```
In [2]: 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 [3]:
```python
data = pd.read_csv('sales.csv')
data
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	
0	32298	CA-2012-124891	31/7/2012	31/7/2012	Same Day	RH-19495	Rick Hansen	Consumer	N
1	26341	IN-2013-77878	5/2/2013	7/2/2013	Second Class	JR-16210	Justin Ritter	Corporate	Wc
2	25330	IN-2013-71249	17/10/2013	18/10/2013	First Class	CR-12730	Craig Reiter	Consumer	
3	13524	ES-2013-1579342	28/1/2013	30/1/2013	First Class	KM-16375	Katherine Murray	Home Office	
4	47221	SG-2013-4320	5/11/2013	6/11/2013	Same Day	RH-9495	Rick Hansen	Consumer	
...	...	...	...	...	...	...	...	...	
51285	29002	IN-2014-62366	19/6/2014	19/6/2014	Same Day	KE-16420	Katrina Edelman	Corporate	
51286	35398	US-2014-102288	20/6/2014	24/6/2014	Standard Class	ZC-21910	Zuschuss Carroll	Consumer	
51287	40470	US-2013-155768	2/12/2013	2/12/2013	Same Day	LB-16795	Laurel Beltran	Home Office	
51288	9596	MX-2012-140767	18/2/2012	22/2/2012	Standard Class	RB-19795	Ross Baird	Home Office	
51289	6147	MX-2012-134460	22/5/2012	26/5/2012	Second Class	MC-18100	Mick Crebagga	Consumer	

51290 rows × 24 columns

In [4]:
```
data['Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

```
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_31052\2887490760.py:1:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the d
efault) was specified. This may lead to inconsistently parsed dates! Speci
fy a format to ensure consistent parsing.
  data['Order Date'] = pd.to_datetime(data['Order Date'])
C:\Users\Aditya Kudva\AppData\Local\Temp\ipykernel_31052\2887490760.py:2:
UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the d
efault) was specified. This may lead to inconsistently parsed dates! Speci
fy a format to ensure consistent parsing.
  data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

In [5]:
```
data['Order Year'] = data['Order Date'].dt.year
data['Order Month'] = data['Order Date'].dt.month
```

In [6]:
```
features = ['Order Year', 'Order Month', 'Ship Mode', 'Segment', 'Market',
X = pd.get_dummies(data[features])  # Convert categorical variables into du
y = data['Sales']
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
X_train, X_test, y_train, y_test
```

Out[7]:
```
(       Order Year  Order Month  Quantity  Discount  Shipping Cost  \
 33162        2012            6         9     0.000           4.21
 42206        2014           11         5     0.000           1.77
 25603        2012            5         4     0.000           7.82
 28126        2013            7         2     0.200           6.41
 17208        2014            1         3     0.100          16.05
 ...           ...          ...       ...       ...            ...
 11284        2013            9         2     0.500          28.64
 44732        2012            4         2     0.500           1.28
 38158        2014            4         3     0.000           2.69
 860          2012            6         4     0.002         219.53
 15795        2012            7         2     0.000          18.30

        Ship Mode_First Class  Ship Mode_Same Day  Ship Mode_Second Clas
s  \
 33162                     0                   0
0
 42206                     0                   0
0
```

In [8]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[8]:
```
▾ LinearRegression
LinearRegression()
```

