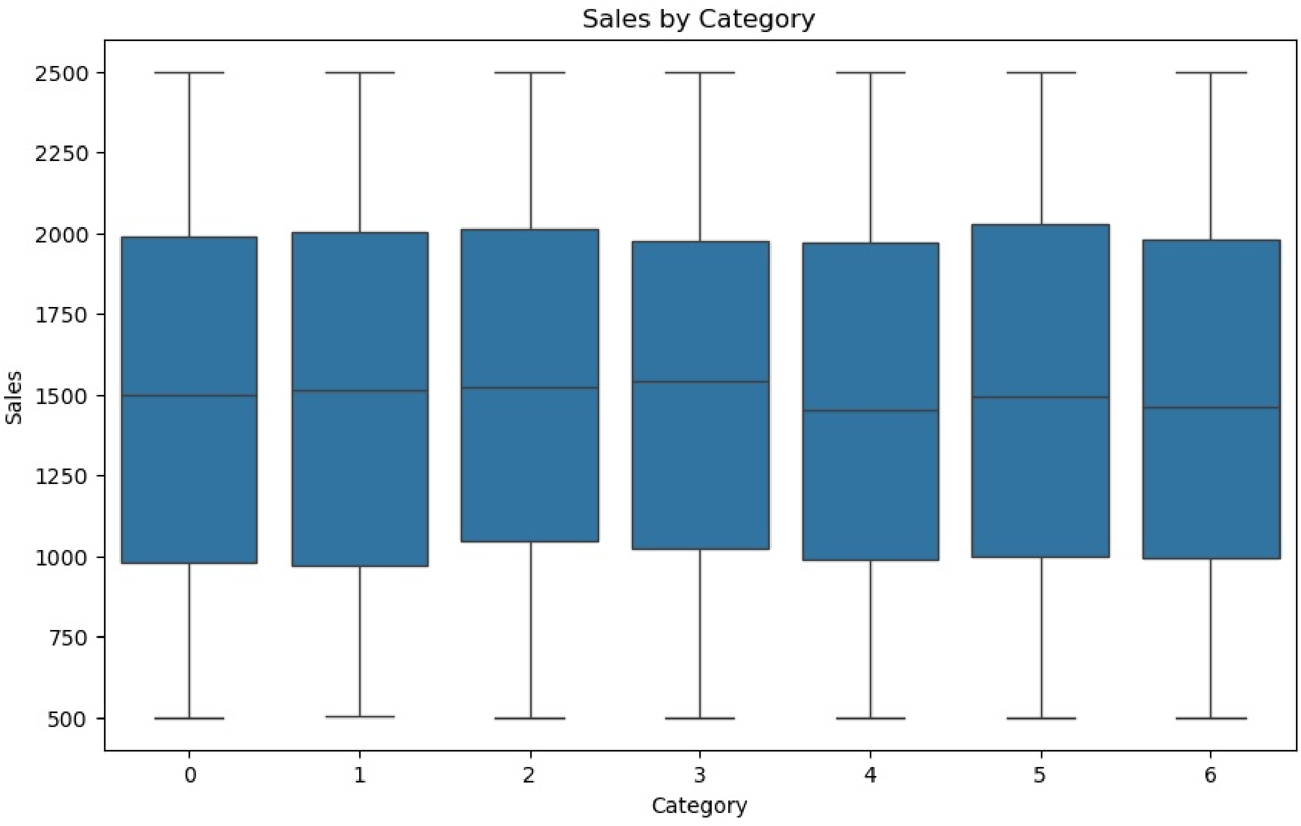
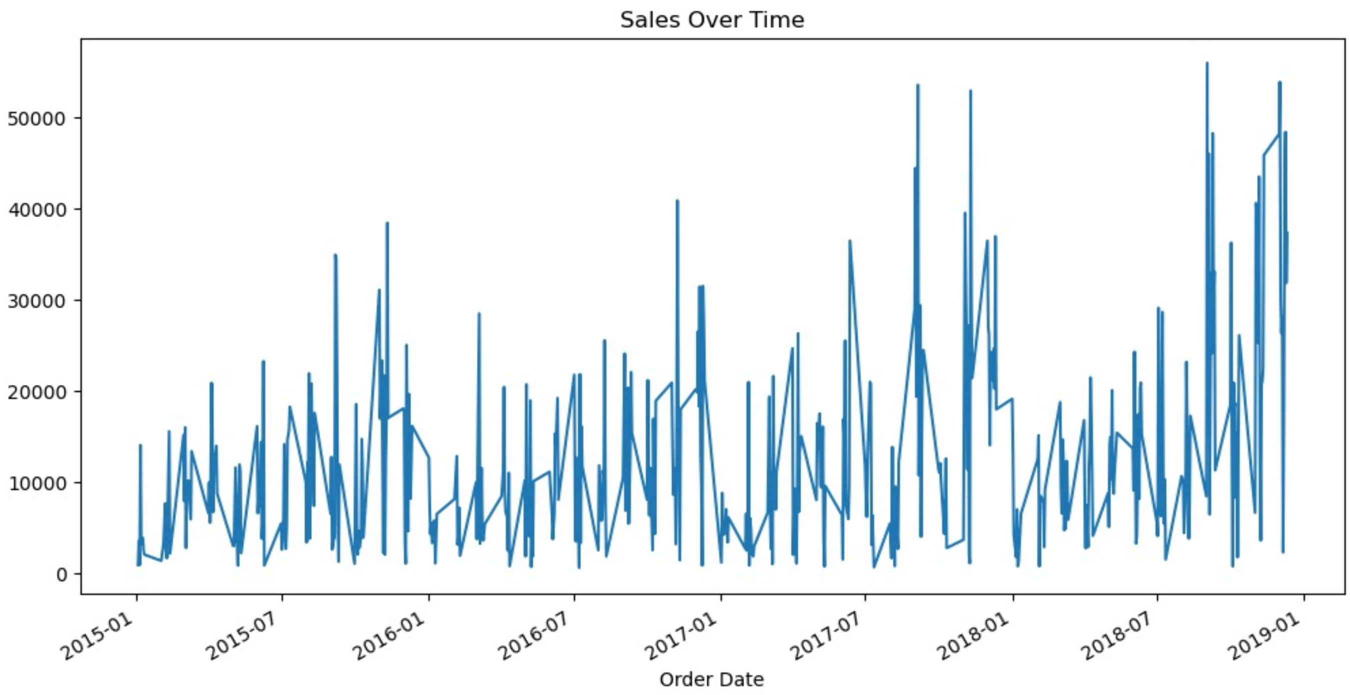
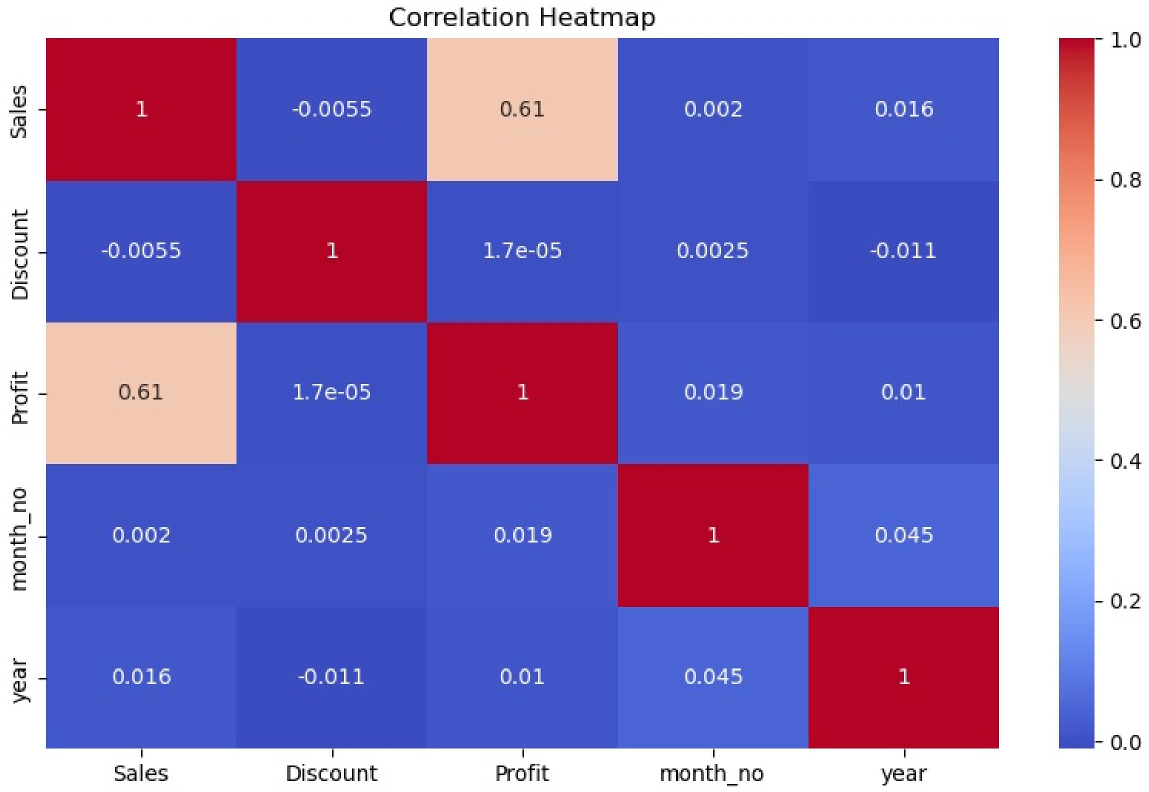
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import LabelEncoder, StandardScaler
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error, r2_score
```

```
In [2]: df = pd.read_csv("C:\\Users\\HP\\Downloads\\Supermart Grocery Sales - Retail Analytics Dataset.csv")

        df.drop_duplicates(inplace=True)
        df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')
        df['month_no'] = df['Order Date'].dt.month
        df['Month'] = df['Order Date'].dt.strftime('%B')
        df['year'] = df['Order Date'].dt.year

        le = LabelEncoder()
        for col in ['Category', 'Sub Category', 'City', 'Region', 'State', 'Month']:
            df[col] = le.fit_transform(df[col])
```

```
In [10]: # Visualizations
         plt.figure(figsize=(10, 6))
         sns.boxplot(x='Category', y='Sales', data=df)
         plt.title('Sales by Category')
         plt.show()

         plt.figure(figsize=(12, 6))
         df.groupby('Order Date')['Sales'].sum().plot()
         plt.title('Sales Over Time')
         plt.show()

         numeric_df = df.select_dtypes(include=['int64', 'float64'])
         plt.figure(figsize=(10, 6))
         sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
         plt.title('Correlation Heatmap')
         plt.show()
```

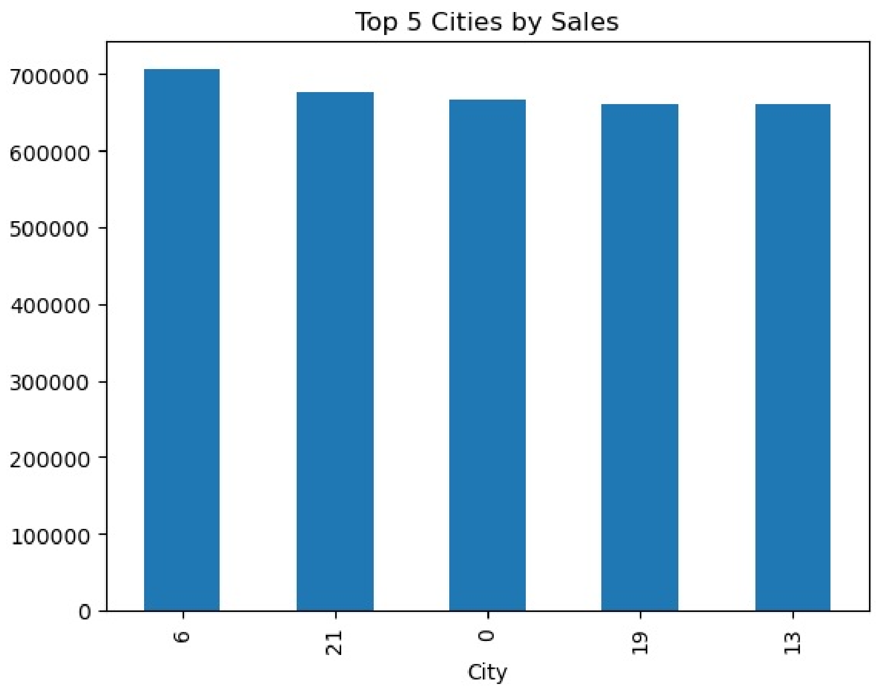






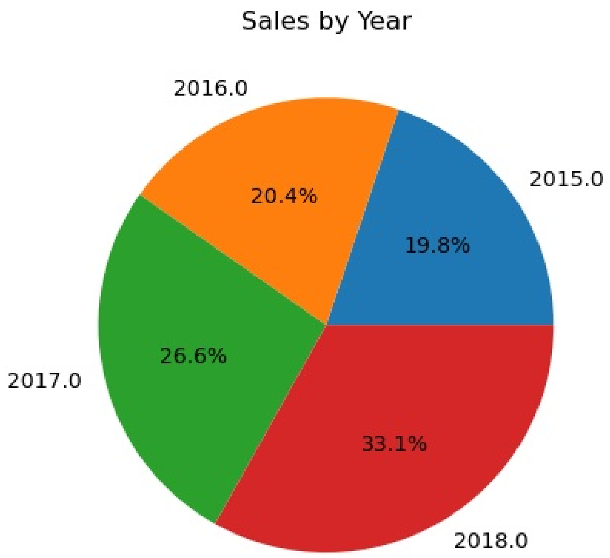
In [4]:

```
# Top 5 Cities
top_cities = df.groupby('City')['Sales'].sum().sort_values(ascending=False).head(5)
top_cities.plot(kind='bar')
plt.title('Top 5 Cities by Sales')
plt.show()
```



In [5]:

```
# Yearly Sales
yearly_sales = df.groupby('year')['Sales'].sum()
plt.pie(yearly_sales, labels=yearly_sales.index, autopct='%1.1f%%')
plt.title('Sales by Year')
plt.show()
```



In [8]:

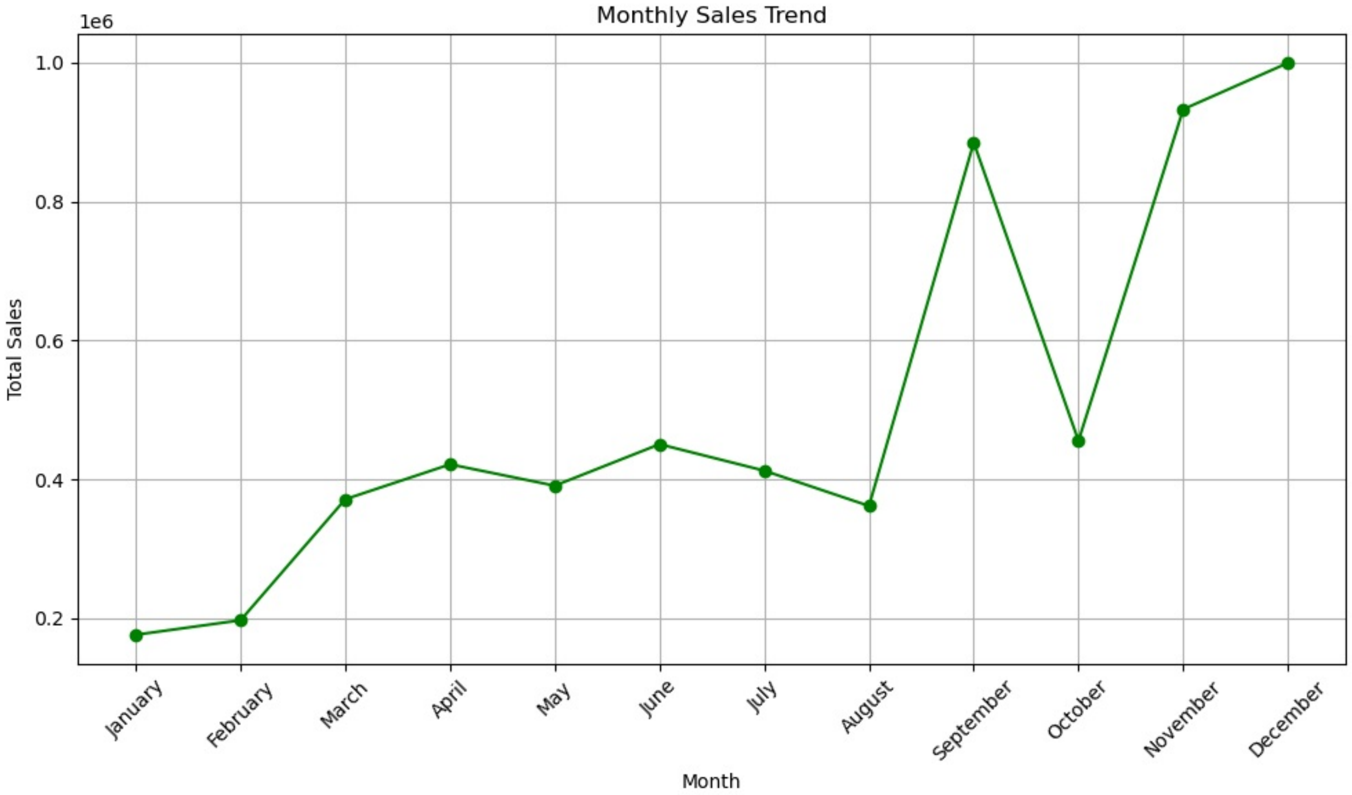
```
from sklearn.impute import SimpleImputer

