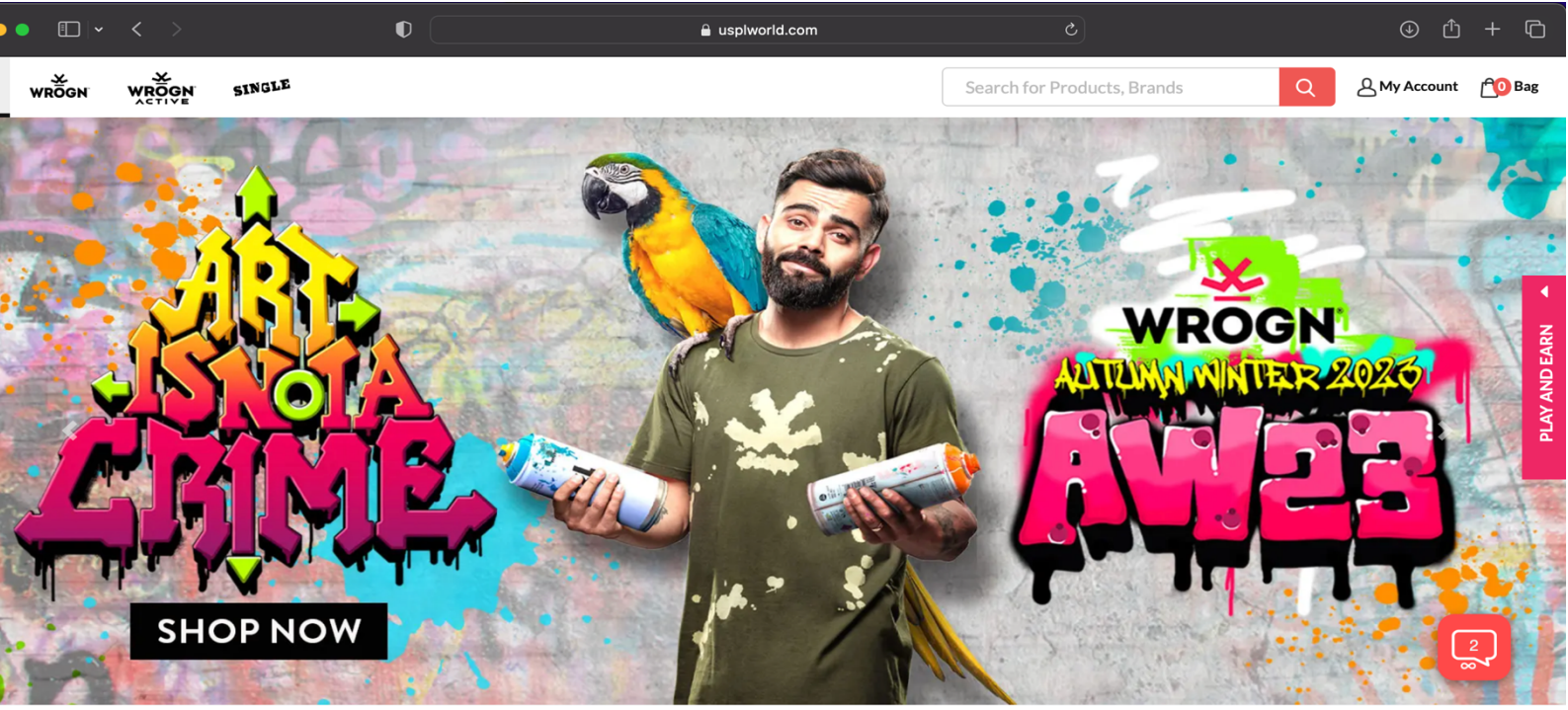


DEV-I (PYTHON)

FINAL PROJECT REPORT

Megha Garg

045030

PGDM (BDA-H)

Wrogn Website Data Analysis

URL OF GITHUB-

https://github.com/MeghaGarg045030/PythonProject_Term1

FORE School of Management
New Delhi

OBJECTIVE

The objective of this project is to systematically gather, analyze, and extract valuable insights from a dataset encompassing details regarding apparel products featured on the 'WROGN' website.

This will involve employing mathematical and statistical analyses, including measures such as mean, standard deviation, and correlation, as well as presenting the findings through compelling data visualizations.

GENERAL DESCRIPTION OF DATA

After data cleaning and preprocessing, our final dataset consists of 12 columns namely,

1. **sku**- Unique product ID
2. **name**- Name of the product
3. **in_stock**- Availability Status of the apparel.
4. **price**- Original Price of the product (before discount)
5. **discount**- Discount applied on the product
6. **display_price** – Price after discount
7. **color**- Color code of product used by the website
8. **categories**- Category to which that apparel belongs.
9. **brand**- brand name i.e, Wrogn
10. **available_size_variants**- Sizes available ranging from 18-23.
11. **priority**- represents the internal priority on which the data is shown.

12. **color_name** - added another column at the time of data preprocessing in which we extracted the name of the color from the name of the apparel.

```
df['color_name'] = df['name'].str.split().str[2]
df
```

```
df[['name','color_name']]
```

	name	color_name
0	Wrogn Men White Slim Fit Floral Cotton Blend C...	White
1	Wrogn Men Sienna Solid Slim Fit Round Neck T-S...	Sienna
2	Wrogn Men Black Solid Jacket	Black
3	Wrogn Men Red Solid Jacket	Red
4	Wrogn Men Black Solid Jacket	Black
...	...	...
1595	Wrogn Men Navy Blue Solid Trouser	Navy
1596	Wrogn Men Black Solid Jogger	Black
1597	Wrogn Men Yellow Slim Fit Polka Dots Cotton Bl...	Yellow
1598	Wrogn Men Olive Solid Jacket	Olive
1599	Wrogn Men Green Solid Jacket	Green

1600 rows × 2 columns

DATA CLEANING AND PREPROCESSING

Data cleaning is essential in data analysis to ensure accuracy and reliability. It removes errors, inconsistencies, and outliers, enabling effective, trustworthy insights and saving time and resources in the long run. Perform data preprocessing tasks to clean and structure the scraped data.

- After extraction of the color name, to see which color code belongs to which color, we used **group by** clause and **lambda function** to find the same.

```
grouped = df.groupby('color')['color_name'].apply(list).reset_index() # grouping the data
grouped['color_name'] = grouped['color_name'].apply(lambda x: list(set(x)))
print(grouped)
```

	color	color_name
0	94	[Black]
1	98	[Blue]
2	99	[Red]
3	102	[Sienna]
4	108	[Green]
5	109	[White]
6	110	[Yellow]
7	119	[Olive]
8	155	[Charcoal]
9	292	[Navy]

Next step for data pre-processing, consists of outlier removal, so to check the outliers in our data we used 2 methods :-

- Z-Score and IQR Score.

```
z_score_price = np.abs((df['price'] - df['price'].mean()) / df['price'].std())
z_score_discount = np.abs((df['discount'] - df['discount'].mean()) / df['discount'].std())


# Identify outliers using a threshold (e.g., Z-Score > 2)
threshold = 2
outliers = df[(z_score_price > threshold) | (z_score_discount > threshold)]


# IQR method for price and discount columns
numeric_columns = ['price', 'discount'] # List of numeric columns
Q1 = df[numeric_columns].quantile(0.25)
Q3 = df[numeric_columns].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Identify outliers using IQR
iqr_outliers = df[(df[numeric_columns] < lower_bound) | (df[numeric_columns] >
upper_bound)].dropna()


print("Z-Score Outliers:")
print(outliers)
print("\nIQR Outliers:")
print(iqr_outliers)
```

```
Z-Score Outliers:
Empty DataFrame
Columns: [sku, name, in_stock, price, discount, display_price, color, categories, brand, available_size_variants, priority, color_name]
Index: []

IQR Outliers:
Empty DataFrame
Columns: [sku, name, in_stock, price, discount, display_price, color, categories, brand, available_size_variants, priority, color_name]
Index: []
```

The final dataset to be used for further analysis:

```
df
```

	sku	name	in_stock	price	discount	display_price	color	categories	brand	available_size_variants	priority	color_name
0	WOSH0201S	Wrogn Men White Slim Fit Floral Cotton Blend C...	True	2099.0	55	944.0	109	Shirts	wrogn	_20__23__18__21__19_	59.0	White
1	WETS0704	Wrogn Men Sienna Solid Slim Fit Round Neck T-S...	True	999.0	50	499.0	102	T Shirts	wrogn	_23__18__20__19__21_	72.0	Sienna
2	WNJK6005	Wrogn Men Black Solid Jacket	True	3999.0	55	1799.0	94	Autumn Wear	wrogn	_21__18__20__19__23__22_	84.0	Black
3	WNJK6019	Wrogn Men Red Solid Jacket	True	4599.0	55	2069.0	99	Autumn Wear	wrogn	_19__23__22__20__18__21_	86.0	Red
4	WNJK6354	Wrogn Men Black Solid Jacket	True	4599.0	55	2069.0	94	Jackets	wrogn	_23_	88.0	Black
...	...	...	...	...	...	...	...	...	...	...	...	...
1595	WNTR4005	Wrogn Men Navy Blue Solid Trouser	True	2999.0	60	1199.0	292	Trousers	wrogn	_14__15__13__12__16__17_	184.0	Navy
1596	AMJG2059	Wrogn Men Black Solid Jogger	True	2399.0	52	1151.0	94	Joggers	wrogn	_20__19__18__21__23_	187.0	Black
1597	WOSH0198S	Wrogn Men Yellow Slim Fit Polka Dots Cotton Bl...	True	2099.0	55	949.0	110	Shirts	wrogn	_19__18__23__20__21_	239.0	Yellow
1598	WNJK6340	Wrogn Men Olive Solid Jacket	True	4499.0	60	1799.0	119	Jackets	wrogn	_19__21__23__18_	256.0	Olive

DATA ANALYSIS

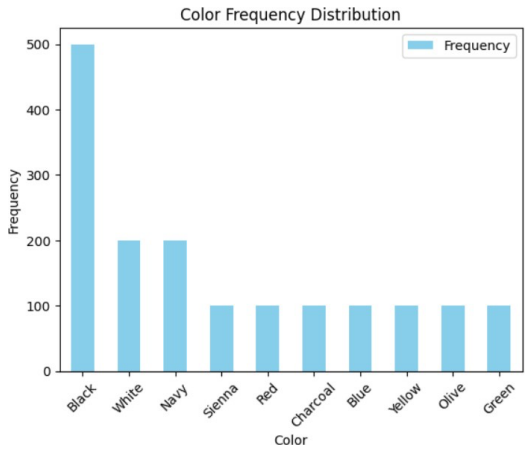
1. COLOR FREQUENCY DISTRIBUTION -

The color frequency distribution graph for clothing apparel offers valuable insights into consumer preferences, seasonal trends, and competitive analysis. It informs inventory management, design choices, and data-driven decision-making, enhancing overall business strategy in the apparel industry.

To find which color apparels are most available and which color apparels are least available on the website.

```
# Making a color frequency distribution.
color_frequency = df['color_name'].value_counts().reset_index()
color_frequency.columns = ['Color', 'Frequency']
# Display color frequency table
table = tabulate(color_frequency, headers='keys', tablefmt='pretty', showindex=False)
print(table)
# Create a bar chart for color frequency distribution
plt.figure(figsize=(8, 6))
color_frequency.plot(x='Color', y='Frequency', kind='bar', color='skyblue')
plt.title('Color Frequency Distribution')
plt.xlabel('Color')
plt.ylabel('Frequency')
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.show()
```

```
+----------+-----------+
|  Color   | Frequency |
+----------+-----------+
|  Black   |    500    |
|  White   |    200    |
|   Navy   |    200    |
|  Sienna  |    100    |
|   Red    |    100    |
| Charcoal |    100    |
|   Blue   |    100    |
|  Yellow  |    100    |
|  Olive   |    100    |
|  Green   |    100    |
+----------+-----------+
<Figure size 800x600 with 0 Axes>
```



2. CATEGORY FREQUENCY DISTRIBUTION -

Creating a categorical frequency distribution graph for apparel helps provide a clear overview of how different clothing categories are distributed within the dataset. This visual representation aids in identifying popular product categories, optimizing inventory management, and tailoring marketing strategies to customer preferences.

To know which category products are most available on the website and which are the least available category.

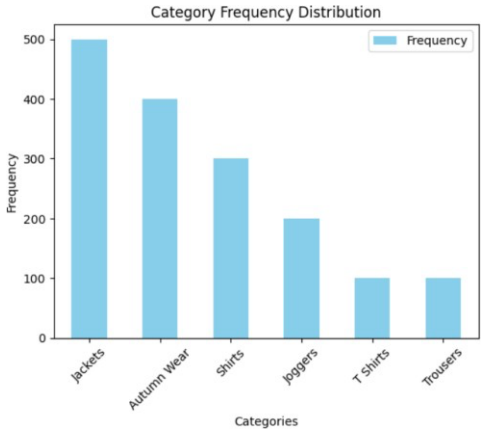
```
category_frequency = df['categories'].value_counts().reset_index()

category_frequency.columns = ['Category', 'Frequency']

# Display category frequency table

table = tabulate(category_frequency, headers='keys', tablefmt='pretty', showindex=False)

print(table)

# Create a bar chart for category frequency distribution

plt.figure(figsize=(6, 4))

category_frequency.plot(x='Category', y='Frequency', kind='bar', color='skyblue')

plt.title('Category Frequency Distribution')

plt.xlabel('Categories')
```

plt.ylabel('Frequency')

plt.xticks(rotation=45) # Rotate x-axis labels for better readability

plt.show()

Category	Frequency
Jackets	500
Autumn Wear	400
Shirts	300
Joggers	200
T Shirts	100
Trousers	100



3. MEAN, STANDARD DEVIATION, MIN & MAX -

The mean offers insight into the average price and discount, while the standard deviation measures the dispersion, helping to identify pricing consistency or outliers. These metrics inform pricing strategies, ensuring competitiveness and profitability in the apparel market.

df[["price","discount","display_price"]].describe()

 # Calculating the basic mean,count,std for the whole data.

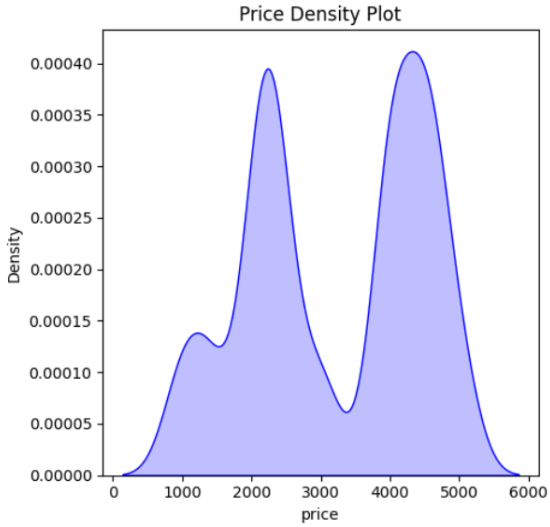
	price	discount	display_price
count	1600.000000	1600.000000	1600.000000
mean	3224.000000	56.500000	1378.312500
std	1245.128092	3.373739	500.105395
min	999.000000	50.000000	499.000000
25%	2174.000000	55.000000	947.750000
50%	3499.000000	55.000000	1399.000000
75%	4349.000000	60.000000	1799.000000
max	4999.000000	60.000000	2069.000000

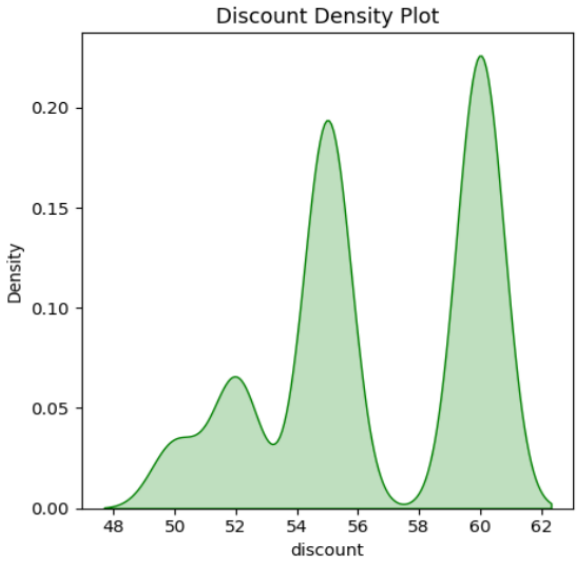
4. DENSITY PLOT FOR PRICE, DISCOUNT AND DISPLAY PRICE -

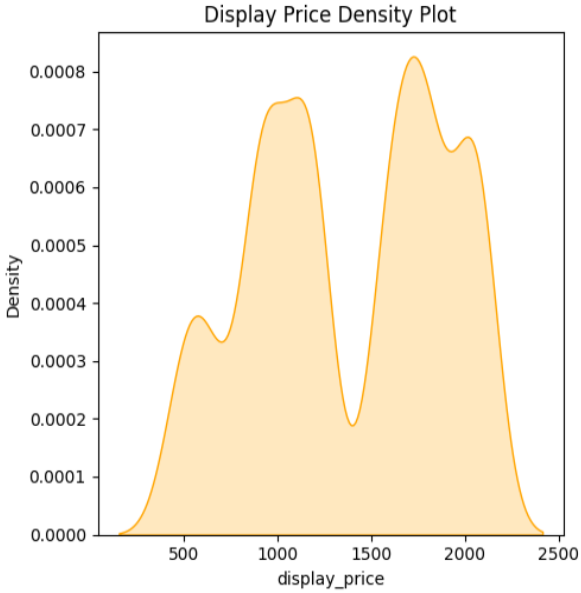
Creating density plots for price, discount, and discounted price of apparel offers a comprehensive visual representation of their distribution and interrelation.

These plots enable a deeper understanding of pricing dynamics, discount effects, and customer behavior, supporting data-driven decisions on pricing strategies and promotional activities within the apparel business.

```
import seaborn as sns
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))
# Create kernel density plots for 'price', 'discount', and'display_price'
sns.kdeplot(data=df['price'], ax=axes[0], fill=True, color='blue')
axes[0].set_title('Price Density Plot')
sns.kdeplot(data=df['discount'], ax=axes[1], fill=True, color='green')
axes[1].set_title('Discount Density Plot')
sns.kdeplot(data=df['display_price'], ax=axes[2], fill=True, color='orange')
axes[2].set_title('Display Price Density Plot')
# Adjust subplot layout
plt.tight_layout()
# Show the kernel density plots
plt.show()
```


Price Density Plot
Density
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
1000
2000
3000
4000
5000
6000
price


Discount Density Plot
Density
0.20
0.15
0.10
0.05
0.00
48
50
52
54
56
58
60
62
discount


Display Price Density Plot
Density
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
500
1000
1500
2000
2500
display_price

5. CORRELATION BETWEEN PRICE AND DISCOUNT -

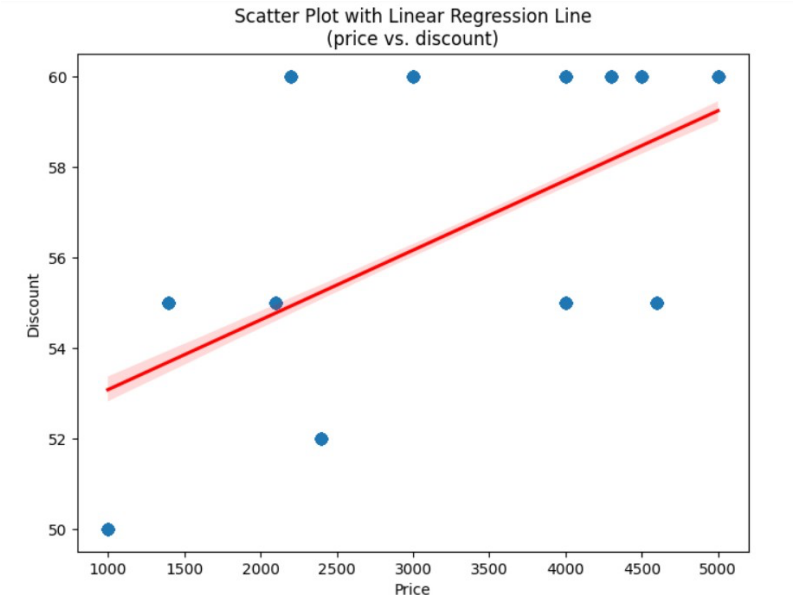
Calculating the correlation between apparel prices and discounts is crucial to determine if there's a relationship between these variables.

A strong negative correlation, for instance, might suggest that as prices increase, discounts decrease, influencing pricing strategies and understanding how price changes affect discounts, aiding in effective pricing decisions in the apparel business.

```
# Find Correlation Between Price And Discount

correlation_coefficient = df['price'].corr(df['discount'])

# Print the correlation coefficient

print(f"Pearson's Correlation Coefficient between Price and Discount:
{correlation_coefficient:.2f}")
```

```
Pearson's Correlation Coefficient between Price and Discount: 0.57
```

```
plt.figure(figsize=(8, 6))

sns.regplot(x='price', y='discount', data=df, scatter_kws={'s': 50}, line_kws={'color': 'red'})

# Add labels and a title

plt.title('Scatter Plot with Linear Regression Line\n(price vs. discount)')

plt.xlabel('Price')

plt.ylabel('Discount')

# Show the plot

plt.show()
```



6. CORRELATION BETWEEN PRICE AND PRIORITY -

Calculating the correlation between apparel prices and their assigned priority provides insights into pricing strategies aligned with product importance.

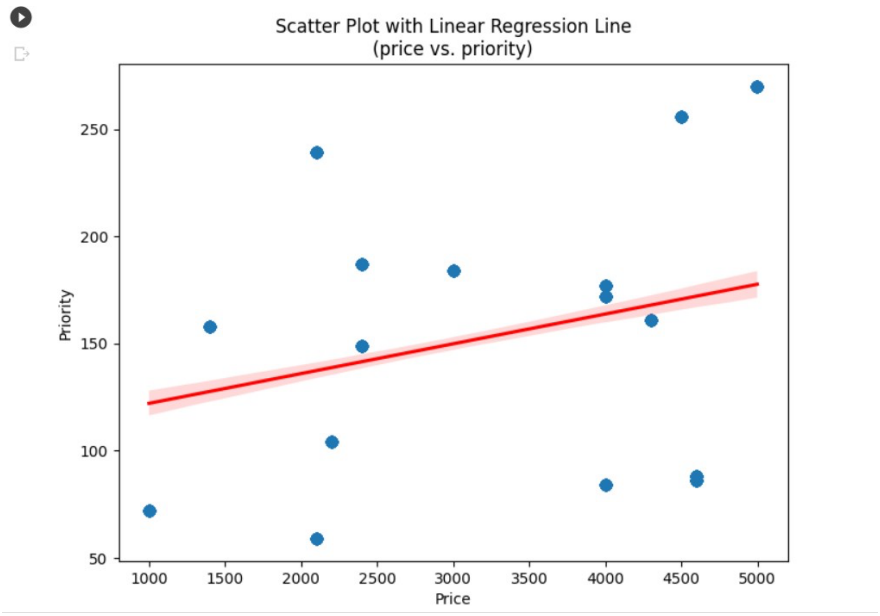
This analysis helps identify whether higher-priced items are also considered higher priority or if there are discrepancies, guiding decisions on pricing, inventory management, and marketing efforts to optimize profitability and customer satisfaction in the apparel business.

```
# Correlation between Price and Priority.

correlation_coefficient = df['price'].corr(df['priority'])

# Print the correlation coefficient

print(f"Pearson's Correlation Coefficient between Price and Discount:
{correlation_coefficient:.2f}")
```

```
Pearson's Correlation Coefficient between Price and Discount: 0.27
```

```
plt.figure(figsize=(8, 6))
```

```
sns.regplot(x='price', y='priority', data=df, scatter_kws={'s': 50}, line_kws={'color': 'red'})
# Add labels and a title
plt.title('Scatter Plot with Linear Regression Line\n(price vs. priority)')
plt.xlabel('Price')
plt.ylabel('Priority')
# Show the plot
plt.show()
```



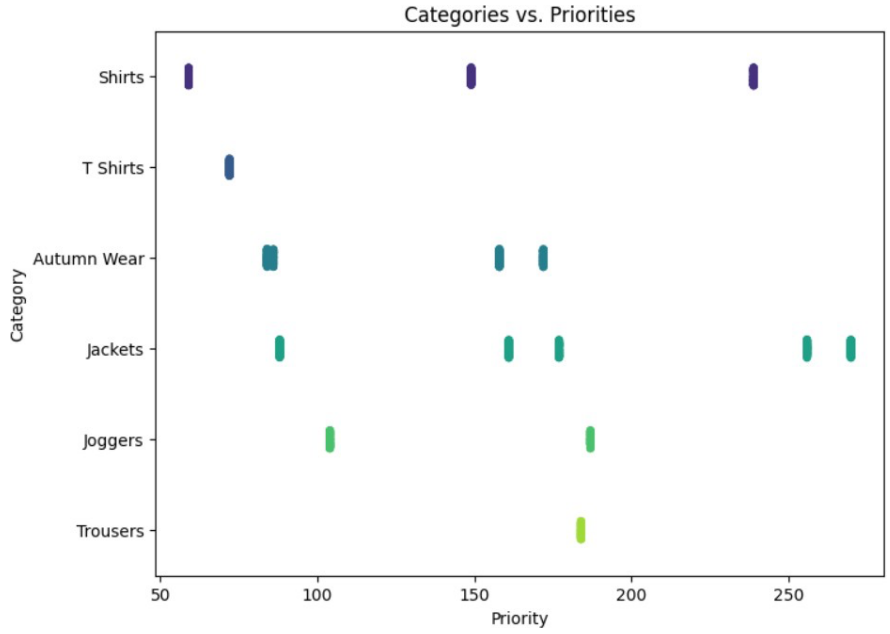
7. RELATIONSHIP BETWEEN CATEGORY AND PRIORITY -

Examining the relationship between apparel category and priority is essential for effective inventory management and customer satisfaction.

This analysis can reveal if certain categories are consistently associated with higher priorities, helping guide decisions on stock allocation, marketing focus, and product promotion strategies in the apparel business.

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt

pivot_table = df.pivot_table(index='categories', columns='priority', aggfunc='size', fill_value=0)

plt.figure(figsize=(8, 6))

sns.stripplot(x='priority', y='categories', data=df, jitter=True, palette='viridis')

plt.title('Categories vs. Priorities')

plt.xlabel('Priority')

plt.ylabel('Category')

plt.show()
```

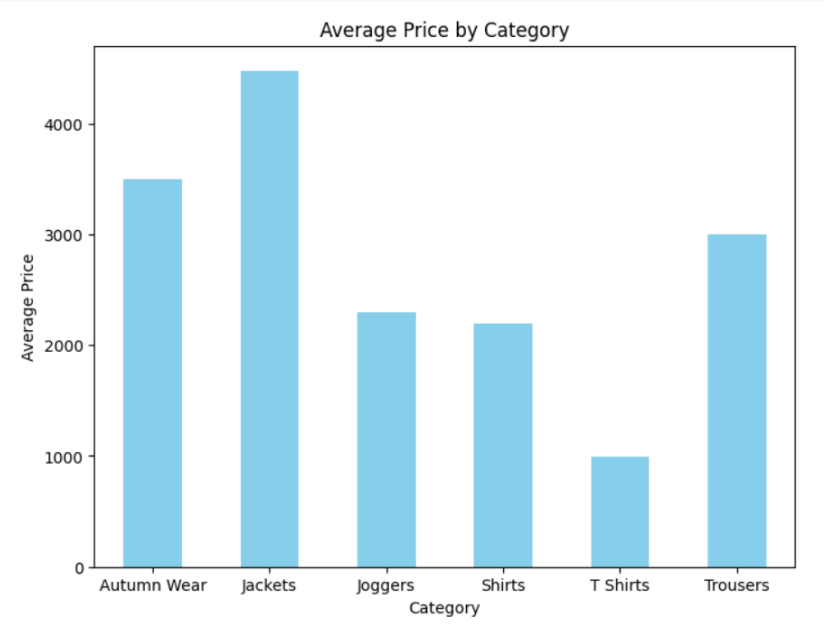


8. AVERAGE PRICE OF EACH CATEGORY -

Calculating the average price for each category of apparel is vital for understanding pricing dynamics within the product range.

This analysis offers valuable insights into category-specific pricing strategies, helps identify premium and budget segments, and supports informed decisions regarding pricing adjustments and marketing targeting in the apparel business.

```
#Calculating Average Price Of Each Category.
average_price_by_category = df.groupby('categories')['price'].mean(
plt.figure(figsize=(8, 6))
average_price_by_category.plot(kind='bar', color='skyblue')
plt.title('Average Price by Category')
plt.xlabel('Category')
plt.ylabel('Average Price')
plt.xticks(rotation=0)
plt.show()
```



9. DIVIDING DATA ON THE BASIS OF CATEGORIES -

Segmenting the data based on apparel category is essential to gain a granular understanding of product performance and customer preferences.

This division enables tailored analyses, such as assessing category-specific trends, optimizing inventory, and crafting targeted marketing

strategies, which are crucial for effective decision-making and competitiveness in the apparel market.

```
grouped = df.groupby('categories')

category_dataframes = {} # Empty Dictionary

# Iterate through the groups and create DataFrames for each category

for category, group in grouped:

    category_dataframes[category] = group

# To Access A Particular Data Frame:- category_dataframes.get('Shirt') or category_dataframes["Shirts"]

category_dataframes["Jackets"].head(5) # Accessing a particular category data.
```

	sku	name	in_stock	price	discount	display_price	color	categories	brand	available_size_variants	priority	color_name
4	WNJK6354	Wrogn Men Black Solid Jacket	True	4599.0	55	2069.0	94	Jackets	wrogn	_23_	88.0	Black
8	WNJK6411	Wrogn Men Black Camouflage Jacket	True	4299.0	60	1719.0	94	Jackets	wrogn	_23__18__21__19_	161.0	Black
10	WNJK6336	Wrogn Men Navy Blue Solid Jacket	True	3999.0	60	1599.0	292	Jackets	wrogn	_23_	177.0	Navy
14	WNJK6340	Wrogn Men Olive Solid Jacket	True	4499.0	60	1799.0	119	Jackets	wrogn	_19__21__23__18_	256.0	Olive
15	WNJK6347	Wrogn Men Green Solid Jacket	True	4999.0	60	1999.0	108	Jackets	wrogn	_19_	270.0	Green

10. CALCULATING MEAN, MEDIAN AND STANDARD DEVIATION FOR “PRICE” AND “DISCOUNT” OF SHIRTS CATEGORY -

To know insights of how the price and discount of shirts ranges.

```
# Calculating the mean, median, and standard deviation for "price" and "discount" in the "Shirts" DataFrame

shirts_df = category_dataframes["Shirts"]

# Calculate the mean, median, and standard deviation for "price"

mean_price = shirts_df["price"].mean()

median_price = shirts_df["price"].median()

stddev_price = shirts_df["price"].std()

# Calculate the mean, median, and standard deviation for "discount"
```

```
mean_discount = shirts_df["discount"].mean()

median_discount = shirts_df["discount"].median()

stddev_discount = shirts_df["discount"].std()

# Print the results

print("Price - Mean:", mean_price)

print("Price - Median:", median_price)

print("Price - Standard Deviation:", stddev_price)

print("Discount - Mean:", mean_discount)

print("Discount - Median:", median_discount)

print("Discount - Standard Deviation:", stddev_discount)
```

```
Price - Mean: 2199.0
Price - Median: 2099.0
Price - Standard Deviation: 141.6576493949657
Discount - Mean: 54.0
Discount - Median: 55.0
Discount - Standard Deviation: 1.416576493949657
```

11. CALCULATING MEAN, MEDIAN AND STANDARD DEVIATION FOR “PRICE” AND “DISCOUNT” OF JACKETS CATEGORY -

To know insights of how the price and discount of Jacket ranges.

```
 # Calculating the mean, median, and standard deviation for "price" and "discount"in the "Jacket" DataFrame

jackets_df = category_dataframes["Jackets"]

# Calculate the mean, median, and standard deviation for "price"

mean_price = jackets_df["price"].mean()

median_price = jackets_df["price"].median()

stddev_price = jackets_df["price"].std()
```

```
# Calculate the mean, median, and standard deviation for "discount"
mean_discount = jackets_df["discount"].mean()
median_discount = jackets_df["discount"].median()
stddev_discount = jackets_df["discount"].std()
# Print the results
print("Price - Mean:", mean_price)
print("Price - Median:", median_price)
print("Price - Standard Deviation:", stddev_price)
print("Discount - Mean:", mean_discount)
print("Discount - Median:", median_discount)
print("Discount - Standard Deviation:", stddev_discount)
```

```
Price - Mean: 4479.0
Price - Median: 4499.0
Price - Standard Deviation: 331.39046346954086
Discount - Mean: 59.0
Discount - Median: 60.0
Discount - Standard Deviation: 2.0020030050087656
```

12. CALCULATING MEAN, MEDIAN AND STANDARD DEVIATION FOR “PRICE” AND “DISCOUNT” OF AUTUMN WEAR CATEGORY-

To know insights of how the price and discount of Autumn Wearranges.

```
# Calculating the mean, median, and standard deviation for "price" and "discount" in the "Autumn Wear" DataFrame
autumn_df = category_dataframes["Autumn Wear"]
# Calculate the mean, median, and standard deviation for "price"
mean_price = autumn_df["price"].mean()
```

```
median_price = autumn_df["price"].median()
stddev_price = autumn_df["price"].std()
# Calculate the mean, median, and standard deviation for "discount"
mean_discount = autumn_df["discount"].mean()
median_discount = autumn_df["discount"].median()
stddev_discount = autumn_df["discount"].std()
# Print the results
print("Price - Mean:", mean_price)
print("Price - Median:", median_price)
print("Price - Standard Deviation:", stddev_price)
print("Discount - Mean:", mean_discount)
print("Discount - Median:", median_discount)
print("Discount - Standard Deviation:", stddev_discount)
```

```
Price - Mean: 3499.0
Price - Median: 3999.0
Price - Standard Deviation: 1238.480757406495
Discount - Mean: 56.25
Discount - Median: 55.0
Discount - Standard Deviation: 2.1677749238103
```

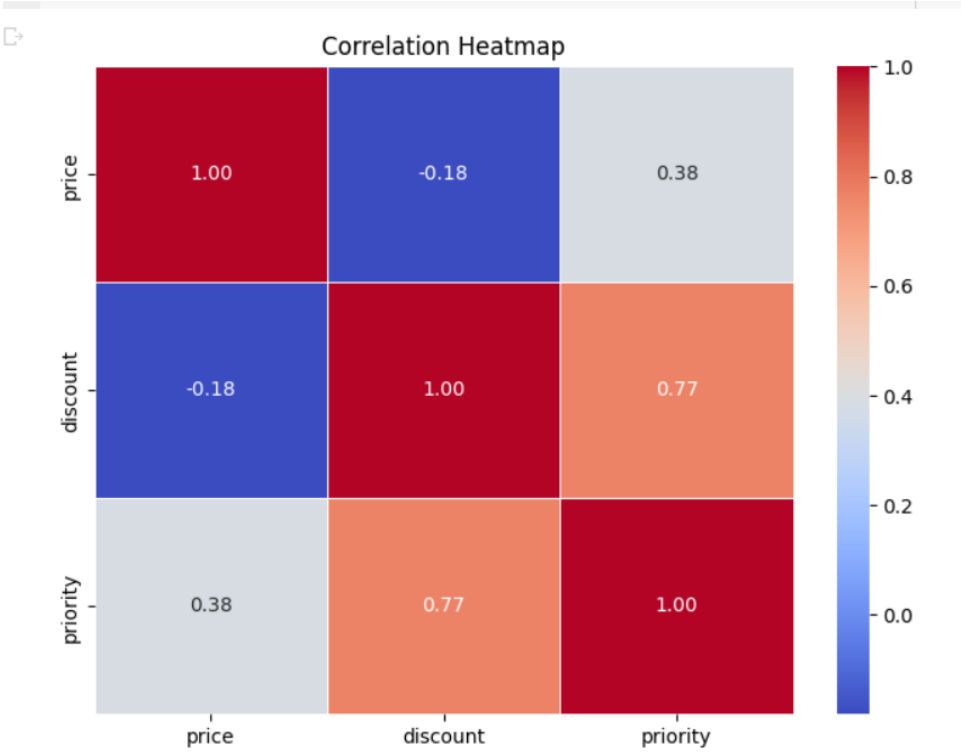
13. CREATING CORRELATION HEAT MAP OF JACKET CATEGORY-

Heatmap is a graphical way to visualize visitor behavior data in the form of hot and cold spots employing a warm-to-cool color scheme. The warm colors indicate sections with the most visitor interaction, red being the area of highest interaction, and the cool colors point to the sections with the lowest interaction.

Generating a correlation heatmap for the "jacket" category apparel is essential to uncover intricate relationships between price, priority, and discounts within this specific product group.

This visual representation aids in identifying potential patterns, optimizing pricing strategies, inventory allocation, and product promotion tactics, thus enhancing decision-making for jackets within the apparel business.

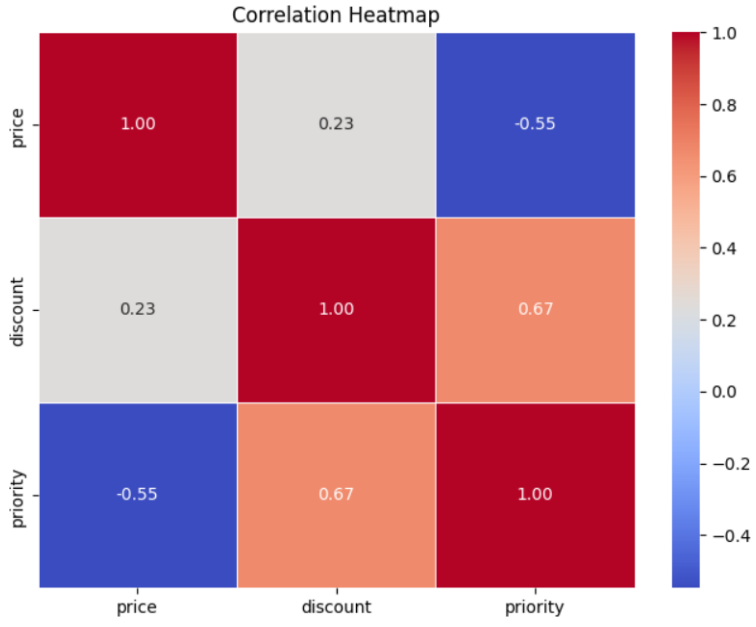
```
columns_for_heatmap = ['price', 'discount', 'priority']

subset_df = category_dataframes["Jackets"][columns_for_heatmap]

# Calculate the correlation matrix for the selected columns

correlation_matrix = subset_df.corr()

# Create a heatmap to visualize the correlations

plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()
```



14. CREATING CORRELATION HEAT MAP OF AUTUMN WEAR CATEGORY -

Heat Map to find how much price, priority and discount are related to each other under the Autumn Wear category.

```
# Creating Correlation HeatMap of the Autumn Wear Dataframe

columns_for_heatmap = ['price', 'discount', 'priority']

subset_df = category_dataframes["Autumn Wear"][columns_for_heatmap]

# Calculate the correlation matrix for the selected columns

correlation_matrix = subset_df.corr()

# Create a heatmap to visualize the correlations

plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()
```



FINDINGS AND INFERENCES

1. **Most popular color**:

   From the frequency table and bar chart, it is evident that Wrogn websites contain much more black color apparel as compared to other colors with the frequency of 500. By analyzing the bar chart, it's possible to identify any dominant color trends.

2. **Most available category:**

   From the frequency table we observe, the number of jackets available are much more as compared to the product of other categories.

3. **Mean, Deviation and Range:**

   Mean of the price: 3224

   Standard deviation: 1245

   Price Range: 999-4999

   Mean of the discount: 56.5

   Standard deviation: 3.4

   Discounts Range: 50-60

   Mean of the display price: 1378

   Standard deviation: 500

   Display price Range: 499-2070

4. **Frequency density:**

Most of the apparels cost either 2000 or 4500 according to the price density plot.

Most of the apparels after applying the discount cost either 1000 or 1750 according to the display price density plot.

So, we interpret that the cost of the apparel is halved after applying the various discounts (50% - 65%) to attract the customers.

5. **Correlations using Linear Regression:**

   Price and discount are moderately correlated because the correlation coefficient is 0.57.

   Priority and price are independent of each other because the correlation coefficient is very low i.e, 0.27.

   Jackets and Shirts usually have apparels in each priority range whereas t-shirts have the lowest priority values.

6. Average price of jackets is the highest (approx. 4500) as compared to others whereas t-shirts have the lowest average price (approx. 1000).

7. Difference between Mean and Median for price and discount of various categories is very low.

8. The price - discount are negatively correlated whereas discount - priority and price - priority are positively correlated under the Jackets category.

9. Price- Discount and Discount - Priority are positively correlated whereas Price - Priority are negatively correlated under the Autumn Wear Category.

MANAGERIAL INSIGHTS AND IMPLICATIONS

1. **Inventory Management**: The color and category frequency distribution can be valuable for inventory management. Knowing which colors are more popular can help in stock management and ordering decisions. For example, since the no. of black colored apparels are high, the brand should focus on introducing and diversifying more color offerings based on current trends and customer's preferences. Careful management in different categories is crucial to ensure product availability.

2. **Season Planning:** Planning inventory and marketing campaigns around seasonal trends to maximize sales. For example, increasing discounts for products which are suitable for that particular season.

3. **Discount Planning**: Data from the discount and priority column can be used to identify the products having lower priority as well as lower discount, such products can be sold with higher discount rates to increase sales of lower priority products.

4. Apparel should be available in all the ranges so as to specify the need of each type of customer. Some customers cannot afford to buy high-cost clothing whereas other customers might prefer high-cost quality products. Price Density plot can be used to assist in the same.

5. **Marketing Campaign:** Data insights are used to target marketing campaigns more effectively, highlighting popular products or promotions. Considering the data insights of Wrogn Brand, ''Black Jackets'' are the most prioritized and popular. Smart ad campaigns can also help in more sales of low priority fewer selling products.

6. **Supply Chain Efficiency:** Optimize the supply chain based on category and color preferences. Ensure that products with higher demand have shorter lead times and are readily available.

7. **Profitability Enhancement:** Data-driven decisions on pricing, discounts, and category management can lead to improved profitability by ensuring that resources are allocated efficiently and in line with customer demands.

8. **Pricing Strategy Refinement:** By examining the mean and standard deviation of prices and discounts, managers can refine pricing strategies for different apparel categories. They can adjust pricing to align with customer expectations and market trends.