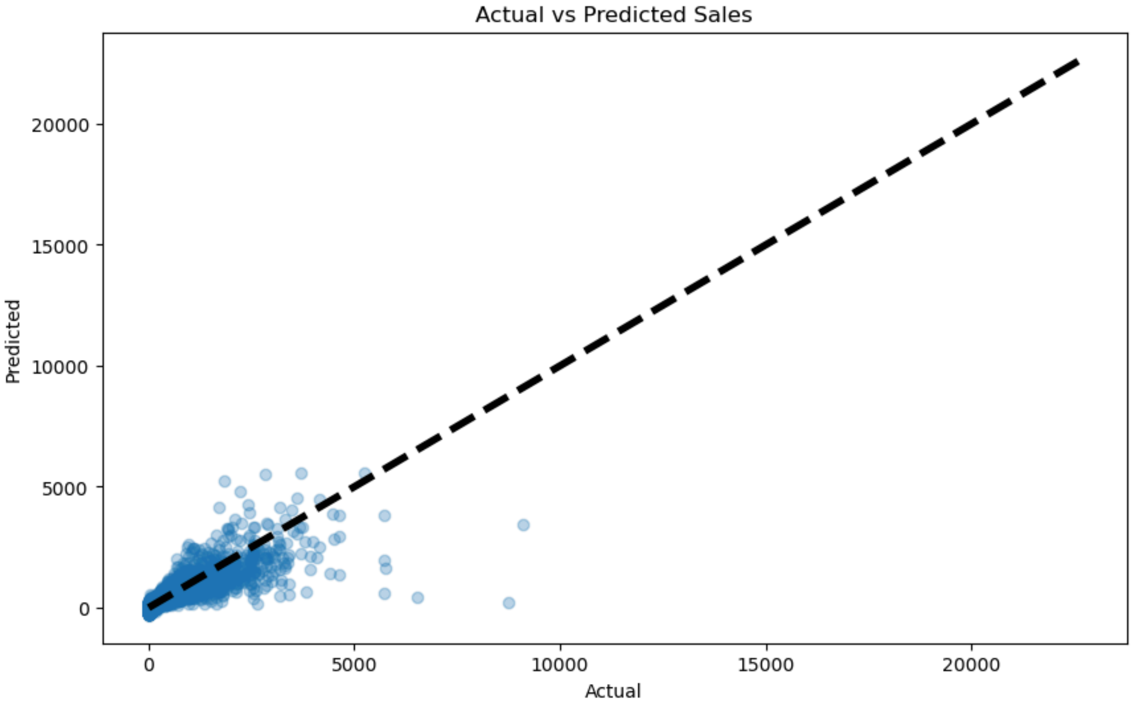
In [9]:

```
y_pred = model.predict(X_test)
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}')
```

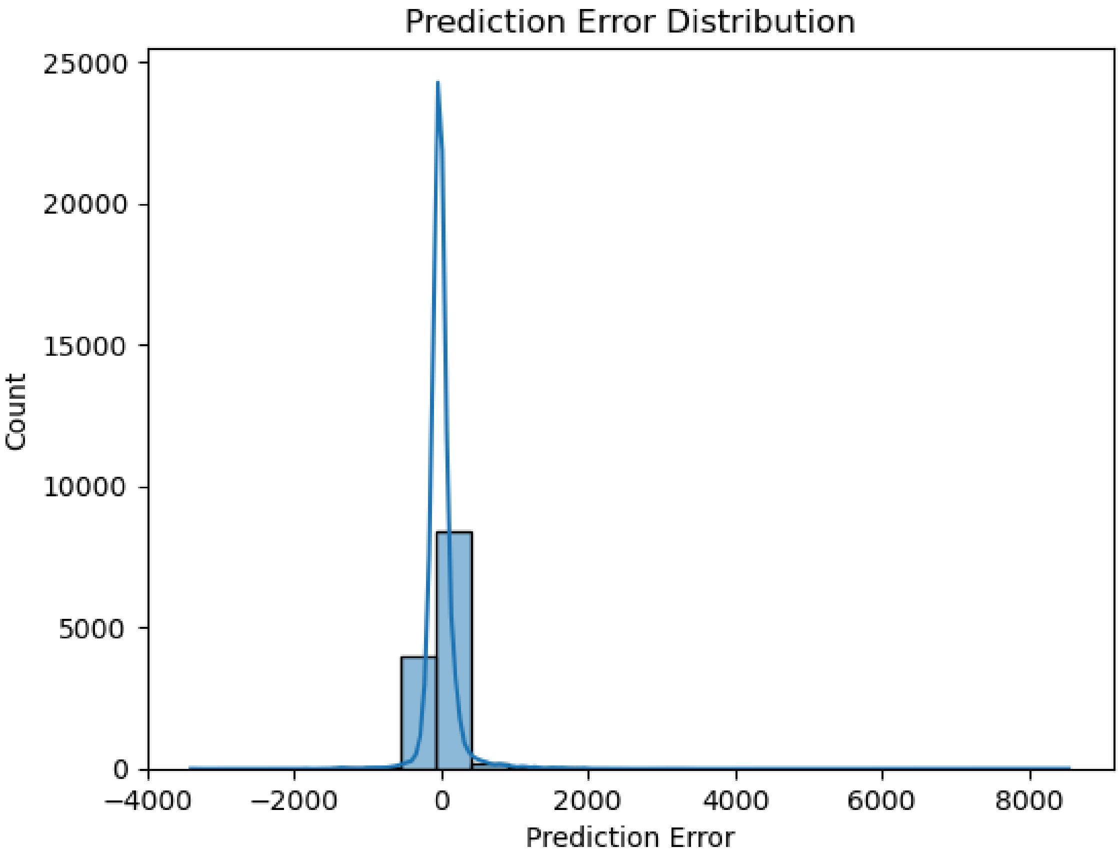
```
Mean Squared Error: 62981.46
R^2 Score: 0.71
```

