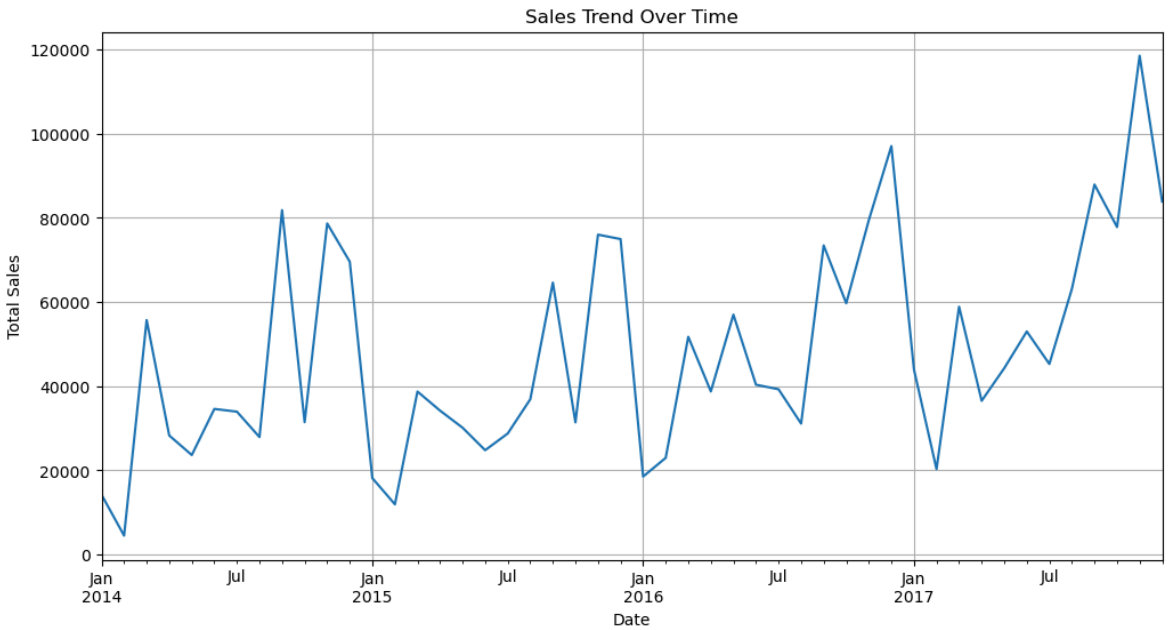
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# Load the dataset
file_path = 'SalesData.csv'  # Adjust the path as necessary
data = pd.read_csv(file_path)

# Inspect the dataset
print("Shape of the dataset:", data.shape)
print("Missing values:\n", data.isnull().sum())
print("Data types:\n", data.dtypes)
```

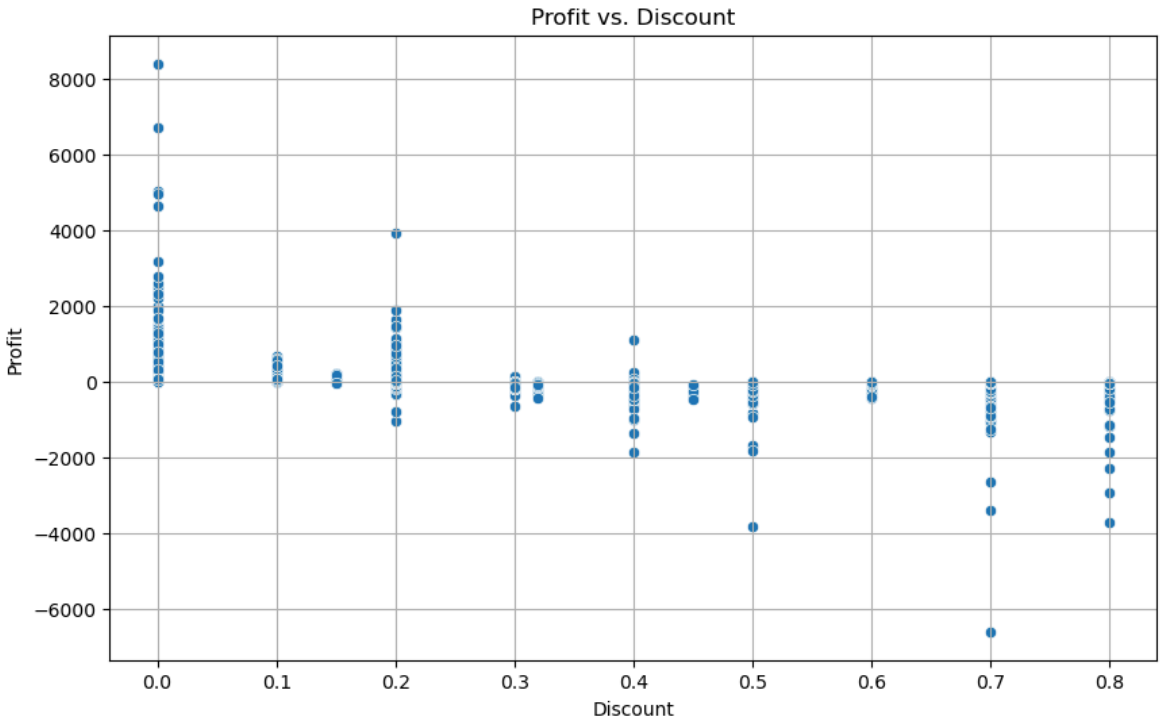
```
Shape of the dataset: (9994, 19)
Missing values:
 Row ID          0
Order ID        0
Order Date      0
Ship Date       0
Ship Mode       0
Customer ID     0
Segment         0
Country         0
City            0
State           0
Region          0
Product ID      0
Category        0
Sub-Category    0
Product Name    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
Data types:
 Row ID           int64
Order ID         object
Order Date       object
Ship Date        object
Ship Mode        object
Customer ID      object
Segment          object
Country          object
City             object
State            object
Region           object
Product ID       object
Category         object
Sub-Category     object
Product Name     object
Sales           float64
Quantity          int64
Discount        float64
Profit          float64
dtype: object
```

In [5]:
```
# Remove duplicates
data.drop_duplicates(inplace=True)

