In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"
```

In [2]:

```python
df = pd.read_csv('retail_price.csv')
```

In [3]:

```python
df.head()
```

Out[3]:

	product_id	product_category_name	month_year	qty	total_price	freight_price	unit_price	product_name_lenght	product_description_lengh
0	bed1	bed_bath_table	01-05-2017	1	45.95	15.100000	45.95	39	16
1	bed1	bed_bath_table	01-06-2017	3	137.85	12.933333	45.95	39	16
2	bed1	bed_bath_table	01-07-2017	6	275.70	14.840000	45.95	39	16
3	bed1	bed_bath_table	01-08-2017	4	183.80	14.287500	45.95	39	16
4	bed1	bed_bath_table	01-09-2017	2	91.90	15.100000	45.95	39	16

5 rows × 30 columns

In [4]:

```python
df.tail()
```

Out[4]:

	product_id	product_category_name	month_year	qty	total_price	freight_price	unit_price	product_name_lenght	product_description_le
671	bed5	bed_bath_table	01-05-2017	1	215.00	8.760000	215.000000	56	
672	bed5	bed_bath_table	01-06-2017	10	2090.00	21.322000	209.000000	56	
673	bed5	bed_bath_table	01-07-2017	59	12095.00	22.195932	205.000000	56	
674	bed5	bed_bath_table	01-08-2017	52	10375.00	19.412885	199.509804	56	
675	bed5	bed_bath_table	01-09-2017	32	5222.36	24.324687	163.398710	56	

5 rows × 30 columns

In [5]:

```python
df.describe()
```

Out[5]:

	qty	total_price	freight_price	unit_price	product_name_lenght	product_description_lenght	product_photos_qty	product_we
count	676.000000	676.000000	676.000000	676.000000	676.000000	676.000000	676.000000	676.(
mean	14.495562	1422.708728	20.682270	106.496800	48.720414	767.399408	1.994083	1847.4
std	15.443421	1700.123100	10.081817	76.182972	9.420715	655.205015	1.420473	2274.8
min	1.000000	19.900000	0.000000	19.900000	29.000000	100.000000	1.000000	100.(
25%	4.000000	333.700000	14.761912	53.900000	40.000000	339.000000	1.000000	348.(
50%	10.000000	807.890000	17.518472	89.900000	51.000000	501.000000	1.500000	950.(
75%	18.000000	1887.322500	22.713558	129.990000	57.000000	903.000000	2.000000	1850.(
max	122.000000	12095.000000	79.760000	364.000000	60.000000	3006.000000	8.000000	9750.(

8 rows × 27 columns

In [6]:

```python
df.columns
```

Out[6]:

```
Index(['product_id', 'product_category_name', 'month_year', 'qty',
       'total_price', 'freight_price', 'unit_price', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_score', 'customers', 'weekday', 'weekend', 'holiday', 'month',
       'year', 's', 'volume', 'comp_1', 'ps1', 'fp1', 'comp_2', 'ps2', 'fp2',
       'comp_3', 'ps3', 'fp3', 'lag_price'],
      dtype='object')
```

In [7]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 676 entries, 0 to 675
Data columns (total 30 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  676 non-null    object 
 1   product_category_name       676 non-null    object 
 2   month_year                  676 non-null    object 
 3   qty                         676 non-null    int64  
 4   total_price                 676 non-null    float64
 5   freight_price               676 non-null    float64
 6   unit_price                  676 non-null    float64
 7   product_name_lenght         676 non-null    int64  
 8   product_description_lenght  676 non-null    int64  
 9   product_photos_qty          676 non-null    int64  
 10  product_weight_g            676 non-null    int64  
 11  product_score               676 non-null    float64
 12  customers                   676 non-null    int64  
 13  weekday                     676 non-null    int64  
 14  weekend                     676 non-null    int64  
 15  holiday                     676 non-null    int64  
 16  month                       676 non-null    int64  
 17  year                        676 non-null    int64  
 18  s                           676 non-null    float64
 19  volume                      676 non-null    int64  
 20  comp_1                      676 non-null    float64
 21  ps1                         676 non-null    float64
 22  fp1                         676 non-null    float64
 23  comp_2                      676 non-null    float64
 24  ps2                         676 non-null    float64
 25  fp2                         676 non-null    float64
 26  comp_3                      676 non-null    float64
 27  ps3                         676 non-null    float64
 28  fp3                         676 non-null    float64
 29  lag_price                   676 non-null    float64
dtypes: float64(15), int64(12), object(3)
memory usage: 158.6+ KB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
df.dtypes
```