# Drop target NaNs (y must not have NaNs)
df = df.dropna(subset=['Sales'])

# Split features and target
X = df.drop(columns=['Order ID', 'Customer Name', 'Order Date', 'Sales'])
y = df['Sales']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Predict on test data
y_pred = model.predict(X_test)

# Evaluate Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MSE: {mse:.2f}")
print(f"R² Score: {r2:.2f}")
```

```
MSE: 212734.13
R² Score: 0.35
```

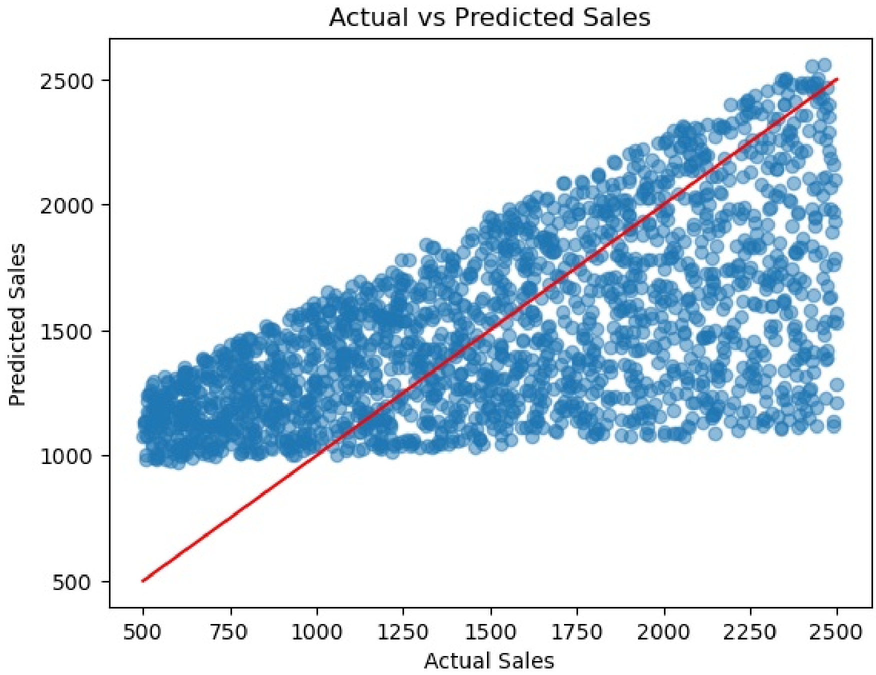
In [11]:

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

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

month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']

monthly_sales = df.groupby('Month')['Sales'].sum().reindex(month_order)

plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', color='green')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [9]:

```
# Actual vs Predicted
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Actual vs Predicted Sales")
plt.show()
```



Actual vs Predicted Sales
Predicted Sales
2500
2000
1500
1000
500
Actual Sales
500
750
1000
1250
1500
1750
2000
2250
2500

```
In [ ]:
```