# Fill missing values
# For numerical columns, fill with mean or median
for column in data.select_dtypes(include=[np.number]).columns:
    data[column] = data[column].fillna(data[column].mean())


# For categorical columns, fill with mode
for column in data.select_dtypes(include=[object]).columns:
   data[column] = data[column].fillna(data[column].mode()[0])


# Convert the 'Order Date' column to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'], format='mixed', errors='
```

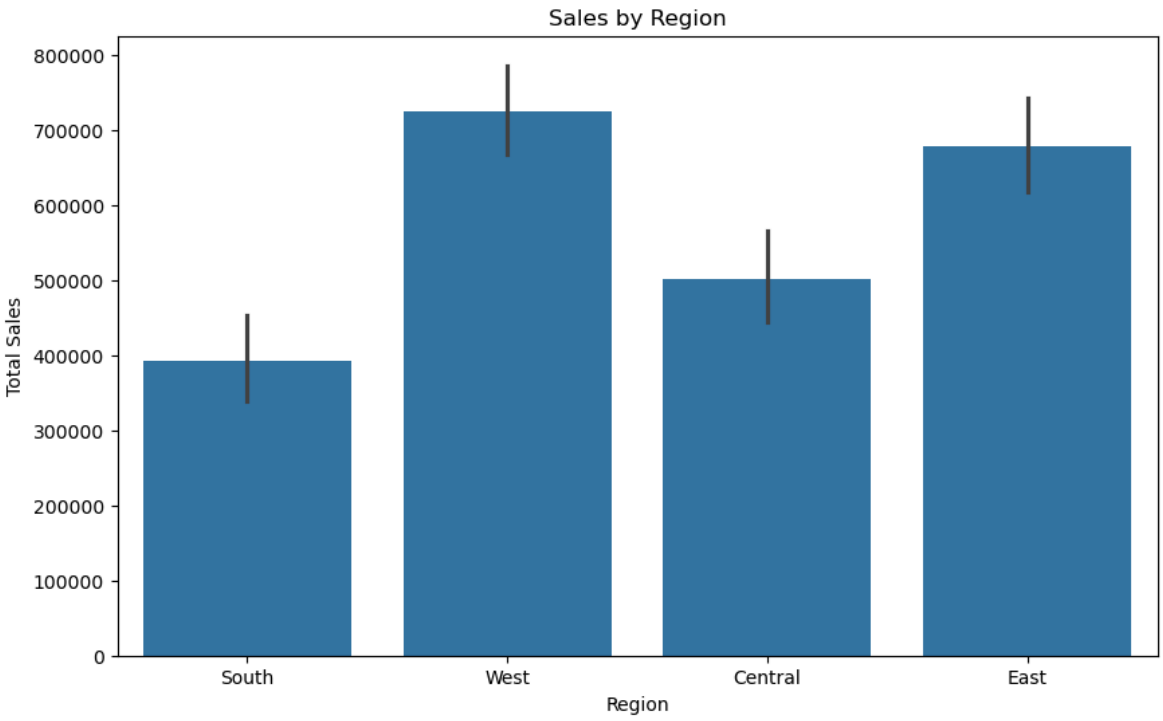
In [7]:
```
# Group by date and sum sales
# Group by date and sum sales
sales_trend = data.groupby(data['Order Date'].dt.to_period('M'))['Sales'].sum()

# Plotting the sales trend over time
plt.figure(figsize=(12, 6))
sales_trend.plot()
plt.title('Sales Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.grid()
plt.show()
```

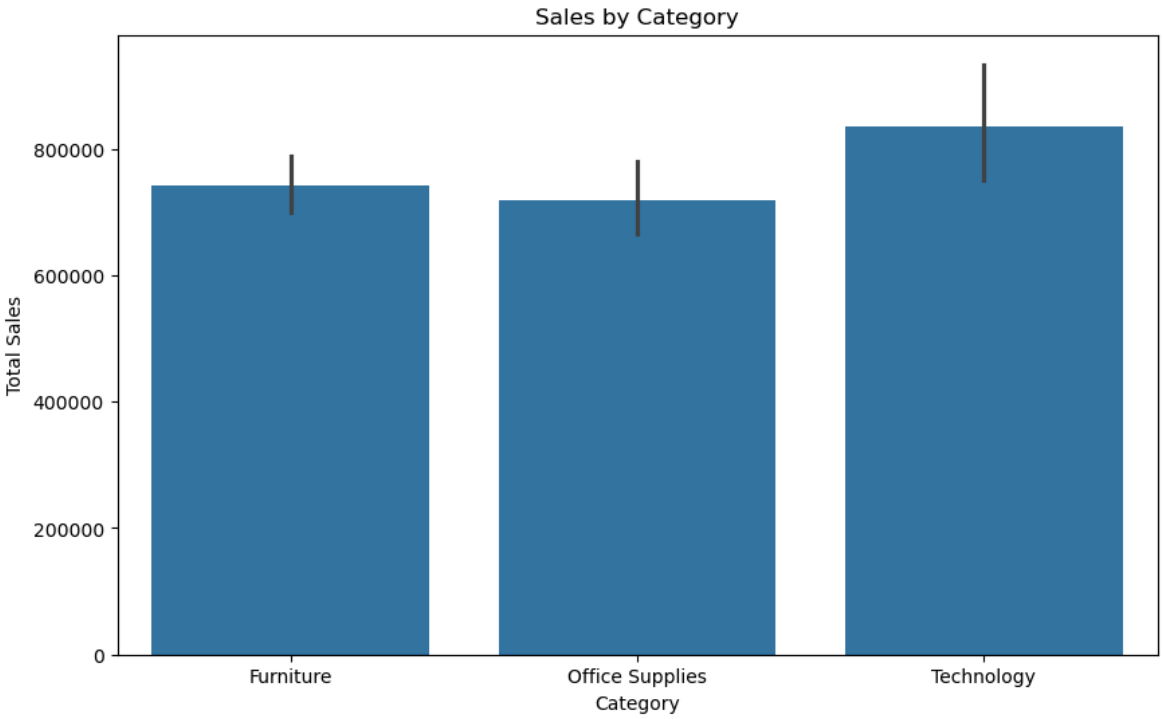


In [8]:
```
# Scatter plot for Profit vs. Discount
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Discount', y='Profit')
plt.title('Profit vs. Discount')
plt.xlabel('Discount')
plt.ylabel('Profit')
plt.grid()
plt.show()
```



In [9]:
```
# Bar plot for Sales by Region
plt.figure(figsize=(10, 6))
sns.barplot(x='Region', y='Sales', data=data, estimator=sum)
plt.title('Sales by Region')
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.show()

# Bar plot for Sales by Category
plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Sales', data=data, estimator=sum)
plt.title('Sales by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.show()
```





In [10]:
```
# Select features and target variable
X = data[['Profit', 'Discount']]
y = data['Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [11]:
```
# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 700271.8880636953
R-squared: -0.18549666591248481
```

In [12]:
```
# Coefficients of the model
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print("Model Coefficients:\n", coefficients)

# Insights
print("\nInsights and Recommendations:")
print("1. For every unit increase in Profit, Sales increase by approximately {:.
print("2. For every unit increase in Discount, Sales decrease by approximately {
print("3. Consider optimizing discount rates to maximize profit while maintainin
```

```
Model Coefficients:
           Coefficient
Profit       1.588871
Discount   257.714994

Insights and Recommendations:
1. For every unit increase in Profit, Sales increase by approximately 1.59.
2. For every unit increase in Discount, Sales decrease by approximately 257.71.
3. Consider optimizing discount rates to maximize profit while maintaining sales.
```

In [ ]: