In [4]:
```
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('/Users/nikitachoudhary/Desktop/ecommerce_customer
```

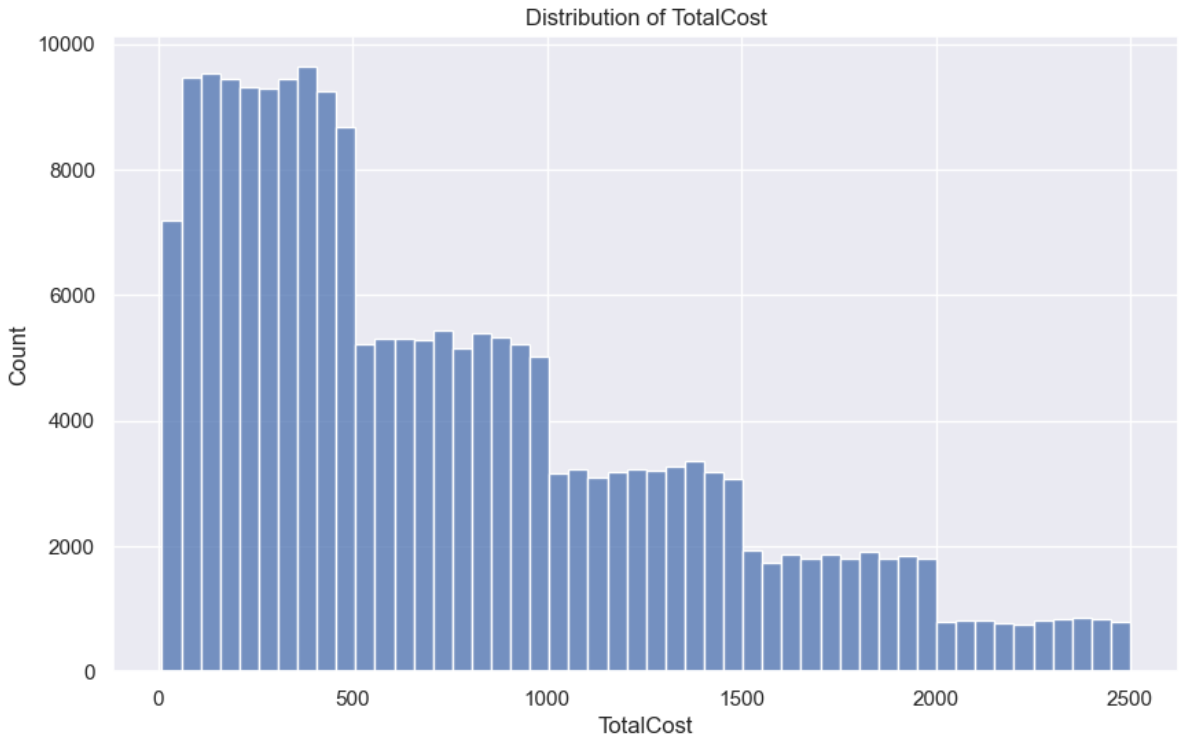
In [5]:
```
df.head()
```

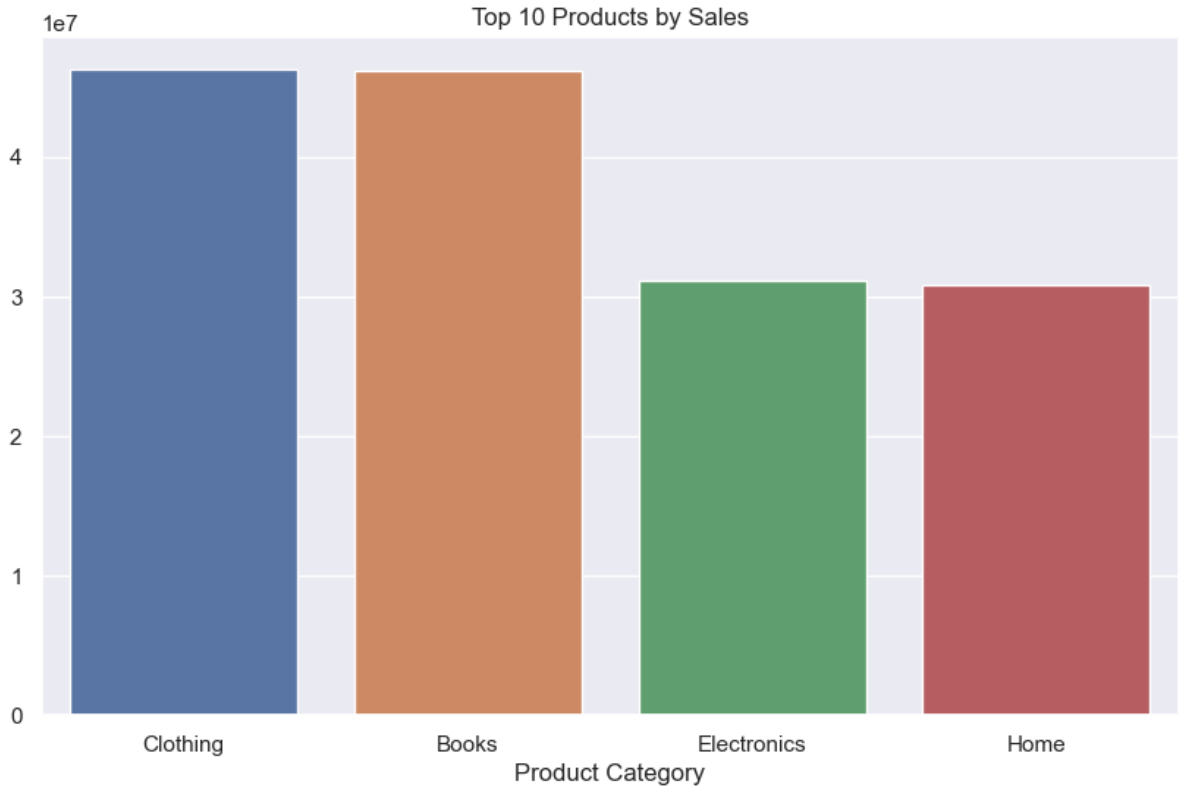
Out[5]:

	Customer ID	Purchase Date	Product Category	Product Price	Quantity	Total Purchase Amount	Payment Method	Customer Age	Retu
0	46251	2020-09-08 09:38:32	Electronics	12	3	740	Credit Card	37	
1	46251	2022-03-05 12:56:35	Home	468	4	2739	PayPal	37	
2	46251	2022-05-23 18:18:01	Home	288	2	3196	PayPal	37	
3	46251	2020-11-12 13:13:29	Clothing	196	1	3509	PayPal	37	
4	13593	2020-11-27 17:55:11	Home	449	1	3452	Credit Card	49	

In [11]:
```
# Data Cleaning
df.dropna(inplace=True)
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])
df['TotalCost'] = df['Quantity'] * df['Product Price']
```

In [12]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the distribution of TotalCost
sns.set()
plt.figure(figsize=(10, 6))
sns.histplot(df['TotalCost'], bins=50)
plt.title('Distribution of TotalCost')
plt.show()