Out[9]:

```
product_id                     object
product_category_name          object
month_year                     object
qty                             int64
total_price                   float64
freight_price                 float64
unit_price                    float64
product_name_lenght             int64
product_description_lenght      int64
product_photos_qty              int64
product_weight_g                int64
product_score                 float64
customers                       int64
weekday                         int64
weekend                         int64
holiday                         int64
month                           int64
year                            int64
s                             float64
volume                          int64
comp_1                        float64
ps1                           float64
fp1                           float64
comp_2                        float64
ps2                           float64
fp2                           float64
comp_3                        float64
ps3                           float64
fp3                           float64
lag_price                     float64
dtype: object
```

In [10]:

```
df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
df.nunique()
```

Out[11]:

```
product_id                     52
product_category_name           9
month_year                     20
qty                            66
total_price                   573
freight_price                 653
unit_price                    280
product_name_lenght            24
product_description_lenght     46
product_photos_qty              7
product_weight_g               45
product_score                  11
customers                      94
weekday                         4
weekend                         3
holiday                         5
month                          12
year                            2
s                             450
volume                         40
comp_1                         88
ps1                             9
fp1                           179
comp_2                        123
ps2                            10
fp2                           242
comp_3                        105
ps3                             9
fp3                           229
lag_price                     307
dtype: int64
```

In [12]:

```
df['product_category_name'].unique()
```

Out[12]:

```
array(['bed_bath_table', 'garden_tools', 'consoles_games',
       'health_beauty', 'cool_stuff', 'perfumery',
       'computers_accessories', 'watches_gifts', 'furniture_decor'],
      dtype=object)
```

In [13]:

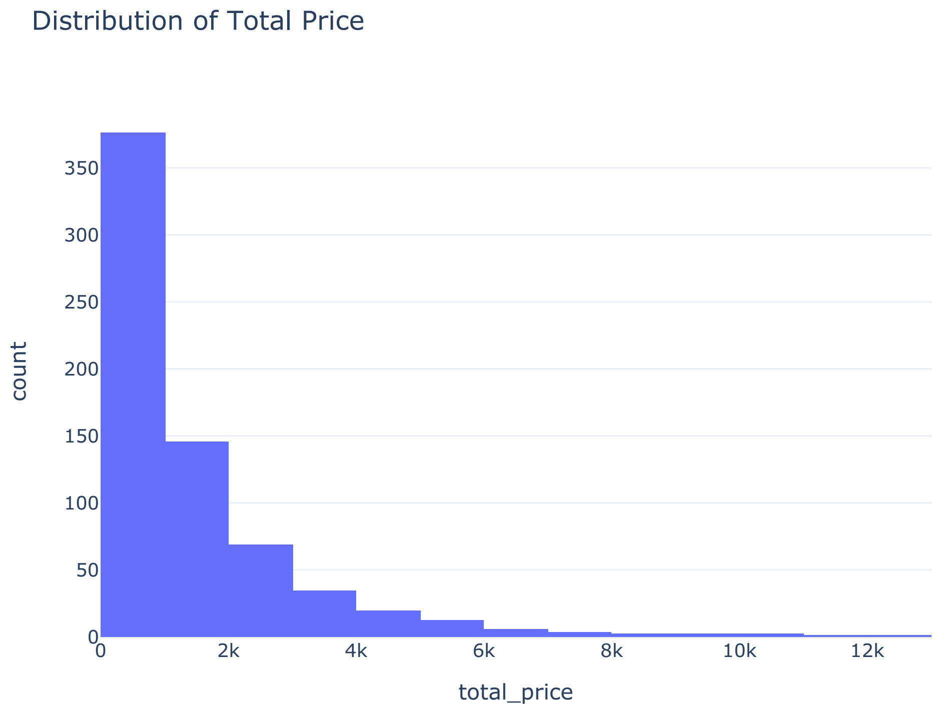
```
df['product_category_name'].value_counts()
```

Out[13]:

```
product_category_name
garden_tools             160
health_beauty            130
watches_gifts            103
computers_accessories     69
bed_bath_table            61
cool_stuff                57
furniture_decor           48
perfumery                 26
consoles_games            22
Name: count, dtype: int64
```

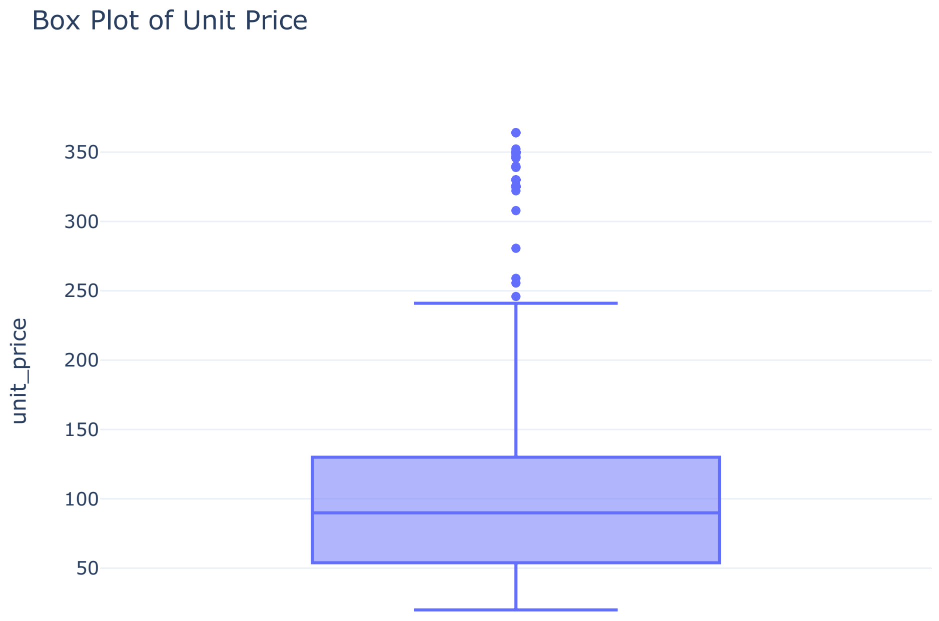
In [16]:

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

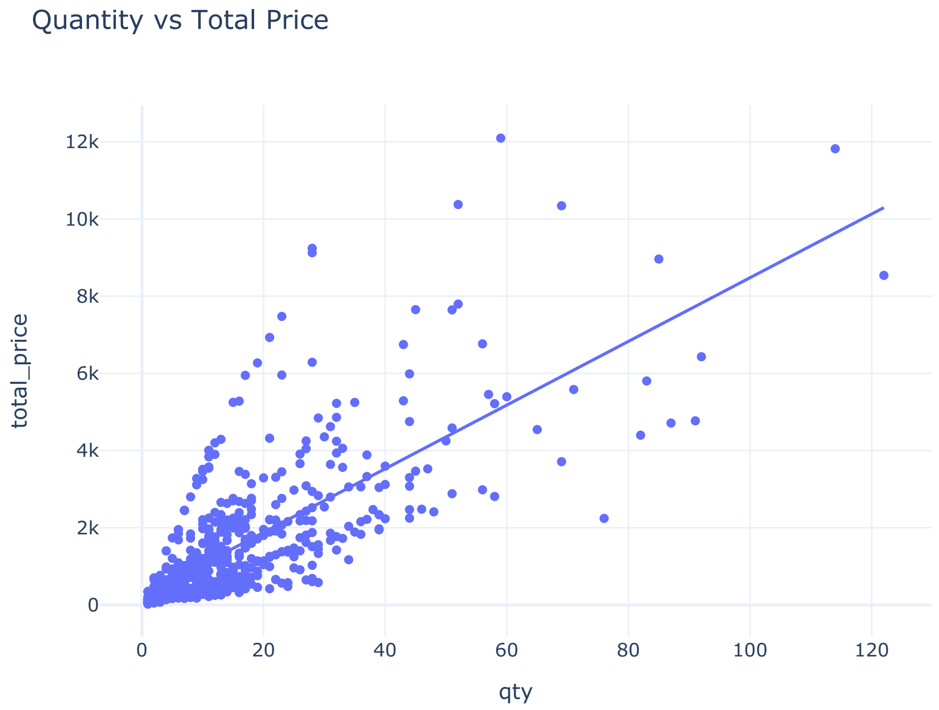


In [17]:

```
fig = px.box(df,
             y='unit_price',
             title='Box Plot of Unit Price')
fig.show()
```



```
In [18]: fig = px.scatter(df,
                          x='qty',
                          y='total_price',
                          title='Quantity vs Total Price', trendline="ols")
         fig.show()
```



```
In [19]: fig = px.bar(df, x='product_category_name',
                      y='total_price',
                      title='Average Total Price by Product Category')
         fig.show()
```

In [20]:

```
fig = px.box(df, x='weekday',
             y='total_price',
             title='Box Plot of Total Price by Weekday')
fig.show()
```

In [21]:

```
fig = px.box(df, x='holiday',
             y='total_price',
             title='Box Plot of Total Price by Holiday')
fig.show()
```

In [23]:

```
df = df.drop('product_id', axis = 1)
```

In [24]:

```
df.head()
```

Out[24]:

	product_category_name	month_year	qty	total_price	freight_price	unit_price	product_name_lenght	product_description_lenght	product_p
0	bed_bath_table	01-05-2017	1	45.95	15.100000	45.95	39	161	
1	bed_bath_table	01-06-2017	3	137.85	12.933333	45.95	39	161	
2	bed_bath_table	01-07-2017	6	275.70	14.840000	45.95	39	161	
3	bed_bath_table	01-08-2017	4	183.80	14.287500	45.95	39	161	
4	bed_bath_table	01-09-2017	2	91.90	15.100000	45.95	39	161	

5 rows × 29 columns

In [27]:

```
from sklearn import preprocessing

# label_encoder object knows
# how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['product_category_name']= label_encoder.fit_transform(df['product_category_name'])
```

In [30]:

```
df = df.drop('month_year', axis = 1)
```

In [31]:

```
df.corr()
```

Out[31]:

	product_category_name	qty	total_price	freight_price	unit_price	product_name_lenght	product_description_
product_category_name	1.000000	-0.050217	0.033230	-0.057583	0.257830	-0.034963	0.
qty	-0.050217	1.000000	0.749605	-0.135521	-0.103432	0.079973	-0.
total_price	0.033230	0.749605	1.000000	0.025848	0.409001	-0.002594	0.
freight_price	-0.057583	-0.135521	0.025848	1.000000	0.203659	0.013398	0.
unit_price	0.257830	-0.103432	0.409001	0.203659	1.000000	-0.170613	0.
product_name_lenght	-0.034963	0.079973	-0.002594	0.013398	-0.170613	1.000000	0.
product_description_lenght	0.144115	-0.022749	0.175376	0.423219	0.280176	0.124510	1.
product_photos_qty	0.112418	0.128515	0.157945	-0.200990	0.076990	0.131951	0.
product_weight_g	-0.113299	-0.034301	0.060092	0.670689	0.112958	-0.044050	0.
product_score	0.166527	-0.004028	0.036119	0.199468	0.042162	0.163520	0.
customers	0.248260	0.441547	0.386389	0.088261	0.043391	0.082239	0.
weekday	0.019489	0.030918	0.018798	-0.016132	-0.011949	0.023797	-0.
weekend	-0.009921	-0.075118	-0.053788	0.030275	-0.000042	-0.018183	-0.
holiday	0.003266	0.211610	0.136558	-0.081518	0.012573	-0.014317	0.
month	0.003729	-0.005129	-0.029918	-0.028336	-0.004249	-0.004250	-0.
year	0.034818	0.058562	0.082140	0.076595	-0.068072	-0.035479	0.
s	-0.047425	0.411001	0.334500	-0.109359	-0.016552	-0.080830	0.
volume	-0.201671	0.049827	-0.088726	0.122097	-0.197233	0.329476	-0.
comp_1	-0.083843	-0.033570	0.144426	-0.013969	0.317113	-0.344125	-0.
ps1	0.461251	-0.047883	0.058941	-0.053927	0.197425	0.019053	0.
fp1	-0.492407	-0.053477	-0.006729	0.306479	-0.004518	-0.079388	0.
comp_2	0.003144	-0.027044	0.203050	-0.084208	0.466459	-0.240613	0.
ps2	0.138379	0.036633	0.113178	0.168881	0.085436	-0.055069	0.
fp2	-0.226351	-0.069855	-0.001240	0.484647	0.026601	0.016903	0.
comp_3	0.437958	-0.068522	0.121114	-0.089285	0.383780	-0.382787	0.
ps3	-0.040489	-0.074466	-0.240526	0.054627	-0.242111	0.117217	0.
fp3	0.001309	-0.086439	-0.077442	0.412115	0.019461	-0.001470	0.
lag_price	0.258267	-0.085885	0.426256	0.201143	0.994453	-0.174862	0.

28 rows × 28 columns

In [32]:

```
correlation_matrix = df.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()
```

In [36]:

```
df['comp_price_diff'] = df['unit_price'] - df['comp_1']

avg_price_diff_by_category = df.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 [37]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

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

```
y = df['total_price']

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

# Train a linear regression model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[37]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [38]:

```
X_train
```

Out[38]:

	qty	unit_price	comp_1	product_score	comp_price_diff
218	6	149.000000	149.000000	4.0	0.000000
18	20	97.588235	59.900000	4.1	37.688235
567	6	98.323333	49.910000	4.3	48.413333
408	3	58.990000	23.990000	3.9	35.000000
657	11	79.800000	119.000000	3.5	-39.200000
...	...	...	...	...	...
71	8	23.990000	23.990000	4.3	0.000000
106	2	99.990000	99.990000	4.3	0.000000
270	18	148.778571	148.778571	4.2	0.000000
435	10	96.656667	49.900000	4.1	46.756667
102	14	50.490000	50.490000	4.3	0.000000

540 rows × 5 columns

In [39]:

```
X_test
```

Out[39]:

	qty	unit_price	comp_1	product_score	comp_price_diff
641	18	174.433333	133.000000	3.8	41.433333
302	14	129.990000	99.990000	4.3	30.000000
369	1	99.900000	75.000000	4.4	24.900000
493	28	77.821429	112.000000	3.9	-34.178571
579	3	99.990000	99.990000	4.2	0.000000
...	...	...	...	...	...
51	8	51.400000	50.545161	4.2	0.854839
204	6	49.910000	49.910000	4.1	0.000000
544	28	325.892857	23.990000	4.2	301.902857
428	21	89.990000	52.406944	4.1	37.583056
247	10	159.000000	49.900000	4.2	109.100000

136 rows × 5 columns

In [40]:

```
y_train
```

Out[40]:

```
218     894.00
18     1956.00
567     589.94
408     176.97
657     876.90
        ...
71      191.92
106     199.98
270    2762.50
435     969.90
102     706.86
Name: total_price, Length: 540, dtype: float64
```

In [41]:

```
y_test
```

Out[41]:

```
641    3139.80
302    1819.86
369      99.90
493    2179.00
579     299.97
        ...
51      411.20
204     299.46
544    9125.00
428    1889.79
247    1590.00
Name: total_price, Length: 136, dtype: float64
```

In [42]:

```
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 [ ]: