Cosmetics Data Analysis

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Read Data

In [3]:

```
Cosmetics = pd.read_csv(r"C:\Users\ratho\.ipynb_checkpoints\DATA\cosmetics.csv")
Cosmetics
```

Out[3]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	1	1
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	1	1
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	1	1
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1
...	...	...	...	...	...	...	...	...
1467	Sun protect	KORRES	Yoghurt Nourishing Fluid Veil Face Sunscreen B...	35	3.9	Water, Alcohol Denat., Potassium Cetyl Phospha...	1	1
1468	Sun protect	KATE SOMERVILLE	Daily Deflector™ Waterlight Broad Spectrum SPF...	48	3.6	Water, Isododecane, Dimethicone, Butyloctyl Sa...	0	0
1469	Sun protect	VITA LIBERATA	Self Tan Dry Oil SPF 50	54	3.5	Water, Dihydroxyacetone, Glycerin, Sclerocarya...	0	0
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	Pro Light Self Tan Bronzing Mist	20	1.0	Water, Dihydroxyacetone, Propylene Glycol, PPG...	0	0
1471	Sun protect	DERMAFLASH	DERMAPROTECT Daily Defense Broad Spectrum SPF 50+	45	0.0	Visit the DERMAFLASH boutique	1	1

1472 rows × 11 columns

Data quick check

In [4]:

```
Cosmetics.head()
```

Out[4]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	1	1	1
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	1	1	1
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1	1
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	1	1	1
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1	1

In [5]:
```
Cosmetics.tail()
```

Out[5]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Norn
1467	Sun protect	KORRES	Yoghurt Nourishing Fluid Veil Face Sunscreen B...	35	3.9	Water, Alcohol Denat., Potassium Cetyl Phospha...	1	1	
1468	Sun protect	KATE SOMERVILLE	Daily Deflector™ Waterlight Broad Spectrum SPF...	48	3.6	Water, Isododecane, Dimethicone, Butyloctyl Sa...	0	0	
1469	Sun protect	VITA LIBERATA	Self Tan Dry Oil SPF 50	54	3.5	Water, Dihydroxyacetone, Glycerin, Sclerocarya...	0	0	
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	Pro Light Self Tan Bronzing Mist	20	1.0	Water, Dihydroxyacetone, Propylene Glycol, PPG...	0	0	
1471	Sun protect	DERMAFLASH	DERMAPROTECT Daily Defense Broad Spectrum SPF 50+	45	0.0	Visit the DERMAFLASH boutique	1	1	

In [6]:
```
Cosmetics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1472 entries, 0 to 1471
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Label        1472 non-null   object 
 1   Brand        1472 non-null   object 
 2   Name         1472 non-null   object 
 3   Price        1472 non-null   int64  
 4   Rank         1472 non-null   float64
 5   Ingredients  1472 non-null   object 
 6   Combination  1472 non-null   int64  
 7   Dry          1472 non-null   int64  
 8   Normal       1472 non-null   int64  
 9   Oily         1472 non-null   int64  
 10  Sensitive    1472 non-null   int64  
dtypes: float64(1), int64(6), object(4)
memory usage: 126.6+ KB
```

In [7]:
```
Cosmetics.shape
```

Out[7]:
```
(1472, 11)
```

In [8]:
```
Cosmetics.dtypes
```

Out[8]:
```
Label           object
Brand           object
Name            object
Price            int64
Rank           float64
Ingredients     object
Combination      int64
Dry              int64
Normal           int64
Oily             int64
Sensitive        int64
dtype: object
```

In [9]:
```
Cosmetics.ndim
```

Out[9]:
```
2
```

In [10]:
```
Cosmetics.nunique()
```

Out[10]:
```
Label             6
Brand           116
Name           1472
Price           146
Rank             29
Ingredients    1334
Combination       2
Dry               2
Normal            2
Oily              2
Sensitive         2
dtype: int64
```

In [11]:
```
Cosmetics.isnull()
```

Out[11]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
1467	False	False	False	False	False	False	False	False	False	False	False
1468	False	False	False	False	False	False	False	False	False	False	False
1469	False	False	False	False	False	False	False	False	False	False	False
1470	False	False	False	False	False	False	False	False	False	False	False
1471	False	False	False	False	False	False	False	False	False	False	False

1472 rows × 11 columns

In [12]:
```
Cosmetics.isnull().sum()
```

Out[12]:
```
Label          0
Brand          0
Name           0
Price          0
Rank           0
Ingredients    0
Combination    0
Dry            0
Normal         0
Oily           0
Sensitive      0
dtype: int64
```

Seperatig cateorical and numerical columns

In [13]:
```
Cat = Cosmetics.select_dtypes(include ='object').columns

num = Cosmetics.select_dtypes(exclude ='object').columns

Cat,num
```

Out[13]:
```
(Index(['Label', 'Brand', 'Name', 'Ingredients'], dtype='object'),
 Index(['Price', 'Rank', 'Combination', 'Dry', 'Normal', 'Oily', 'Sensitive'], dtype
='object'))
```

Categorical column analysis

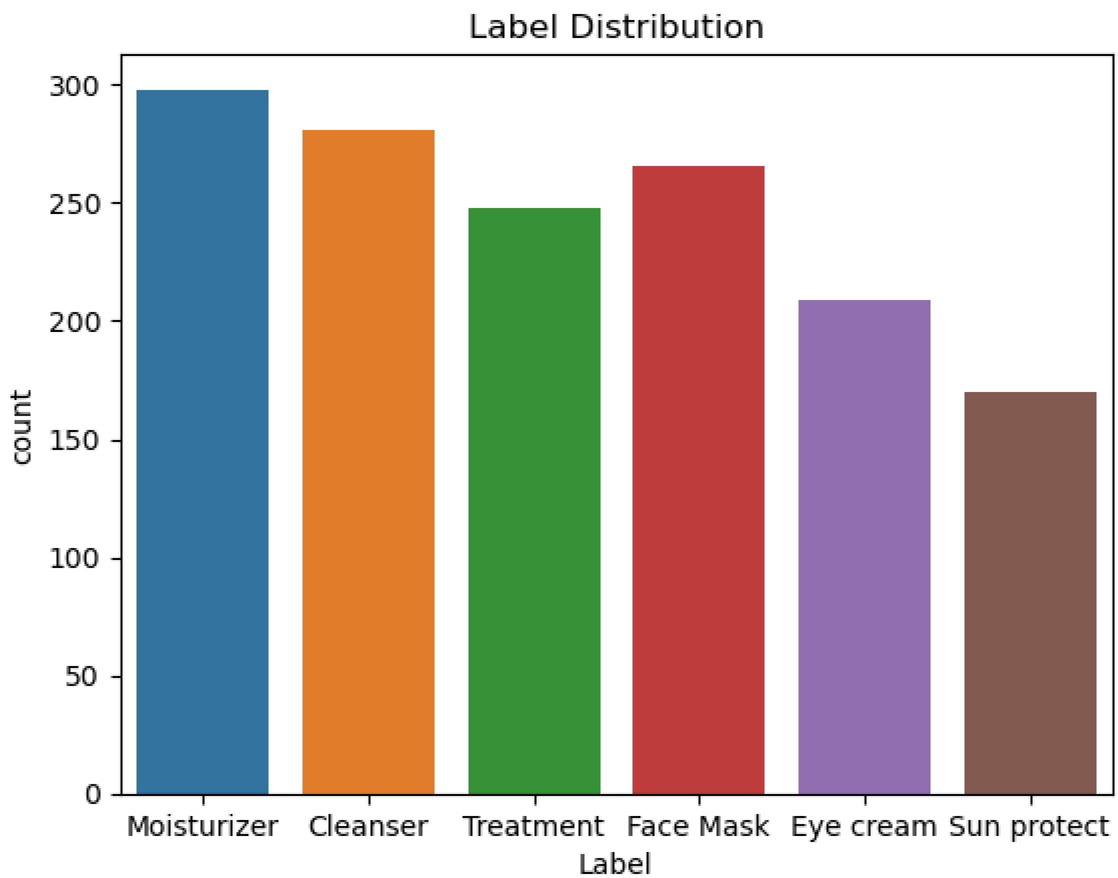
In [14]:
```
Cat
```

Out[14]: Index(['Label', 'Brand', 'Name', 'Ingredients'], dtype='object')

In [15]:
```
# Frequency Distribution
gender_counts = Cosmetics['Label'].value_counts()
print("Label Frequency:\n", gender_counts)

# Plot the Distribution
sns.countplot(x='Label', data=Cosmetics)
plt.title('Label Distribution')
plt.show()
```

```
Label Frequency:
 Moisturizer    298
Cleanser       281
Face Mask      266
Treatment      248
Eye cream      209
Sun protect    170
Name: Label, dtype: int64
```



Observations:

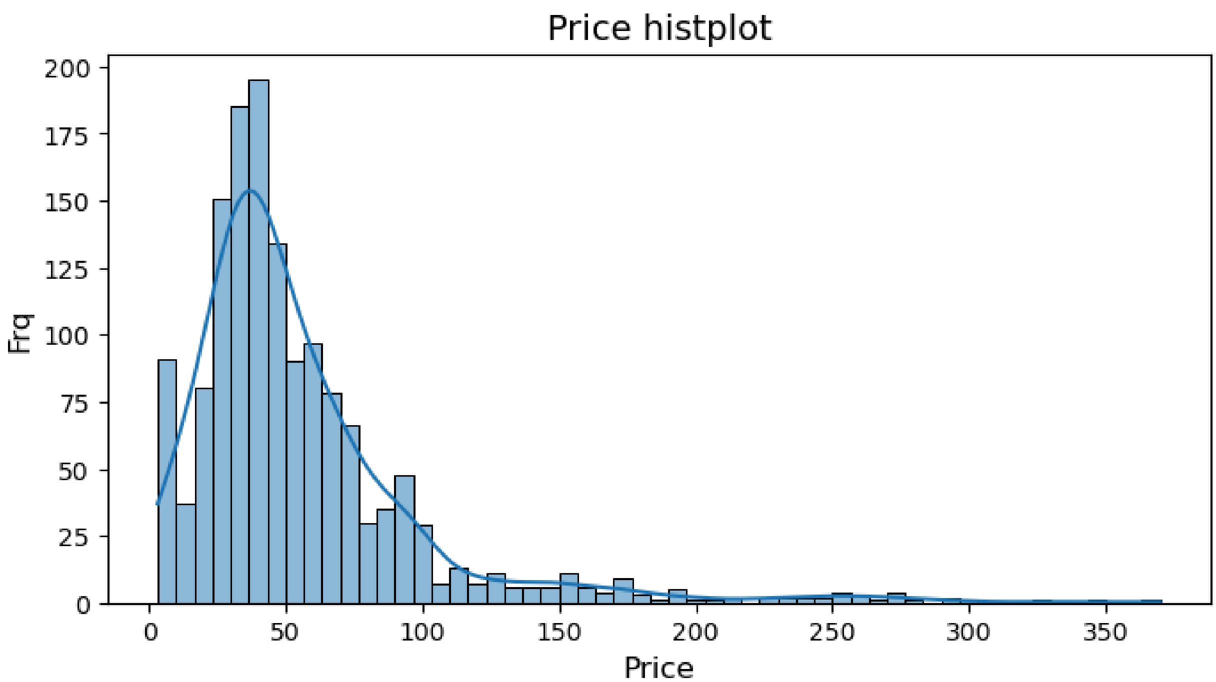
1. "Moisturizer" has the highest count (~300), while "Sun Protect" has the lowest.
2. "Eye Cream" and "Sun Protect" are underrepresented compared to other categories.

Insights:

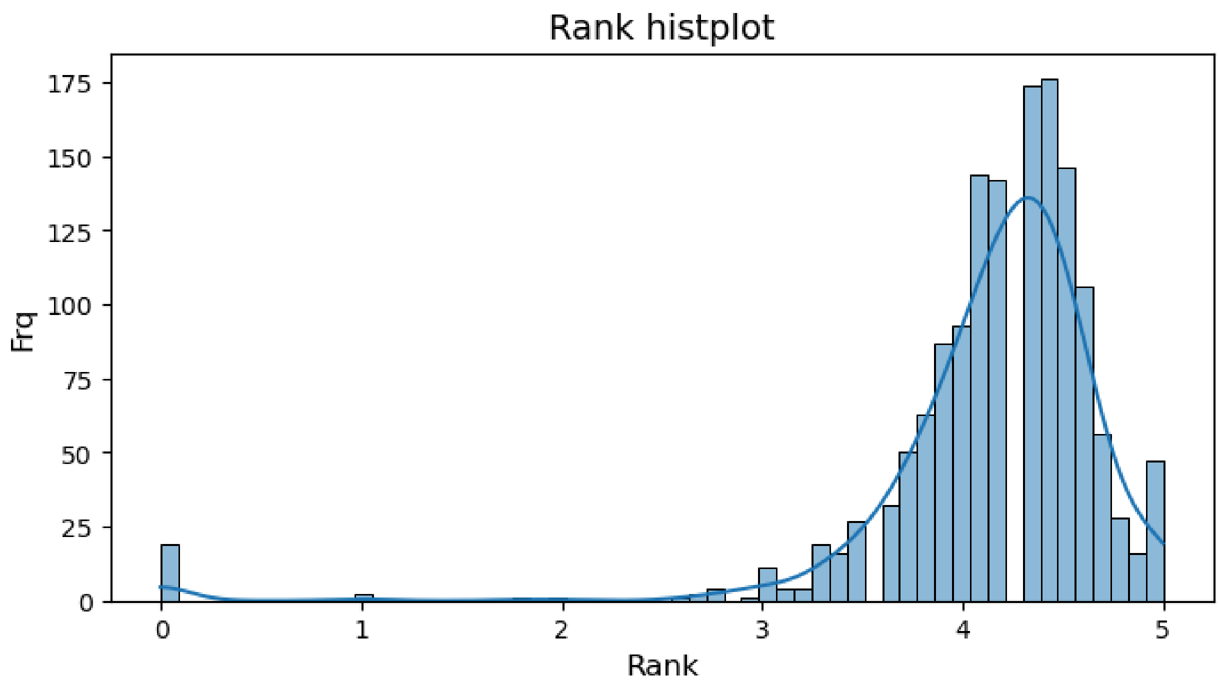
1. The dataset shows slight imbalance, which may affect model performance.

2. Oversampling or balancing techniques can improve predictions for underrepresented categories.
3. High counts for "Moisturizer" may indicate greater demand or focus, while lower counts suggest niche products.

In [16]:
```python
plt.figure(figsize= (8,4))
sns.histplot(Cosmetics['Price'],kde = True)
plt.title('Price histplot',fontsize = 14)
plt.xlabel('Price',fontsize = 12)
plt.ylabel('Frq',fontsize = 12)
plt.show()
```



In [17]:
```python
plt.figure(figsize= (8,4))
sns.histplot(Cosmetics['Rank'],kde = True)
plt.title('Rank histplot',fontsize = 14)
plt.xlabel('Rank',fontsize = 12)
plt.ylabel('Frq',fontsize = 12)
plt.show()
```



observations based on the provided histogram of "Rank":

Distribution Shape:

- The histogram shows a right-skewed distribution. Most of the frequency is concentrated around ranks between 3 and 5.
- The curve suggests a unimodal pattern, peaking at around rank 4.

Peak:

- The highest frequency occurs near rank 4, with over 150 counts.

Low Frequency for Low Ranks:

- Very few observations have ranks close to 0 or 1, indicating that low ranks are rare.

Frequency Decreases Beyond the Peak:

- The frequency gradually decreases beyond the peak rank of 4, with fewer occurrences at rank 5.

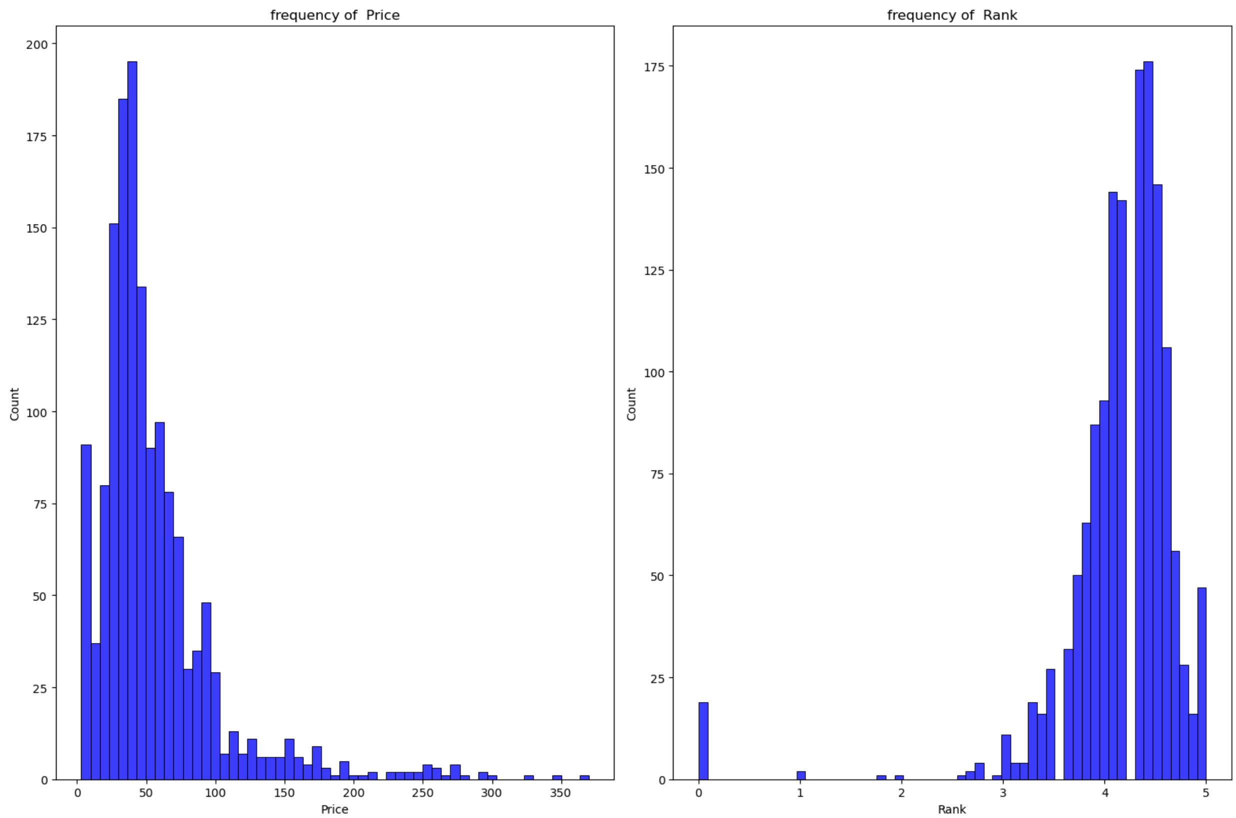
Anomalies:

- There is a slight bump at rank 0, which might represent an outlier or a specific subcategory.

In [18]:
```
num_cols = ['Price', 'Rank']

plt.figure(figsize=(15,10))
for i, col in enumerate(num_cols):
    plt.subplot(1, 2, i+1)
    sns.histplot(x=Cosmetics[col], color='blue')
    plt.title(f'frequency of  {col}', fontsize=12)
    plt.xlabel(col, fontsize=10)
```

```
plt.tight_layout()
plt.show()
```

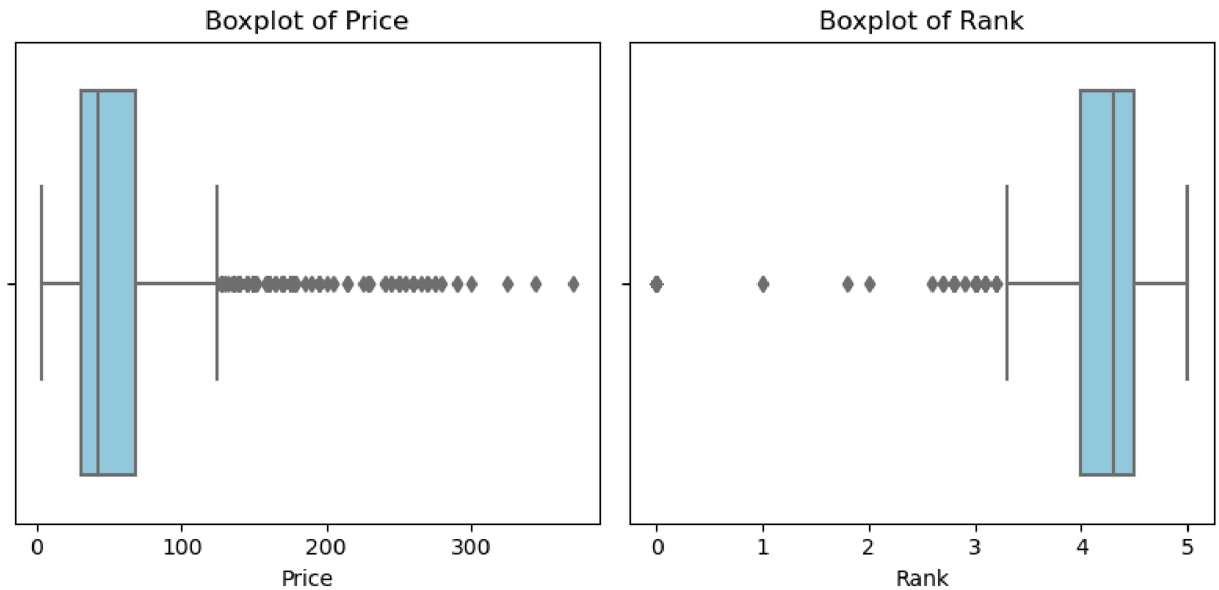


Outlier Detection

In [19]:

```
# Create boxplots for selected numerical columns
numerical_cols = ['Price', 'Rank']

plt.figure(figsize=(8, 4))
for i, col in enumerate(numerical_cols):
    plt.subplot(1, 2, i+1)
    sns.boxplot(x=Cosmetics[col], color='skyblue')
    plt.title(f'Boxplot of {col}', fontsize=12)
    plt.xlabel(col, fontsize=10)
plt.tight_layout()
plt.show()
```



Observations and Insights:

Boxplot of Price:

- Skewness and Outliers: The price data is highly right-skewed, with many outliers extending beyond the upper whisker.
- Median and IQR: The median price lies well below 100, and the interquartile range (IQR) is small, suggesting that most prices are clustered in the lower range.
- Outliers: Prices above 150 are considered outliers and could indicate premium or niche products.

> Insight: Most products are priced affordably, but a few high-priced products exist that may require further analysis, such as identifying if these are luxury or specialized items.

Boxplot of Rank:

- Skewness: The rank distribution is left-skewed, with most data concentrated towards higher ranks (near 4 and 5).
- Median and IQR: The median rank is close to 4, indicating that most products have high ranks.
- Outliers: A small number of products have low ranks (near 0 or 1), which are outliers.

> Insight: The majority of products are well-ranked, suggesting good overall quality or customer satisfaction. The lower-ranked outliers may need attention to understand why they are performing poorly.

Outlier analysis

In [20]:
```
import pandas as pd

# assuming num is athr list of numerica column names in the dataframe
num = ['Price', 'Rank']


# loop through all numerical columns and remove outliers using IQR

for col in num :
    # calculate Q1(25 th percentile) and Q3 (75th percentile)
    Q1 = Cosmetics[col].quantile(0.25)
    Q3 = Cosmetics[col].quantile(0.75)

    # Calculate IQR (interquartile range)
    IQR = Q3 - Q1

    #Define outer bounds
    lower_bound = Q1 -1.5 * IQR
    upper_bound = Q3 +1.5 * IQR

    # Remove rows where the column value is an outlier
    Cosmetics = Cosmetics[(Cosmetics[col] >= lower_bound) & (Cosmetics[col] <= upper_b

#verify the data aafter removing thr outliers
Cosmetics.head()
```

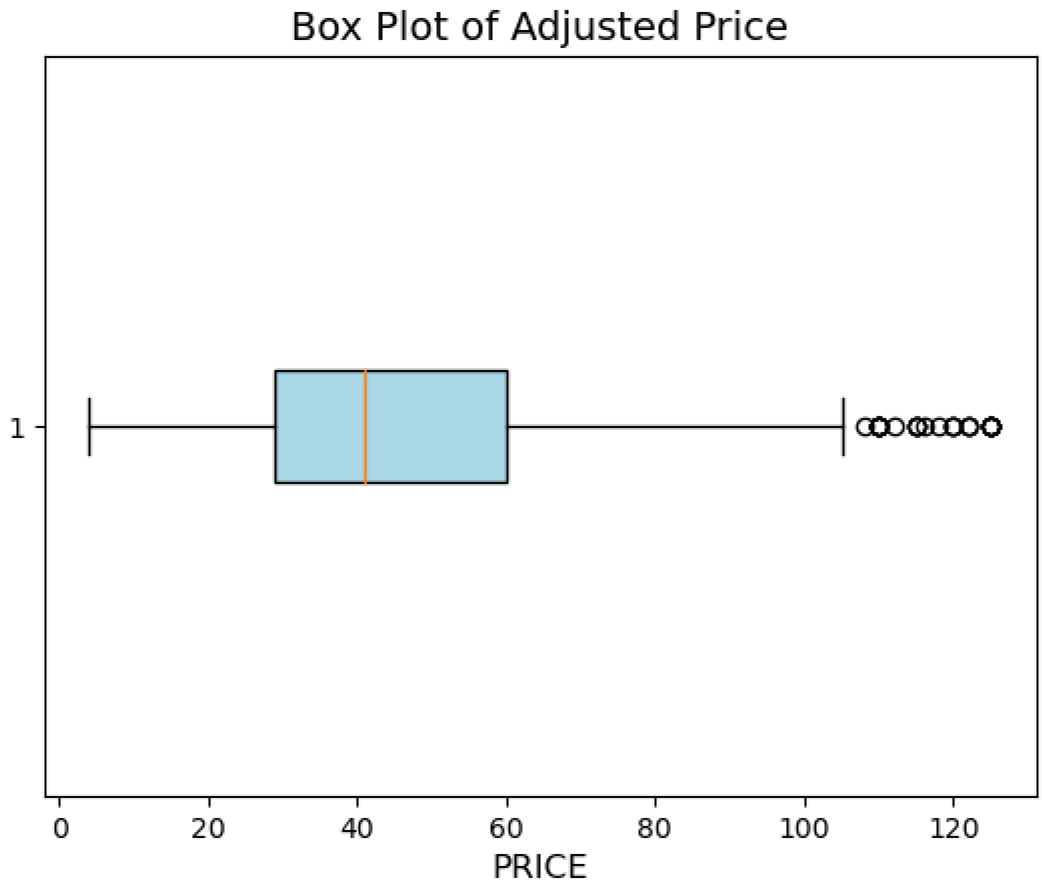
Out[20]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Norm
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1	
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1	
5	Moisturizer	TATCHA	The Water Cream	68	4.2	Water, Saccharomyces/Camellia Sinensis Leaf/Cl...	1	0	
6	Moisturizer	DRUNK ELEPHANT	Lala Retro™ Whipped Cream	60	4.2	Water, Glycerin, Caprylic/ Capric Triglyceride...	1	1	
7	Moisturizer	DRUNK ELEPHANT	Virgin Marula Luxury Facial Oil	72	4.4	100% Unrefined Sclerocraya Birrea (Marula) Ker...	1	1	

In [21]:
```
# Create a horizontal box plot for the 'p_wage_1' column
plt.boxplot(Cosmetics['Price'], vert=False, patch_artist=True,
            boxprops=dict(facecolor='lightblue'))

# Add title and labels for better clarity
plt.title('Box Plot of Adjusted Price', fontsize=14)
plt.xlabel('PRICE', fontsize=12)
```

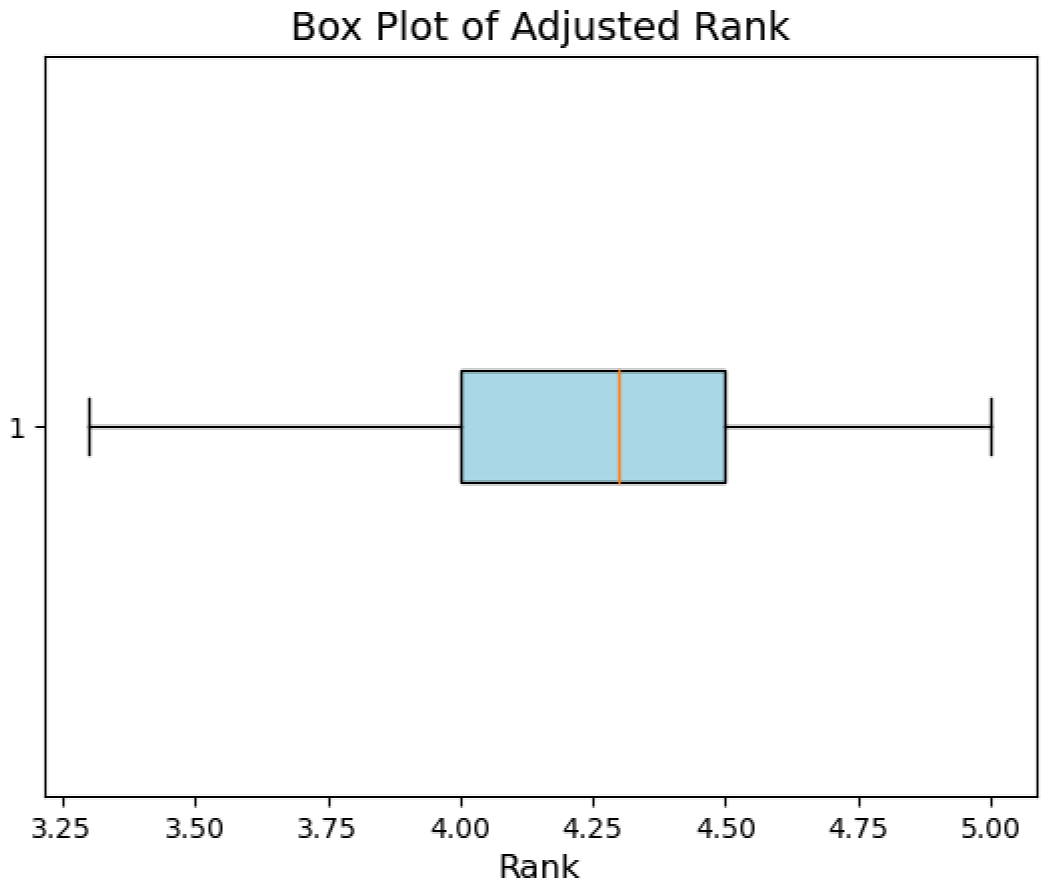
```
# Display the box plot
plt.show()
```



In [22]:
```
# Create a horizontal box plot for the 'p_wage_1' column
plt.boxplot(Cosmetics['Rank'], vert=False, patch_artist=True,
            boxprops=dict(facecolor='lightblue'))

# Add title and labels for better clarity
plt.title('Box Plot of Adjusted Rank', fontsize=14)
plt.xlabel('Rank', fontsize=12)

# Display the box plot
plt.show()
```



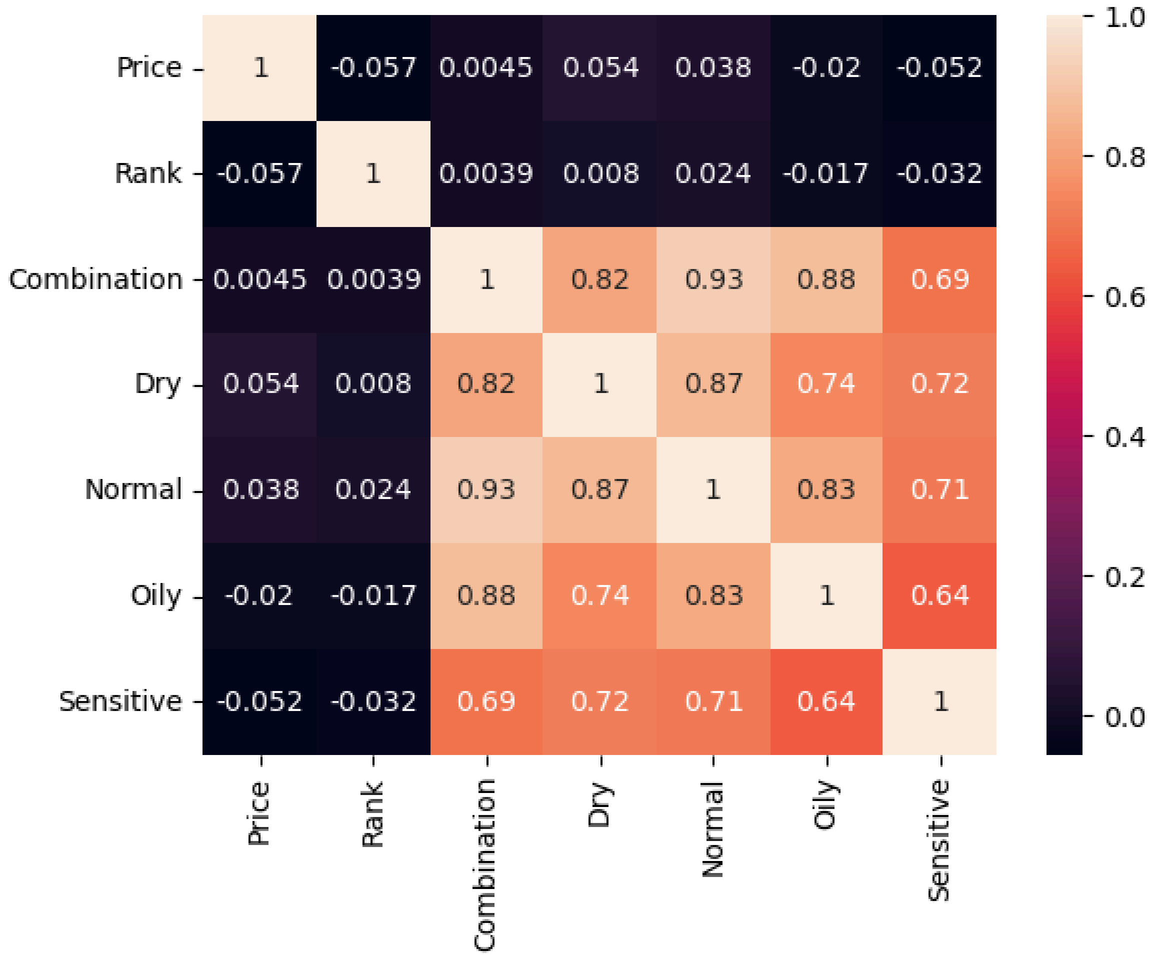
Correlation Analysis

In [23]:
```
corr=Cosmetics.corr()
corr
```

Out[23]:

	Price	Rank	Combination	Dry	Normal	Oily	Sensitive
Price	1.000000	-0.057187	0.004481	0.054110	0.038076	-0.019887	-0.051645
Rank	-0.057187	1.000000	0.003948	0.007981	0.023757	-0.016854	-0.032418
Combination	0.004481	0.003948	1.000000	0.819797	0.927527	0.881865	0.687978
Dry	0.054110	0.007981	0.819797	1.000000	0.867450	0.736823	0.719502
Normal	0.038076	0.023757	0.927527	0.867450	1.000000	0.829735	0.710575
Oily	-0.019887	-0.016854	0.881865	0.736823	0.829735	1.000000	0.642223
Sensitive	-0.051645	-0.032418	0.687978	0.719502	0.710575	0.642223	1.000000

In [24]:
```
sns.heatmap(corr,annot=True)
plt.show()
```



Observations and Insights:

Price and Rank:

Price and Rank do not influence each other much. The correlation is very weak (-0.057).

Skin Types Relationship:

- Combination skin type has a strong connection with Normal (0.93), Oily (0.88), and Dry (0.82) skin.
- This means products for combination skin are also suitable for these types.
- Dry skin is closely related to Normal (0.87) and somewhat to Oily (0.74).
- Sensitive skin has a moderate connection with Dry (0.72), Normal (0.71), and Oily (0.64).

Sensitive Skin: Products for Sensitive skin are a bit different because they don't overlap as strongly with other skin types.

Price and Skin Types:

- Price doesn't depend much on skin types, as the correlations are very small.

Key Takeaways:

- Products for Combination, Normal, and Dry skin types often overlap, so one product can work for multiple skin types.
- Sensitive skin products may need a unique focus since they are less connected to other skin types.
- Price and Rank are not strong factors when choosing products for a specific skin type.

Converting ctegorical to numerical

In [25]:
```
Cat
```

Out[25]:
```
Index(['Label', 'Brand', 'Name', 'Ingredients'], dtype='object')
```

In [26]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in Cat:
    Cosmetics[column] = le.fit_transform(Cosmetics[column])
(Cosmetics)
```

Out[26]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive
2	3	28	902	68	4.4	693	1	1	1	1	0
4	3	48	1327	38	4.1	1146	1	1	1	1	1
5	3	104	1165	68	4.2	1132	1	0	1	1	1
6	3	28	670	60	4.2	834	1	1	1	1	0
7	3	28	1253	72	4.4	75	1	1	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1465	4	108	781	34	4.1	54	0	0	0	0	0
1466	4	54	321	48	3.9	572	0	0	0	0	0
1467	4	61	1322	35	3.9	475	1	1	1	1	1
1468	4	54	322	48	3.6	991	0	0	0	0	0
1469	4	109	1031	54	3.5	696	0	0	0	0	0

1339 rows × 11 columns

In [27]:
```
Cosmetics
```

Out[27]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive
2	3	28	902	68	4.4	693	1	1	1	1	0
4	3	48	1327	38	4.1	1146	1	1	1	1	1
5	3	104	1165	68	4.2	1132	1	0	1	1	1
6	3	28	670	60	4.2	834	1	1	1	1	0
7	3	28	1253	72	4.4	75	1	1	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1465	4	108	781	34	4.1	54	0	0	0	0	0
1466	4	54	321	48	3.9	572	0	0	0	0	0
1467	4	61	1322	35	3.9	475	1	1	1	1	1
1468	4	54	322	48	3.6	991	0	0	0	0	0
1469	4	109	1031	54	3.5	696	0	0	0	0	0

1339 rows × 11 columns

Scaling the data

In [28]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
cos_scaled = pd.DataFrame(scaler.fit_transform(Cosmetics), columns=Cosmetics.columns)
```

In [29]:

```
cos_scaled
```

Out[29]:

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Se
0	0.6	0.247788	0.674141	0.528926	0.647059	0.566176	1.0	1.0	1.0	1.0	
1	0.6	0.424779	0.991779	0.280992	0.470588	0.936275	1.0	1.0	1.0	1.0	
2	0.6	0.920354	0.870703	0.528926	0.529412	0.924837	1.0	0.0	1.0	1.0	
3	0.6	0.247788	0.500747	0.462810	0.529412	0.681373	1.0	1.0	1.0	1.0	
4	0.6	0.247788	0.936472	0.561983	0.647059	0.061275	1.0	1.0	1.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	
1334	0.8	0.955752	0.583707	0.247934	0.470588	0.044118	0.0	0.0	0.0	0.0	
1335	0.8	0.477876	0.239910	0.363636	0.352941	0.467320	0.0	0.0	0.0	0.0	
1336	0.8	0.539823	0.988042	0.256198	0.352941	0.388072	1.0	1.0	1.0	1.0	
1337	0.8	0.477876	0.240658	0.363636	0.176471	0.809641	0.0	0.0	0.0	0.0	
1338	0.8	0.964602	0.770553	0.413223	0.117647	0.568627	0.0	0.0	0.0	0.0	

1339 rows × 11 columns