# Plot the top 10 products by sales
top_products = df.groupby('Product Category')['TotalCost'].sum().so
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.index, y=top_products.values)
plt.title('Top 10 Products by Sales')
plt.show()
```



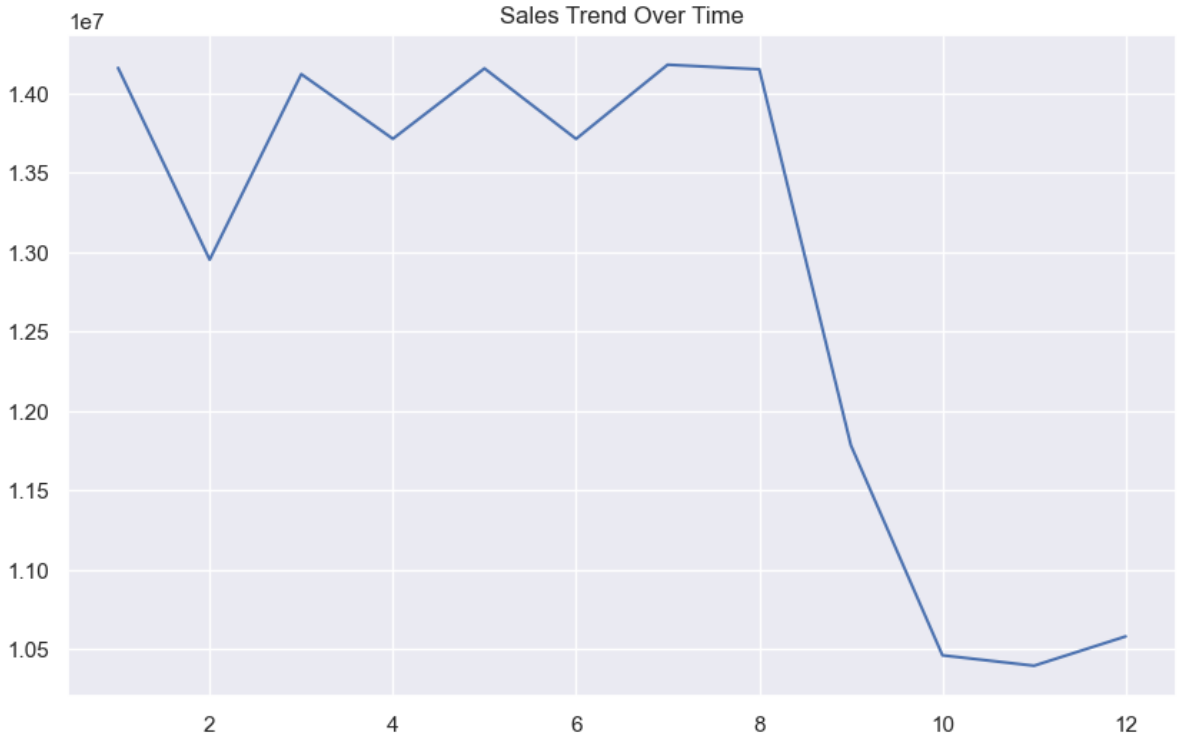


In [19]:
```python
from sklearn.cluster import KMeans

# Select relevant columns for customer segmentation
customer_data = df[['Customer ID', 'TotalCost', 'Quantity']]

# Perform K-Means clustering
kmeans = KMeans(n_clusters=5)
customer_data['Cluster'] = kmeans.fit_predict(customer_data)
```

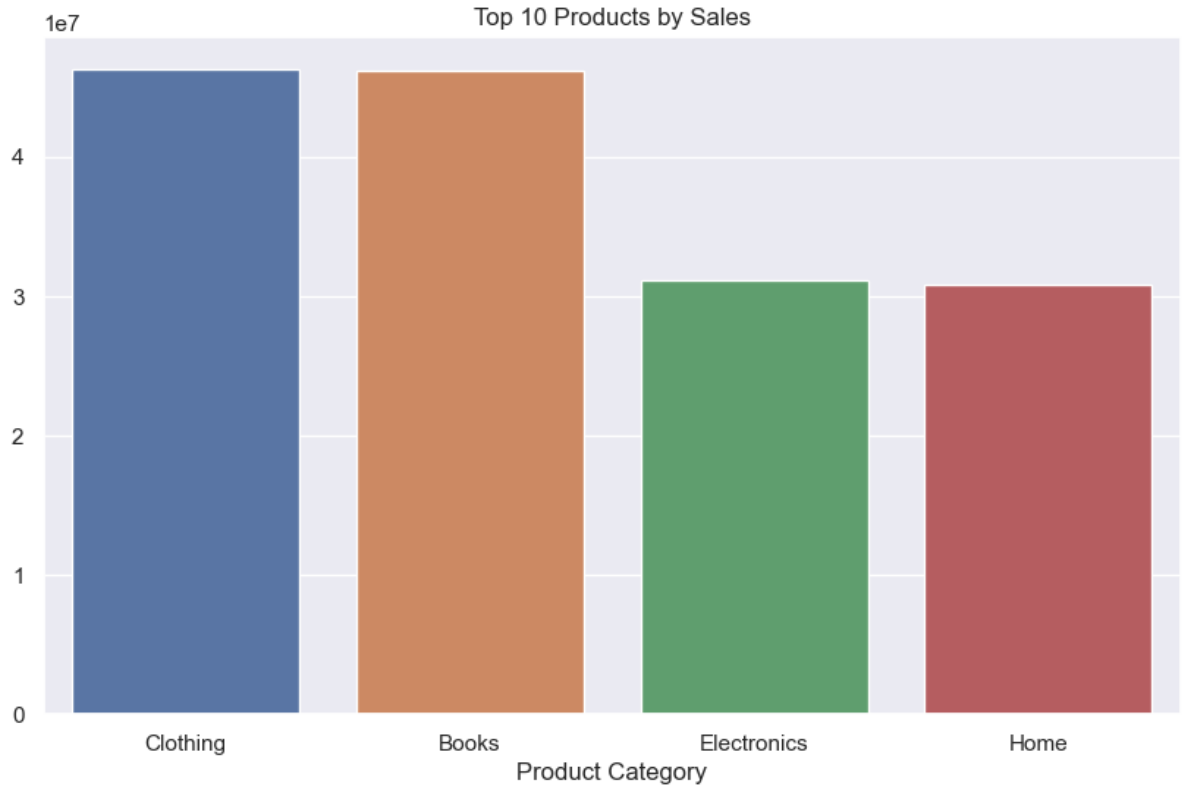
```
# Plot the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x='TotalCost', y='Quantity', hue='Cluster', data=cu
plt.title('Customer Segmentation')
plt.show()
```

```
/Users/nikitachoudhary/anaconda3/lib/python3.11/site-packages/skle
arn/cluster/_kmeans.py:1412: FutureWarning: The default value of `
n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/var/folders/ms/9x1dj4pd79s4p94sfht6dzn00000gn/T/ipykernel_10071/2
712809805.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  customer_data['Cluster'] = kmeans.fit_predict(customer_data)
```

In [15]:
```
import matplotlib.pyplot as plt

# Plot the sales trend over time
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])
df['Month'] = df['Purchase Date'].dt.month
sales_trend = df.groupby('Month')['TotalCost'].sum()
plt.figure(figsize=(10, 6))
plt.plot(sales_trend.index, sales_trend.values)
plt.title('Sales Trend Over Time')
plt.show()
```



In [16]:
```
# Plot the top 10 products by sales
top_products = df.groupby('Product Category')['TotalCost'].sum().so
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.index, y=top_products.values)
plt.title('Top 10 Products by Sales')
plt.show()
```



In [17]:
```
from statsmodels.tsa.arima.model import ARIMA

# Select the sales data for time series analysis
sales_data = df.groupby('Purchase Date')['TotalCost'].sum()

# Perform ARIMA modeling
model = ARIMA(sales_data, order=(1, 1, 1))
model_fit = model.fit()
print(model_fit.summary())
```

```
/Users/nikitachoudhary/anaconda3/lib/python3.11/site-packages/stat
smodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and
so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/nikitachoudhary/anaconda3/lib/python3.11/site-packages/stat
smodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and
so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/nikitachoudhary/anaconda3/lib/python3.11/site-packages/stat
smodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has
been provided, but it has no associated frequency information and
```

```
so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)

                               SARIMAX Results
==============================================================================
============
Dep. Variable:              TotalCost   No. Observations:
202223
Model:                 ARIMA(1, 1, 1)   Log Likelihood
-1577790.718
Date:                Fri, 11 Apr 2025   AIC
3155587.436
Time:                        02:05:04   BIC
3155618.087
Sample:                             0   HQIC
3155596.453
                                - 202223
Covariance Type:                  opg
==============================================================================
============
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------------
ar.L1         -0.0010      0.002     -0.444      0.657      -0.005
0.003
ma.L1         -1.0000    4.2e-05  -2.38e+04      0.000      -1.000
-1.000
sigma2      3.503e+05   1112.546    314.865      0.000    3.48e+05
3.52e+05
==============================================================================
=================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
26901.01
Prob(Q):                              1.00   Prob(JB):
0.00
Heteroskedasticity (H):               1.01   Skew:
0.89
Prob(H) (two-sided):                  0.51   Kurtosis:
2.99
==============================================================================
=================

Warnings:
[1] Covariance matrix calculated using the outer product of gradie
nts (complex-step).
```

In [25]:
```
from scipy.stats import ttest_ind

# Select two groups for hypothesis testing
group1 = df[df['Churn'] == 0]['TotalCost']
group2 = df[df['Churn'] == 1]['TotalCost']

# Perform t-test
t_stat, p_val = ttest_ind(group1, group2)
print('T-Statistic:', t_stat)
print('P-Value:', p_val)
```

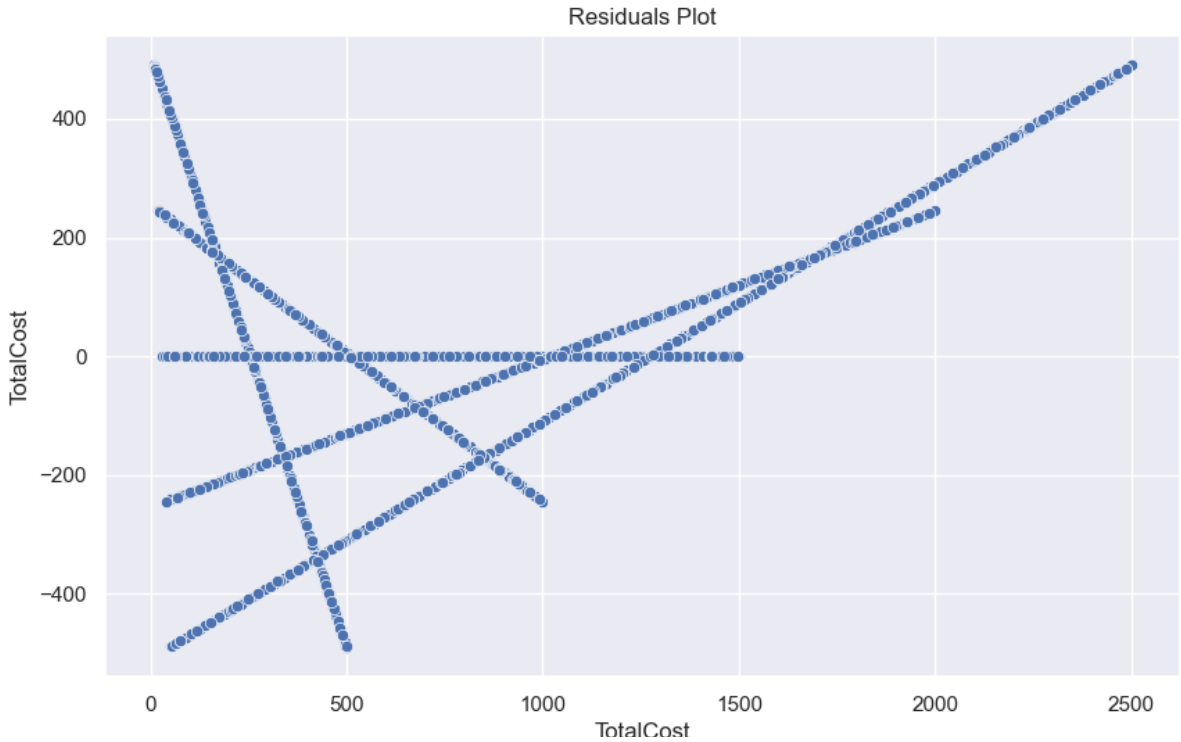
```
T-Statistic: 1.6183595167139375
P-Value: 0.10558669932571442
```

In [22]:
```
import seaborn as sns

