In [4]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"
path=r'C:\Users\saisr\Downloads\retail_price.csv'
data = pd.read_csv(path)
print(data.head())
```

```
  product_id product_category_name  month_year  qty  total_price  \
0       bed1        bed_bath_table  01-05-2017    1        45.95
1       bed1        bed_bath_table  01-06-2017    3       137.85
2       bed1        bed_bath_table  01-07-2017    6       275.70
3       bed1        bed_bath_table  01-08-2017    4       183.80
4       bed1        bed_bath_table  01-09-2017    2        91.90

   freight_price  unit_price  product_name_lenght  product_description_lenght  \
0      15.100000       45.95                   39                         161
1      12.933333       45.95                   39                         161
2      14.840000       45.95                   39                         161
3      14.287500       45.95                   39                         161
4      15.100000       45.95                   39                         161

   product_photos_qty  ...  comp_1  ps1        fp1      comp_2  ps2  \
0                   2  ...    89.9  3.9  15.011897  215.000000  4.4
1                   2  ...    89.9  3.9  14.769216  209.000000  4.4
2                   2  ...    89.9  3.9  13.993833  205.000000  4.4
3                   2  ...    89.9  3.9  14.656757  199.509804  4.4
4                   2  ...    89.9  3.9  18.776522  163.398710  4.4

         fp2  comp_3  ps3        fp3  lag_price
0   8.760000   45.95  4.0  15.100000      45.90
1  21.322000   45.95  4.0  12.933333      45.95
2  22.195932   45.95  4.0  14.840000      45.95
3  19.412885   45.95  4.0  14.287500      45.95
4  24.324687   45.95  4.0  15.100000      45.95

[5 rows x 30 columns]
```

In [5]:
```
print(data.isnull().sum())
```

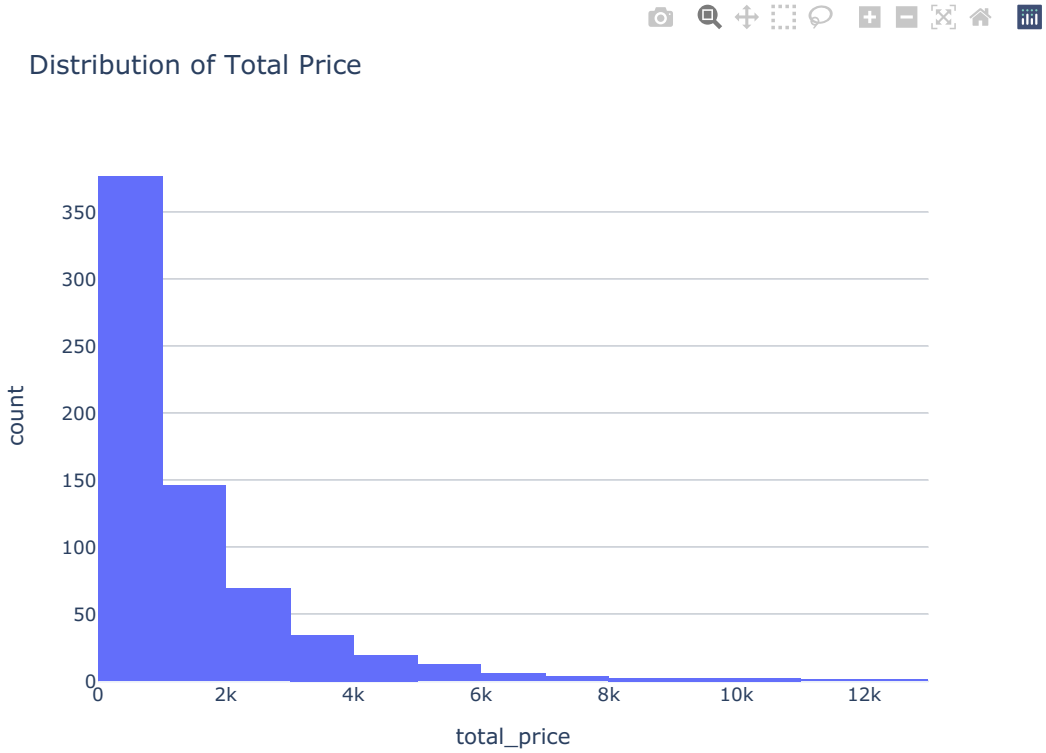
```
product_id                    0
product_category_name         0
month_year                    0
qty                           0
total_price                   0
freight_price                 0
unit_price                    0
product_name_lenght           0
product_description_lenght    0
product_photos_qty            0
product_weight_g              0
product_score                 0
customers                     0
weekday                       0
weekend                       0
holiday                       0
month                         0
year                          0
s                             0
volume                        0
comp_1                        0
ps1                           0
fp1                           0
comp_2                        0
ps2                           0
fp2                           0
comp_3                        0
ps3                           0
fp3                           0
lag_price                     0
dtype: int64
```

In [6]:
```
print(data.describe())
```

```
              qty   total_price  freight_price  unit_price  \
count  676.000000    676.000000     676.000000  676.000000
mean    14.495562   1422.708728      20.682270  106.496800
std     15.443421   1700.123100      10.081817   76.182972
min      1.000000     19.900000       0.000000   19.900000
25%      4.000000    333.700000      14.761912   53.900000
50%     10.000000    807.890000      17.518472   89.900000
75%     18.000000   1887.322500      22.713558  129.990000
max    122.000000  12095.000000      79.760000  364.000000

       product_name_lenght  product_description_lenght  product_photos_qty  \
count           676.000000                  676.000000          676.000000
mean             48.720414                  767.399408            1.994083
std               9.420715                  655.205015            1.420473
min              29.000000                  100.000000            1.000000
25%              40.000000                  339.000000            1.000000
50%              51.000000                  501.000000            1.500000
75%              57.000000                  903.000000            2.000000
max              60.000000                 3006.000000            8.000000

       product_weight_g  product_score   customers  ...       comp_1  \
count        676.000000     676.000000  676.000000  ...  676.000000
mean        1847.498521       4.085503   81.028107  ...   79.452054
std         2274.808483       0.232021   62.055560  ...   47.933358
min          100.000000       3.300000    1.000000  ...   19.900000
25%          348.000000       3.900000   34.000000  ...   49.910000
50%          950.000000       4.100000   62.000000  ...   69.900000
75%         1850.000000       4.200000  116.000000  ...  104.256549
max         9750.000000       4.500000  339.000000  ...  349.900000

              ps1         fp1      comp_2         ps2         fp2      comp_3  \
count  676.000000  676.000000  676.000000  676.000000  676.000000  676.000000
mean     4.159467   18.597610   92.930079    4.123521   18.620644   84.182642
std      0.121652    9.406537   49.481269    0.207189    6.424174   47.745789
min      3.700000    0.095439   19.900000    3.300000    4.410000   19.900000
25%      4.100000   13.826429   53.900000    4.100000   14.485000   53.785714
50%      4.200000   16.618984   89.990000    4.200000   16.811765   59.900000
75%      4.200000   19.732500  117.888889    4.200000   21.665238   99.990000
max      4.500000   57.230000  349.900000    4.400000   57.230000  255.610000

              ps3         fp3   lag_price
count  676.000000  676.000000  676.000000
mean     4.002071   17.965007  107.399684
std      0.233292    5.533256   76.974657
min      3.500000    7.670000   19.850000
25%      3.900000   15.042727   55.668750
50%      4.000000   16.517110   89.900000
75%      4.100000   19.447778  129.990000
max      4.400000   57.230000  364.000000

[8 rows x 27 columns]
```

In [7]:

```
fig = px.histogram(data,
                   x='total_price',
                   nbins=20,
                   title='Distribution of Total Price')
fig.show()
```



In [8]:
```
fig = px.box(data,
             y='unit_price',
             title='Box Plot of Unit Price')
fig.show()
```

In [9]:
```
fig = px.scatter(data,
                 x='qty',
                 y='total_price',
                 title='Quantity vs Total Price', trendline="ols")
fig.show()
```

In [10]:
```
fig = px.bar(data, x='product_category_name',
             y='total_price',
             title='Average Total Price by Product Category')
fig.show()
```

In [11]:
```
fig = px.box(data, x='weekday',
             y='total_price',
             title='Box Plot of Total Price by Weekday')
fig.show()
```

In [18]:
```
fig = px.box(data, x='holiday',
             y='total_price',
             title='Box Plot of Total Price by Holiday')
fig.show()
```

In [12]:
```
correlation_matrix = data.corr()
fig = go.Figure(go.Heatmap(x=correlation_matrix.columns,
                           y=correlation_matrix.columns,
                           z=correlation_matrix.values))
fig.update_layout(title='Correlation Heatmap of Numerical Features')
fig.show()
```

```
C:\Users\saisr\AppData\Local\Temp\ipykernel_19340\1131337044.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
```

In [13]:
```
data['comp_price_diff'] = data['unit_price'] - data['comp_1']

avg_price_diff_by_category = data.groupby('product_category_name')['comp_price_diff'].mean().reset_index()

fig = px.bar(avg_price_diff_by_category,
             x='product_category_name',
             y='comp_price_diff',
             title='Average Competitor Price Difference by Product Category')
fig.update_layout(
    xaxis_title='Product Category',
    yaxis_title='Average Competitor Price Difference'
)
fig.show()
```

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

X = data[['qty', 'unit_price', 'comp_1',
          'product_score', 'comp_price_diff']]
```

```
y = data['total_price']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42)
```

In [16]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[16]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [17]:
```
y_pred = model.predict(X_test)

fig = go.Figure()
fig.add_trace(go.Scatter(x=y_test, y=y_pred, mode='markers',
                         marker=dict(color='blue'),
                         name='Predicted vs. Actual Retail Price'))
fig.add_trace(go.Scatter(x=[min(y_test), max(y_test)], y=[min(y_test), max(y_test)],
                         mode='lines',
                         marker=dict(color='red'),
                         name='Ideal Prediction'))
fig.update_layout(
    title='Predicted vs. Actual Retail Price',
    xaxis_title='Actual Retail Price',
    yaxis_title='Predicted Retail Price'
)
fig.show()
```

In [ ]:
```

```