In [10]:

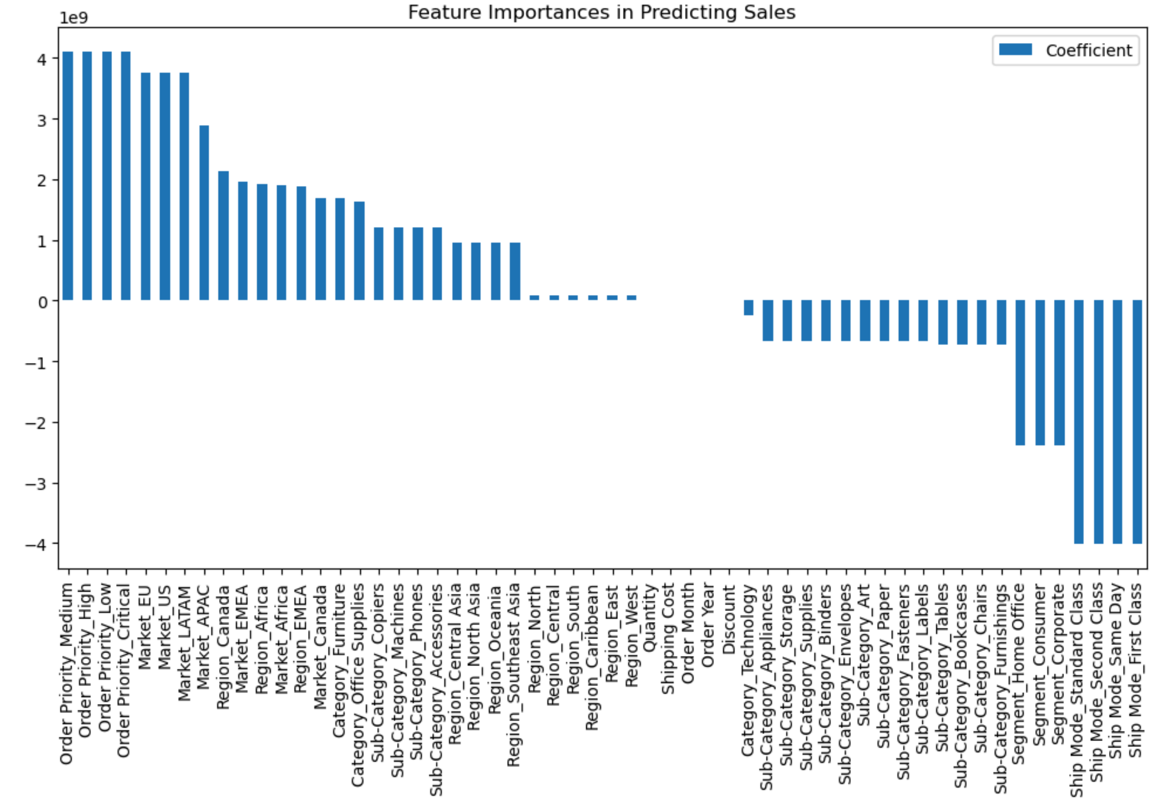
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Sales')
plt.show()
```



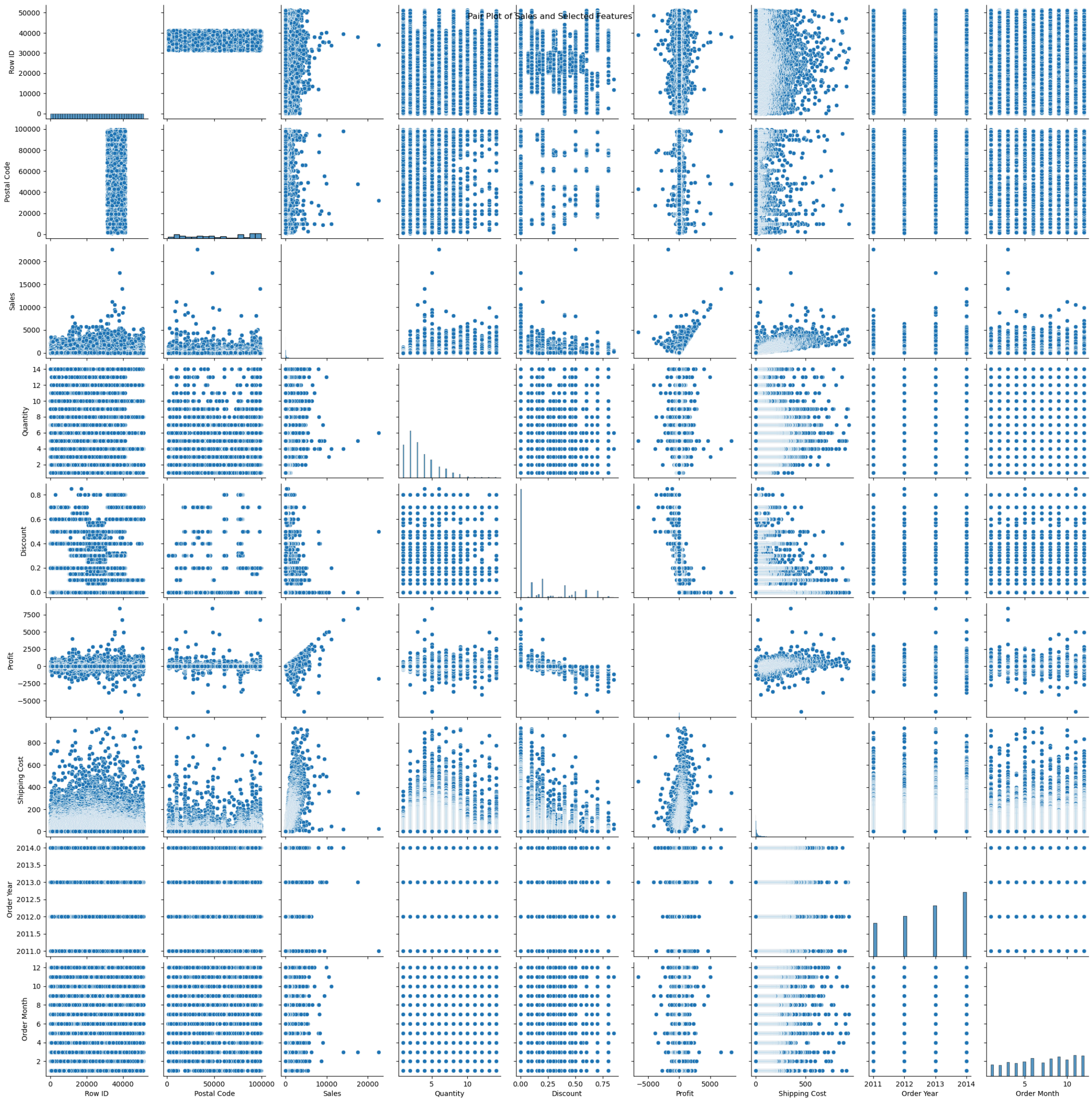
In [11]:
```
errors = y_test - y_pred
sns.histplot(errors, bins=25, kde=True)
plt.xlabel('Prediction Error')
plt.title('Prediction Error Distribution')
plt.show()
```



In [12]:
```
feature_importance = pd.DataFrame(model.coef_, index=X.columns, columns=['C
feature_importance.sort_values(by='Coefficient', ascending=False, inplace=T
feature_importance.plot(kind='bar', figsize=(12, 6))
plt.title('Feature Importances in Predicting Sales')
plt.show()
```



In [13]:
```
sns.pairplot(data)
plt.suptitle('Pair Plot of Sales and Selected Features', verticalalignment=
plt.show()
```



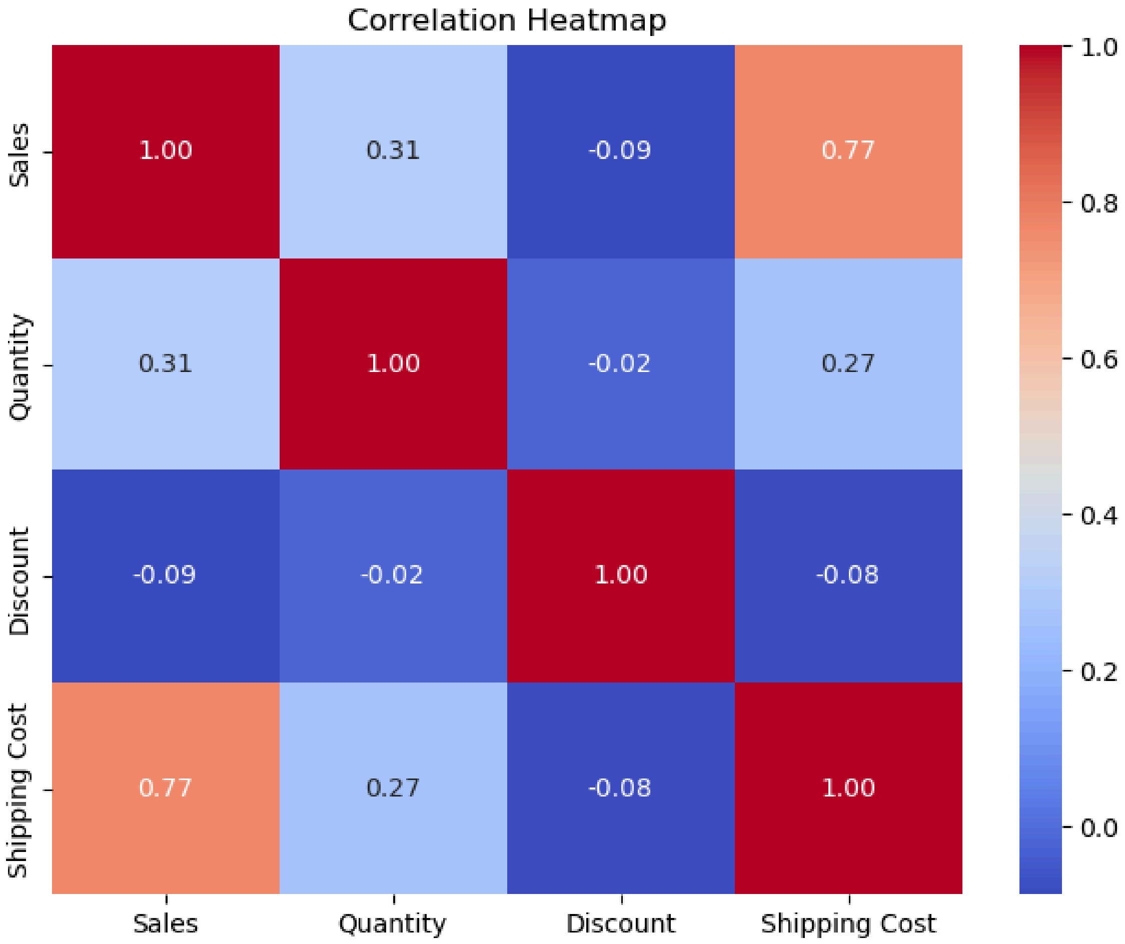
In [14]:
```
correlation_matrix = data[['Sales', 'Quantity', 'Discount', 'Shipping Cost'
correlation_matrix
```

Out[14]:

	Sales	Quantity	Discount	Shipping Cost
Sales	1.000000	0.313577	-0.086722	0.768073
Quantity	0.313577	1.000000	-0.019875	0.272649
Discount	-0.086722	-0.019875	1.000000	-0.079056
Shipping Cost	0.768073	0.272649	-0.079056	1.000000

In [15]:

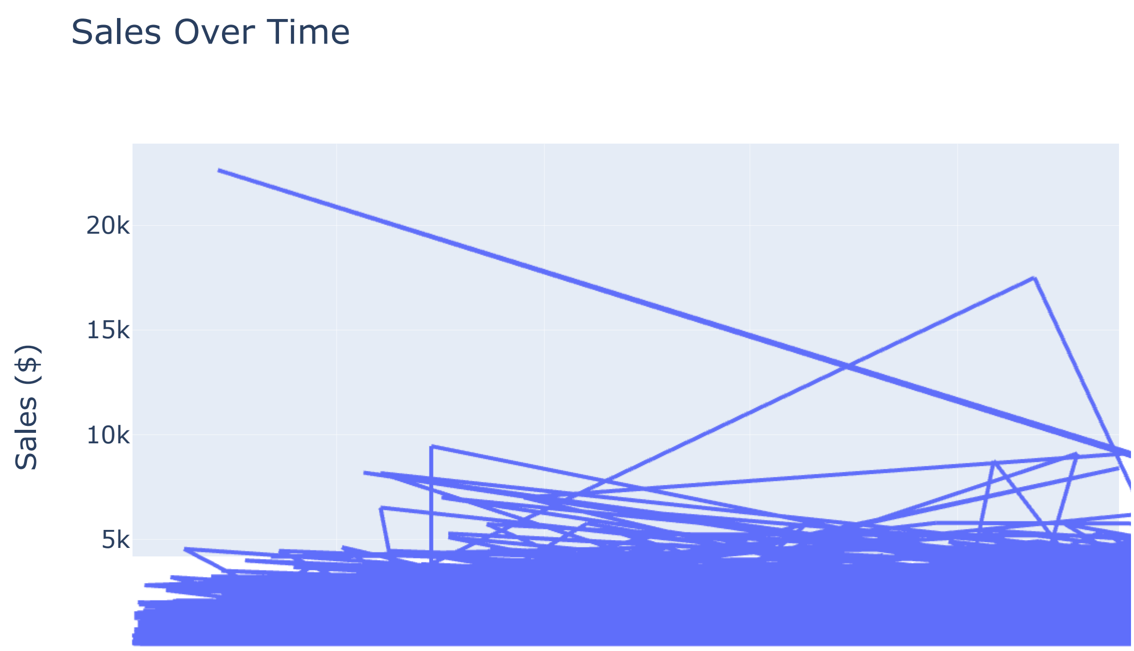
```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [16]:

```
import plotly.express as px
```

In [17]:
```
fig = px.line(data, x='Order Date', y='Sales', title='Sales Over Time', labe
fig.update_xaxes(rangeslider_visible=True)
fig.show()
```



In [18]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [19]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

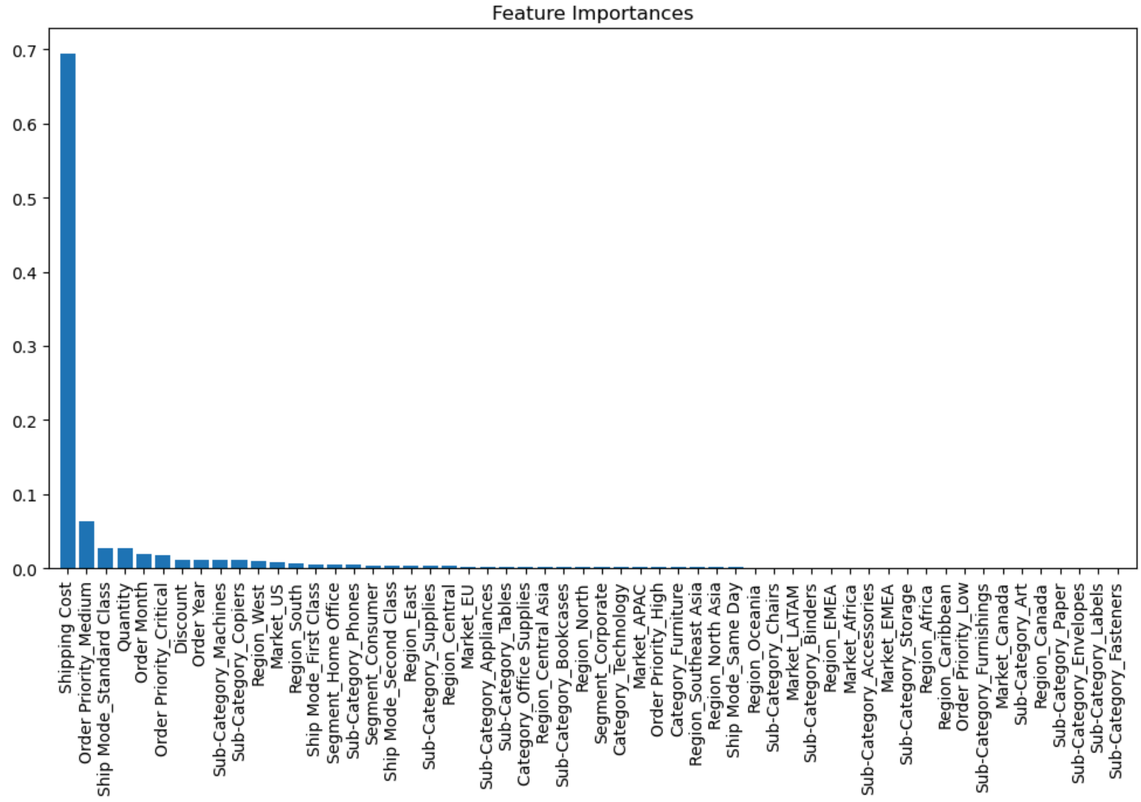
Out[19]:
```
▾          RandomForestRegressor

RandomForestRegressor(random_state=42)
```

In [20]:
```
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
```

In [21]:

```
plt.figure(figsize=(12, 6))
plt.title('Feature Importances')
plt.bar(range(X_train.shape[1]), importances[indices], align='center')
plt.xticks(range(X_train.shape[1]), X_train.columns[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```



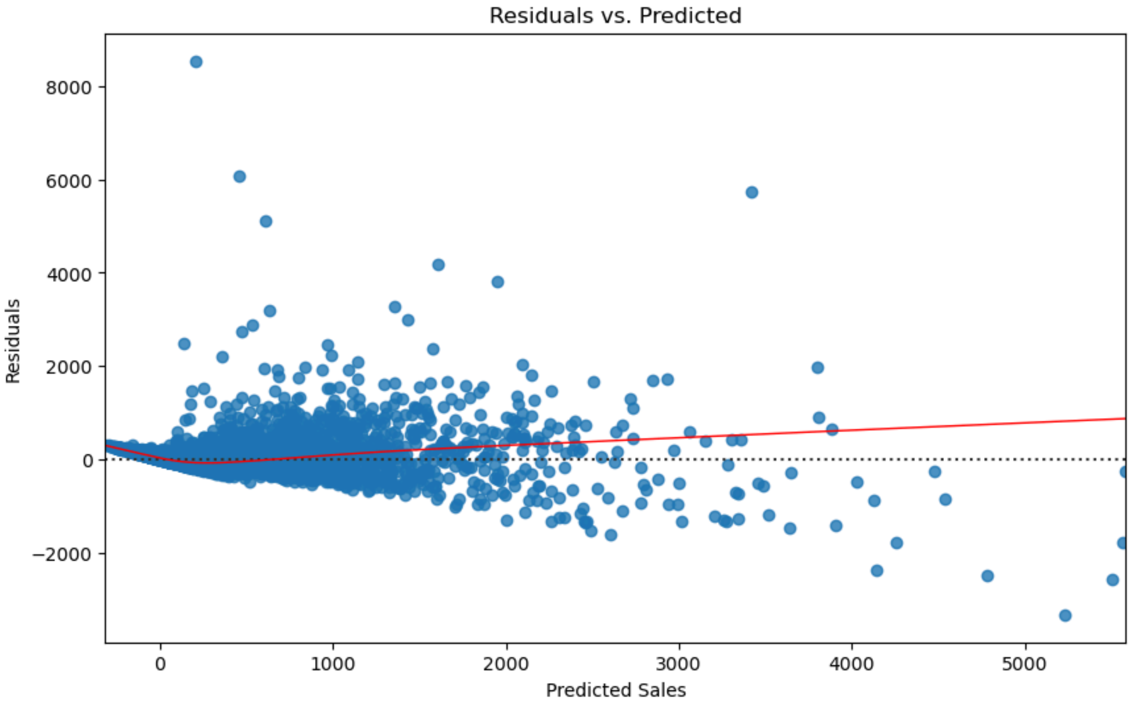
In [22]:

```
residuals = y_test - y_pred
residuals
```

Out[22]:

```
49728   -22.035868
45547    92.248218
15664    43.484829
40561   -34.142389
49426    43.925078
           ...
46156    55.910974
18754   -43.388948
29983   -32.241150
24864   -84.280607
12001    73.362399
Name: Sales, Length: 12823, dtype: float64
```

In [23]:
```
plt.figure(figsize=(10, 6))
sns.residplot(x=y_pred, y=residuals, lowess=True, line_kws={'color': 'red',
plt.title('Residuals vs. Predicted')
plt.xlabel('Predicted Sales')
plt.ylabel('Residuals')
plt.show()
```



In [ ]: