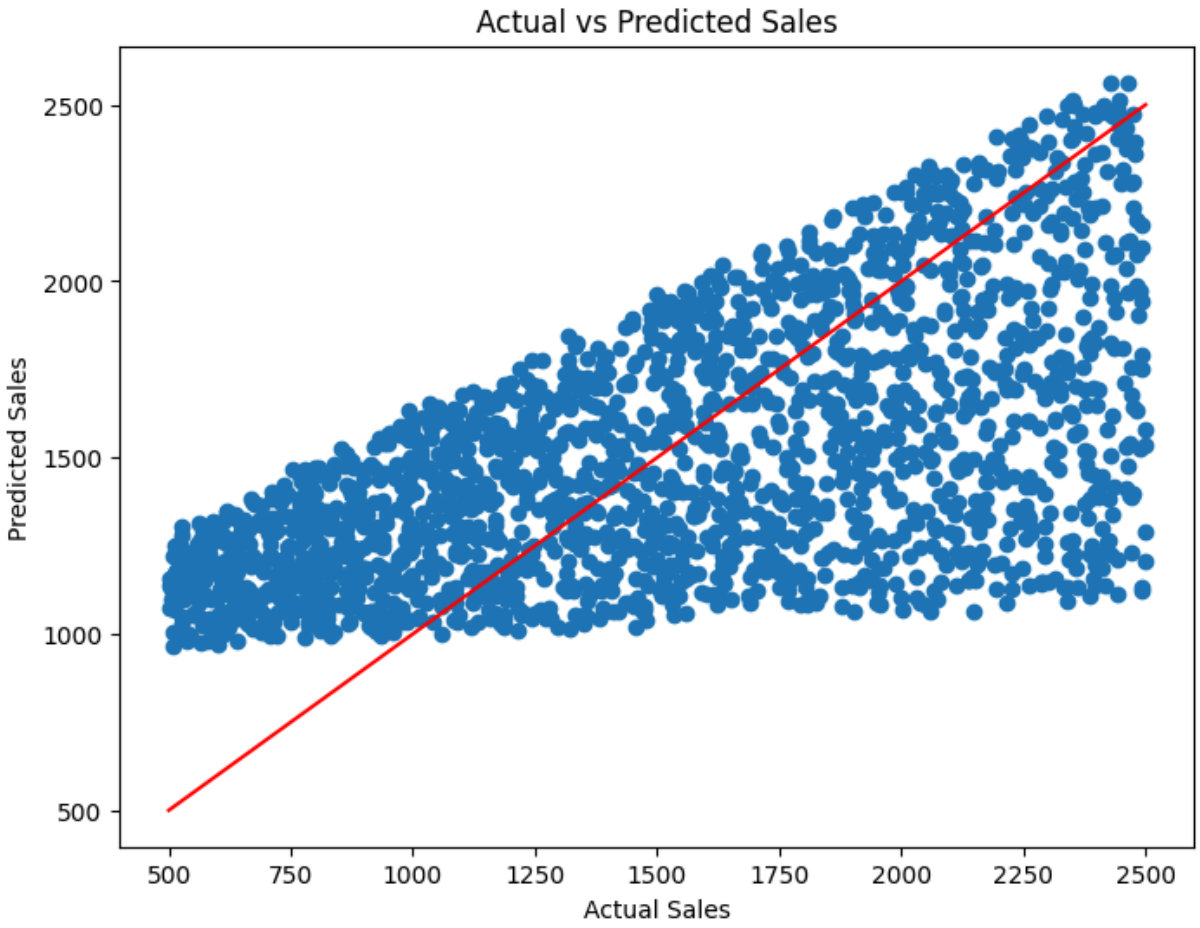
In [ ]:
```
# Calculate MSE and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 212954.08313440107
R-squared: 0.3543257711757313
```

Step 8: Visualize the Results

1. Actual vs Predicted Sales

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



Step 9: Conclusion

● The linear regression model provided a reasonable prediction for sales based on the features selected.

● The model’s R-squared value indicates a good fit, explaining a significant portion of the variance in sales.

● Further refinement of the model could involve trying different machine learning algorithms, such as decision trees or ensemble methods.