# Select relevant columns for correlation analysis
corr_data = df[['TotalCost', 'Quantity', 'Product Price']]

# Plot the correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(corr_data.corr(), annot=True)
plt.title('Correlation Matrix')
plt.show()
```

In [23]:
```
from sklearn.linear_model import LinearRegression

# Select relevant columns for regression analysis
```

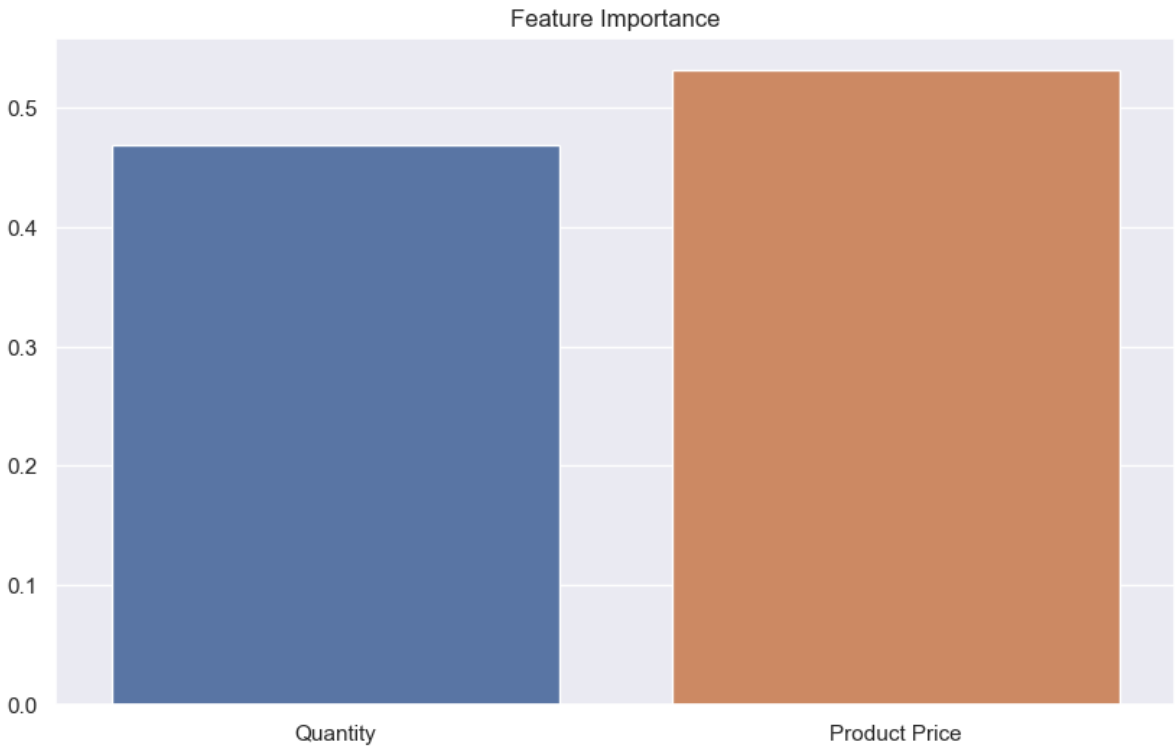
```
X = df[['Quantity', 'Product Price']]
y = df['TotalCost']

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Print the coefficients
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Evaluate the model
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)

# Plot the residuals
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=residuals)
plt.title('Residuals Plot')
plt.show()
```

```
Coefficients: [254.81706543   3.00210656]
Intercept: -765.1391941824811
Mean Squared Error: 39774.78629165224
```



In [24]:
```
from sklearn.ensemble import RandomForestRegressor

# Create and fit the random forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)

# Plot the feature importance
feature_importance = model.feature_importances_
plt.figure(figsize=(10, 6))
sns.barplot(x=X.columns, y=feature_importance)
plt.title('Feature Importance')
plt.show()
```

```
Mean Squared Error: 0.0
```



In [ ]: