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

#EDA -1 Bike Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('BIKE DETAILS.csv')
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1061,\n  \"fields\":
[\n    {\n      \"column\": \"name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 279,\n
\"samples\": [\n          \"Hero Xtreme Sports\",\n          \"Hero
Honda Passion\",\n          \"TVS Streak\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"selling_price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
56304,\n        \"min\": 5000,\n        \"max\": 760000,\n
\"num_unique_values\": 130,\n        \"samples\": [\n          72000,\
n          160000,\n          26000\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"year\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 4,\n        \"min\": 1988,\n
\"max\": 2020,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          2012,\n          2003,\n          2020\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"seller_type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"Dealer\",\n          \"Individual\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"owner\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"2nd owner\",\n          \"4th owner\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"km_driven\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51623,\n        \"min\": 350,\n        \"max\": 880000,\n
\"num_unique_values\": 304,\n        \"samples\": [\n          19500,\
n          11500\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ex_showroom_price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 77496.58718945317,\n        \"min\":
30490.0,\n        \"max\": 1278000.0,\n        \"num_unique_values\":
230,\n        \"samples\": [\n          54000.0,\n          64589.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
df.shape
```

```
(1061, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1061 entries, 0 to 1060
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   name               1061 non-null   object
 1   selling_price      1061 non-null   int64
 2   year               1061 non-null   int64
 3   seller_type        1061 non-null   object
 4   owner              1061 non-null   object
 5   km_driven          1061 non-null   int64
 6   ex_showroom_price  626 non-null    float64
dtypes: float64(1), int64(3), object(3)
memory usage: 58.1+ KB
```

Q1.What is the range of selling prices in the dataset?

```
range = max(df['selling_price'])-min(df['selling_price'])
range
```

```
755000
```

#Q2.What is the median selling price for bikes in the datasetD

```
median = np.median(df['selling_price'])
median
```

```
45000.0
```

#Q3. What is the most common seller type?

```
import statistics
mode = statistics.mode(df['selling_price'])
mode
```

```
25000
```

Q4.How many bikes have driven more than 50,000 kilometers?

```
No_of_bike = df[df['km_driven']>50000].shape[0]
No_of_bike
```

```
170
```

#Q5.What is the average km_driven value for each ownership type?

```
average = df.groupby('owner')['km_driven'].mean()
average
```

```
owner
1st owner     32816.583333
2nd owner     39288.991870
3rd owner     33292.181818
4th owner    311500.000000
Name: km_driven, dtype: float64
```

#Q6. What proportion of bikes are from the year 2015 or older?

```
older_2015 = df[df['year']<=2015].shape[0]
p=df[df['year']<=2015].shape[0]/df.shape[0]
print(older_2015)
print(p)
#proportion = 601/1061
```

```
601
0.5664467483506126
```

#Q7.What is the trend of missing values across the dataset?

```
missing_value = df.isnull().sum()
missing_value
```

```
name                   0
selling_price          0
year                   0
seller_type            0
owner                  0
```

```
km_driven              0
ex_showroom_price    435
dtype: int64
```

#Q8.What is the highest ex_showroom_price recorded, and for which bike?

```
maximum_price = df['ex_showroom_price'].max()
Bike = df[df['ex_showroom_price']==maximum_price]['name'].values
print("Bike_name", Bike)
print("Price", maximum_price)
```

```
Bike_name ['Harley-Davidson Street Bob']
Price 1278000.0
```

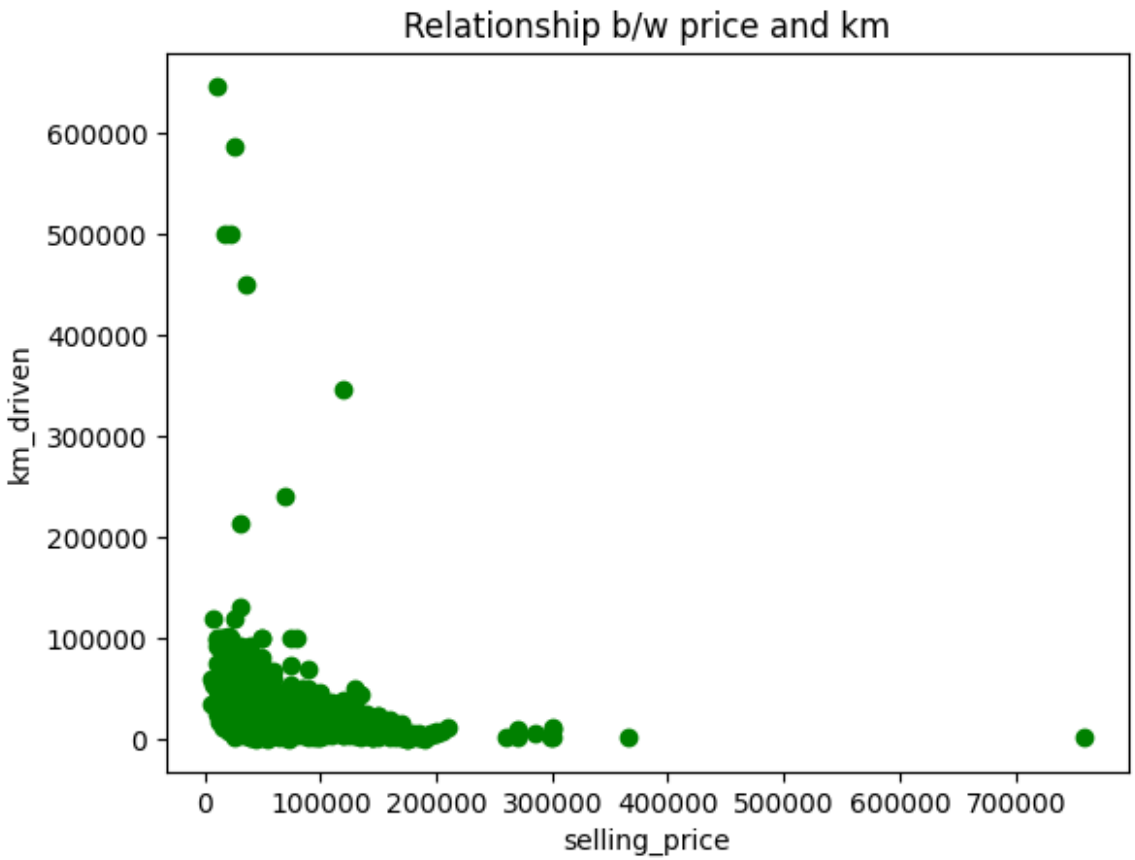
#Q9.What is the total number of bikes listed by each seller type?

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

```
seller_type
Individual    1055
Dealer           6
Name: count, dtype: int64
```

#Q10. What is the relationship between selling_price and km_driven for first-owner bikes?

```
x=df[df['owner']=='1st owner']
plt.scatter(x['selling_price'],x['km_driven'],color='green')
plt.xlabel('selling_price')
plt.ylabel('km_driven')
plt.title('Relationship b/w price and km')
plt.show()
```



#Q11. Identify and remove outliers in the km_driven column using the IQR method?

```
Q1 =df['km_driven' ].quantile(0.25)
Q3 = df['km_driven'].quantile(0.75)
IQR =Q3-Q1
outlier_free_data =df[(df['km_driven']>=Q1 - 1.5 *IQR) &
(df['km_driven'] <= Q3 + 1.5 * IQR)]
print("Data after removing outliers:", outlier_free_data.shape[0])
```

```
Data after removing outliers: 1022
```

```
outlier_free_data
```

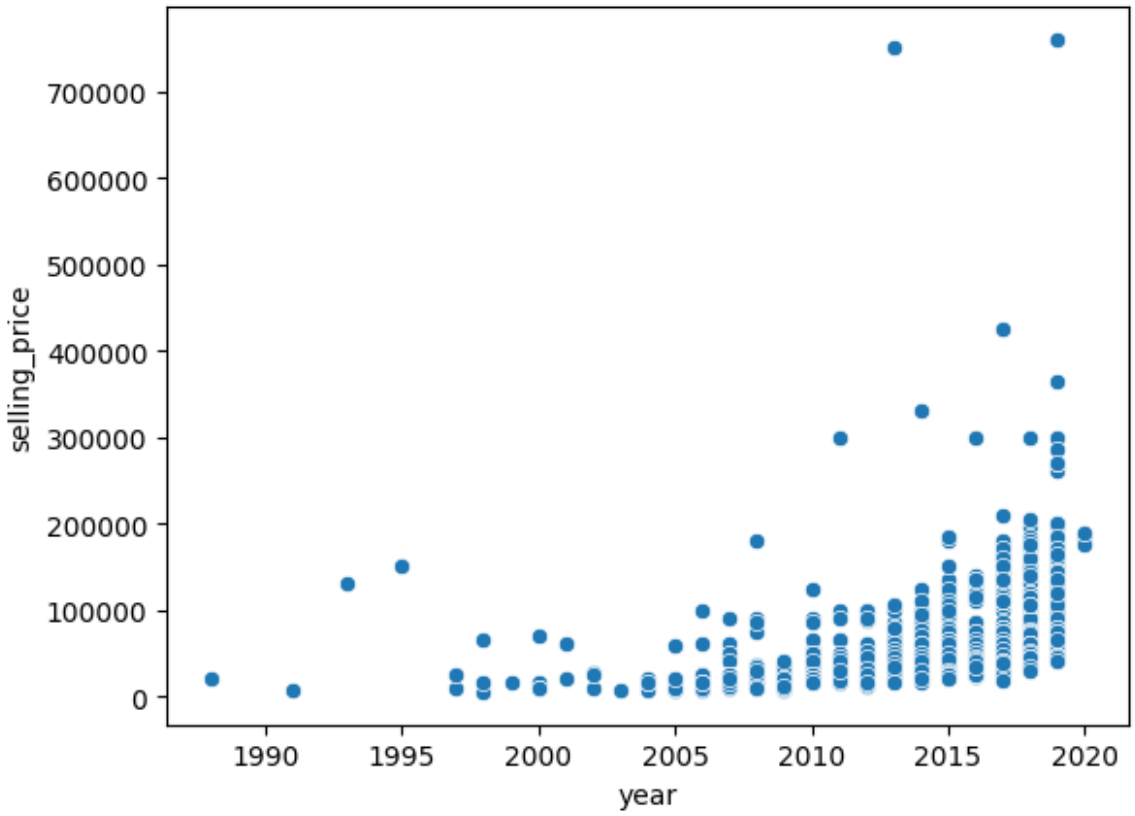

```
{"summary":"{\n  \"name\": \"outlier_free_data\",\n  \"rows\": 1022,\n
\"fields\": [\n    {\n      \"column\": \"name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 277,\n        \"samples\": [\n          \"Hero
Xtreme Sports\",\n          \"Yamaha Cygnus Ray ZR\",\n
\"Yamaha SZ-S\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"selling_price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 56868,\n        \"min\": 5000,\n
\"max\": 760000,\n        \"num_unique_values\": 127,\n
\"samples\": [\n          43000,\n          14500,\n          760000\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"year\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 4,\n
\"min\": 1988,\n        \"max\": 2020,\n        \"num_unique_values\":
28,\n        \"samples\": [\n          2012,\n          2003,\n
2020\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"seller_type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Dealer\",\n          \"Individual\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"owner\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"2nd owner\",\n          \"4th owner\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"km_driven\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
19552,\n        \"min\": 350,\n        \"max\": 86000,\n
\"num_unique_values\": 283,\n        \"samples\": [\n          42000,\
n          27370\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ex_showroom_price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 78560.06865368086,\n        \"min\":
30490.0,\n        \"max\": 1278000.0,\n        \"num_unique_values\":
229,\n        \"samples\": [\n          54000.0,\n          69983.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"outlier_free_data"}
```


#Q12.Perform a bivariate analysis to visualize the relationship between year and selling_price?

```
sns.scatterplot(x='year', y='selling_price', data=df)
```

```
<Axes: xlabel='year', ylabel='selling_price'>
```



#Q13.What is the average depreciation in selling price based on the bike's age (current year - manufacturing year)?

```
 # Average depreciation based on bike's age
current_year = 2024
df['age'] =current_year -  df['year']
avg_depreciation= df.groupby('age')['selling_price'].mean()
print("Average depreciation in selling price:\n", avg_depreciation)
```

```
Average depreciation in selling price:
 age
4     183333.333333
5     119689.511628
6      87660.374046
7      78894.736842
8      58469.018692
9      56500.000000
10     48668.131868
11     51136.986301
12     35748.400000
13     35655.721311
14     31793.333333
15     22267.857143
16     34289.285714
17     24927.586207
```

```
18     23380.000000
19     16978.571429
20     15100.000000
21      8000.000000
22     20666.666667
23     40000.000000
24     20833.333333
25     15000.000000
26     28333.333333
27     17500.000000
29    150000.000000
31    130000.000000
33      6000.000000
36     20000.000000
Name: selling_price, dtype: float64
```

#Q14.Which bike names are priced significantly above the average price for their manufacturing year?

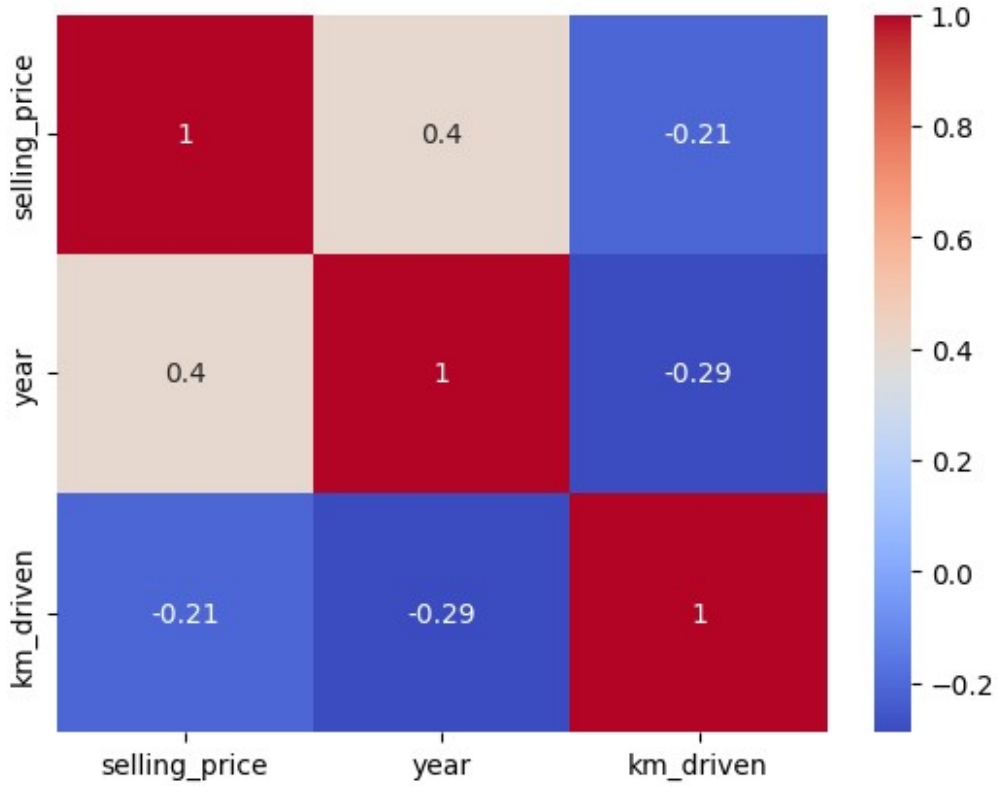
```
avg_price = df.groupby('year')['selling_price'].mean()
df[df['selling_price'] > 1.5 * df['year'].map(avg_price)][['name',
'year', 'selling_price']]
```


```
{"summary":"{\n  \"name\": \"'year', 'selling_price']]\",\n  \"rows\":
157,\n  \"fields\": [\n    {\n      \"column\": \"name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 66,\n        \"samples\": [\n          \"Royal
Enfield Electra Twinspark\",\n          \"KTM RC390\",\n
\"Royal Enfield Classic Gunmetal Grey\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"year\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 1998,\n
\"max\": 2019,\n        \"num_unique_values\": 17,\n
\"samples\": [\n          2018,\n          2008,\n          2017\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"selling_price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
94401,\n        \"min\": 35000,\n        \"max\": 760000,\n
\"num_unique_values\": 53,\n        \"samples\": [\n          138000,\
n          40000,\n          119000\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


#Q15.Develop a correlation matrix for numeric columns and visualize it using a heatmap?

```
import matplotlib.pyplot as plt
sns.heatmap(df[['selling_price', 'year',
'km_driven']].corr(),annot=True, cmap='coolwarm')
```

```
plt.show()
```



EDA-2 Car Sales

```
df1 =pd.read_csv("Car Sale.csv")
df1
```


```
{"summary":"{\n  \"name\": \"df1\",\n  \"rows\": 23906,\n  \"fields\":
[\n    {\n      \"column\": \"Car_id\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 23906,\n
\"samples\": [\n          \"C_CND_003394\",\n
\"C_CND_000517\",\n          \"C_CND_000537\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Date\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 612,\n
\"samples\": [\n          \"4/25/2022\",\n          \"10/15/2022\",\n
\"3/24/2022\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Customer Name\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3021,\n
\"samples\": [\n          \"Albertine\",\n          \"Orlane\",\n
\"Kelly\"\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Annual Income\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
720006,\n        \"min\": 10080,\n        \"max\": 11200000,\n
\"num_unique_values\": 2508,\n        \"samples\": [\n
5355000,\n          542000\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Dealer_Name\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 28,\n
\"samples\": [\n          \"Race Car Help\",\n          \"New Castle
Ford Lincoln Mercury\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Company\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 30,\n
\"samples\": [\n          \"Plymouth\",\n          \"Audi\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Model\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 154,\n        \"samples\": [\n
\"Forester\",\n          \"Breeze\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Engine\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
2,\n        \"samples\": [\n          \"Overhead Camshaft\",\n
\"Double\\u00c2\\u00a0Overhead Camshaft\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Transmission\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"Manual\",\n          \"Auto\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Color\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Black\",\n          \"Red\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price ($)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
14788,\n        \"min\": 1200,\n        \"max\": 85800,\n
\"num_unique_values\": 870,\n        \"samples\": [\n          13100,\
n          44000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Dealer_No \",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"06457-3834\",\n          \"60504-7114\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"Body Style\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Passenger\",\n          \"Sedan\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Phone\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 867491,\n
\"min\": 6000101,\n        \"max\": 8999579,\n
\"num_unique_values\": 23804,\n        \"samples\": [\n
8955220,\n          8396717\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Dealer_Region\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Middletown\",\n          \"Aurora\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df1"}
```


Q1. What is the average selling price of cars for each dealer, and how does it compare across different dealers?

```
import pandas as pd
file_path = '/mnt/data/Car Sale.csv'
data = pd.read_csv('Car Sale.csv')

# Calculate the average selling price for each dealer
avg_price_by_dealer = data.groupby('Dealer_Name')['Price
($)'].mean().sort_values(ascending=False)

# Display top 10 dealers by average price
print(avg_price_by_dealer.head(10))

summary_stats = avg_price_by_dealer.describe()
print(summary_stats)
```

```
Dealer_Name
U-Haul CO                                28769.919006
Classic Chevy                            28602.014446
Rabun Used Car Sales                     28527.536177
Iceberg Rentals                          28522.958533
Enterprise Rent A Car                    28312.580800
Scrivener Performance Engineering        28297.371589
Gartner Buick Hyundai Saab               28247.621019
Saab-Belle Dodge                         28190.139888
Capitol KIA                              28189.703822
Race Car Help                            28163.372706
```

```
Name: Price ($), dtype: float64
count       28.000000
mean     28048.527307
std        346.560928
min      27217.261563
25%      27863.777027
50%      28103.658625
75%      28204.510171
max      28769.919006
Name: Price ($), dtype: float64
```

#Q2. Which car brand (Company) has the highest variation in prices, and what does this tell us about the pricing trends?

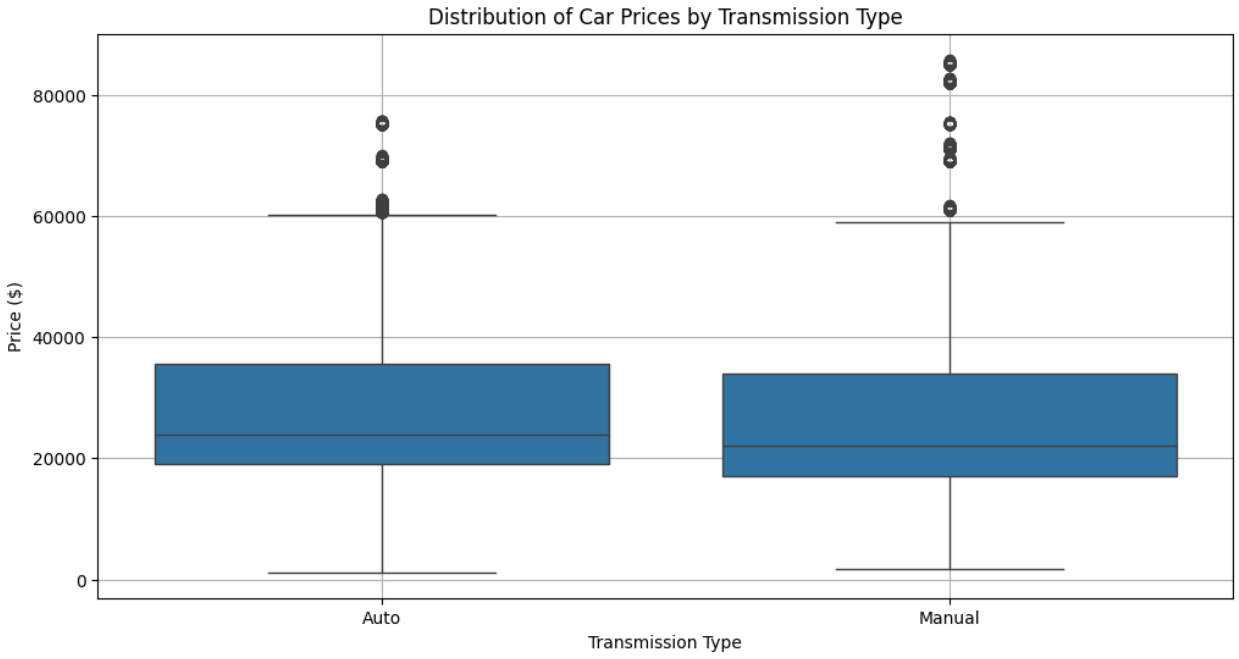
```
import pandas as pd
file_path = 'Car Sale.csv'
data = pd.read_csv('Car Sale.csv')
price_variation_by_company = data.groupby('Company')['Price
($)'].std().sort_values(ascending=False)
highest_variation = price_variation_by_company.idxmax()
highest_variation_value = price_variation_by_company.max()
top_10_variations = price_variation_by_company.head(10)
print("Top 10 Companies with Highest Price Variation:")
print(top_10_variations)
print(f"\nCompany with Highest Variation: {highest_variation} ($
{highest_variation_value:.2f})")
```

```
Top 10 Companies with Highest Price Variation:
Company
Lincoln     19658.050211
Saab        19653.740089
Cadillac    19517.120220
Plymouth    19065.997338
Lexus       17852.923492
Buick       17142.232626
Mercury     16445.172195
Nissan      16214.264017
Saturn      15990.223671
Ford        15849.090227
Name: Price ($), dtype: float64

Company with Highest Variation: Lincoln ($19658.05)
```

#Q3.What is the distribution of car prices for each transmission type, and how do the interquartile ranges compare?

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

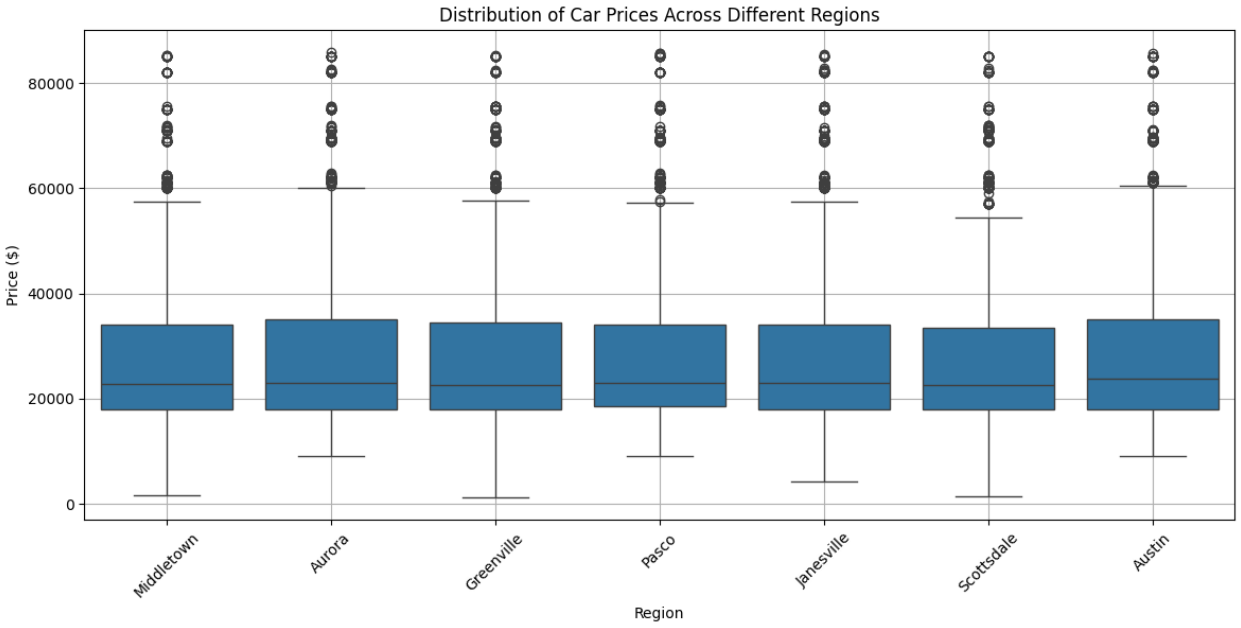
```
# Load the dataset
file_path = 'Car Sale.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# Boxplot for distribution of car prices by transmission type
plt.figure(figsize=(12, 6))
sns.boxplot(x='Transmission', y='Price ($)', data=data)
plt.title('Distribution of Car Prices by Transmission Type')
plt.xlabel('Transmission Type')
plt.ylabel('Price ($)')
plt.grid(True)
plt.show()

# Calculate the IQR for each transmission type
iqr_by_transmission = data.groupby('Transmission')['Price ($)'].agg(
    Q1=lambda x: x.quantile(0.25),
    Q3=lambda x: x.quantile(0.75),
    IQR=lambda x: x.quantile(0.75) - x.quantile(0.25)
)

print(iqr_by_transmission)
```




```
                    Q1       Q3      IQR
Transmission
Auto           19000.0  35500.0  16500.0
Manual         17000.0  34000.0  17000.0
```

#Q4.What is the distribution of car prices across different regions?

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
file_path = 'Car Sale.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# Boxplot for distribution of car prices by region
plt.figure(figsize=(14, 6))
sns.boxplot(x='Dealer_Region', y='Price ($)', data=data)
plt.title('Distribution of Car Prices Across Different Regions')
plt.xlabel('Region')
plt.ylabel('Price ($)')
plt.xticks(rotation=45)  # Rotate region labels for readability
plt.grid(True)
plt.show()

# Summary statistics for each region
summary_by_region = data.groupby('Dealer_Region')['Price
($)'].describe()

print(summary_by_region)
```

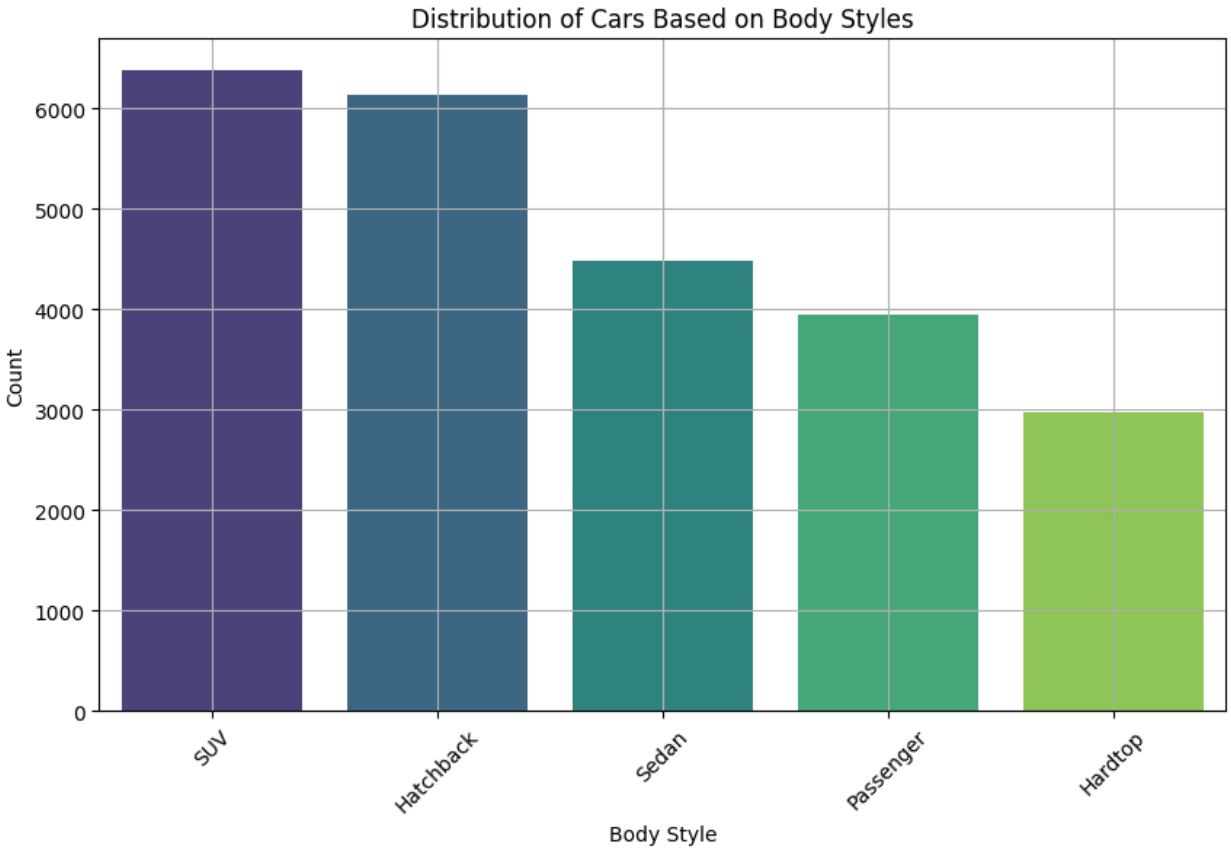



```
                count          mean           std     min      25%
50%  \
Dealer_Region

Aurora         3130.0  28334.626837  15026.207252  9000.0  18001.0
23000.0
Austin         4135.0  28341.603628  14903.884549  9000.0  18001.0
23801.0
```

```
Greenville     3128.0  28180.819054  15101.538328  1200.0  18001.0
22500.0
Janesville     3821.0  27833.350955  14344.995638  4300.0  18001.0
23000.0
Middletown     3128.0  27856.338875  14619.842395  1700.0  18000.0
22750.0
Pasco          3131.0  28119.039923  14659.315941  9000.0  18500.5
23000.0
Scottsdale     3433.0  27954.958928  14902.916820  1450.0  18000.0
22600.0

                    75%      max
Dealer_Region
Aurora          35000.0  85800.0
Austin          35001.0  85601.0
Greenville      34500.0  85200.0
Janesville      34000.0  85400.0
Middletown      34000.0  85300.0
Pasco           34000.0  85600.0
Scottsdale      33500.0  85001.0
```

#Q5.What is the distribution of cars based on body styles?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# Count the number of cars in each body style category
body_style_counts = data['Body Style'].value_counts()

# Plot the distribution of body styles
plt.figure(figsize=(10, 6))
sns.barplot(x=body_style_counts.index, y=body_style_counts.values,
palette='viridis')
plt.title('Distribution of Cars Based on Body Styles')
plt.xlabel('Body Style')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotate labels for readability
plt.grid(True)
plt.show()

# Display the counts
print(body_style_counts)
```

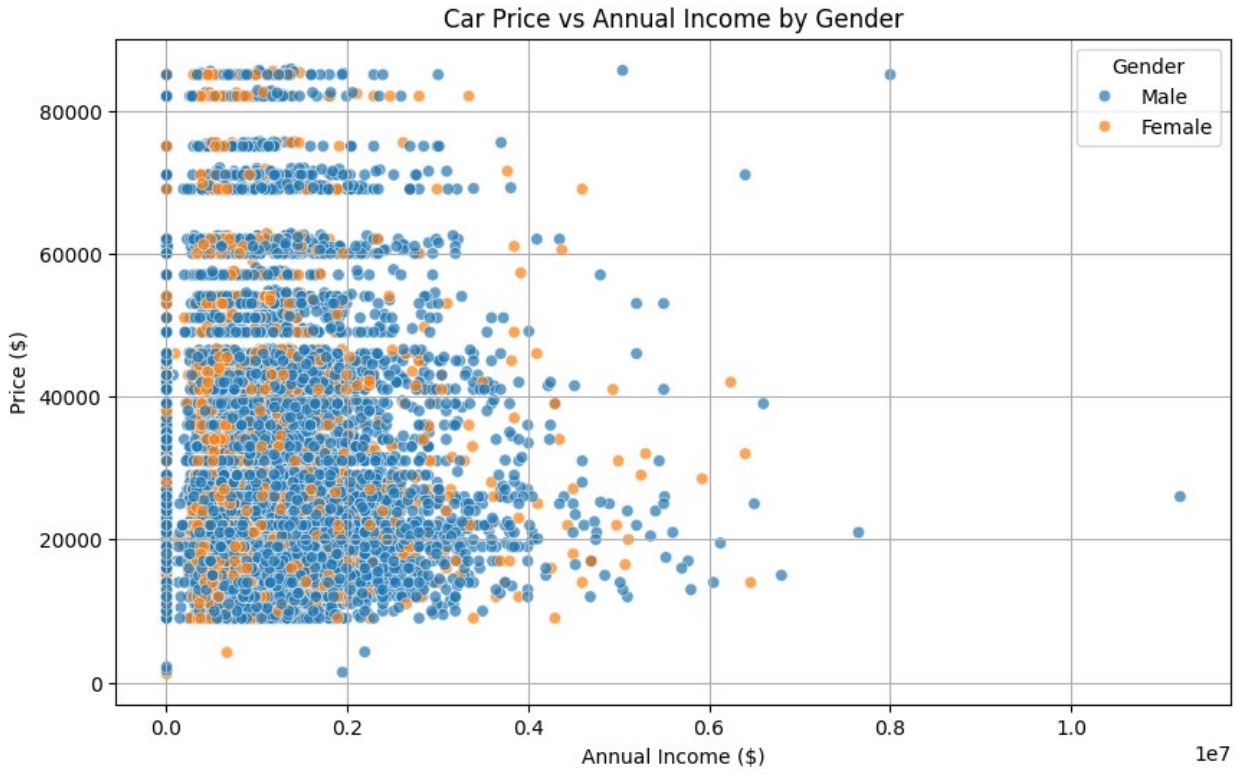



```
Body Style
SUV          6374
Hatchback    6128
Sedan        4488
Passenger    3945
Hardtop      2971
Name: count, dtype: int64
```

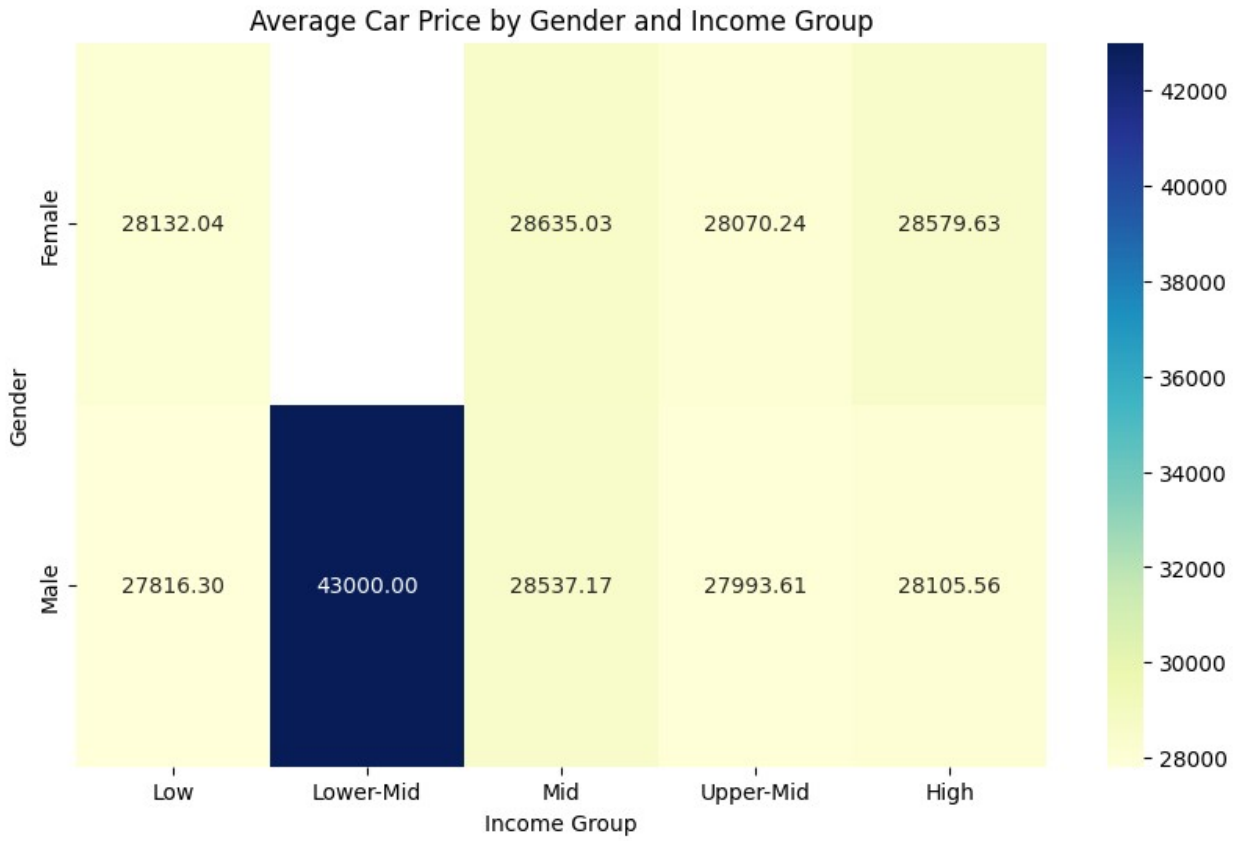
#Q6.How does the average selling price of cars vary by customer gender and annual income?

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
file_path = 'Car Sale.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# Scatter plot for Price vs Annual Income, colored by Gender
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Annual Income', y='Price ($)',
hue='Gender', alpha=0.7)
plt.title('Car Price vs Annual Income by Gender')
plt.xlabel('Annual Income ($)')
plt.ylabel('Price ($)')
```

```
plt.grid(True)
plt.legend(title='Gender')
plt.show()

# Calculate average price by gender
avg_price_by_gender = data.groupby('Gender')['Price ($)'].mean()
print("Average Price by Gender:")
print(avg_price_by_gender)

# Analyze price trends based on income groups
data['Income Group'] = pd.cut(data['Annual Income'], bins=[0, 50000,
100000, 500000, 1000000, data['Annual Income'].max()],
                              labels=['Low', 'Lower-Mid', 'Mid',
'Upper-Mid', 'High'])

# Average price by gender and income group
avg_price_by_gender_income = data.groupby(['Gender', 'Income Group'])
['Price ($)'].mean().unstack()
print("\nAverage Price by Gender and Income Group:")
print(avg_price_by_gender_income)
# Heatmap for visualization
plt.figure(figsize=(10, 6))
sns.heatmap(avg_price_by_gender_income, annot=True, fmt=".2f",
cmap='YlGnBu')
plt.title('Average Car Price by Gender and Income Group')
plt.xlabel('Income Group')
plt.ylabel('Gender')
plt.show()
```



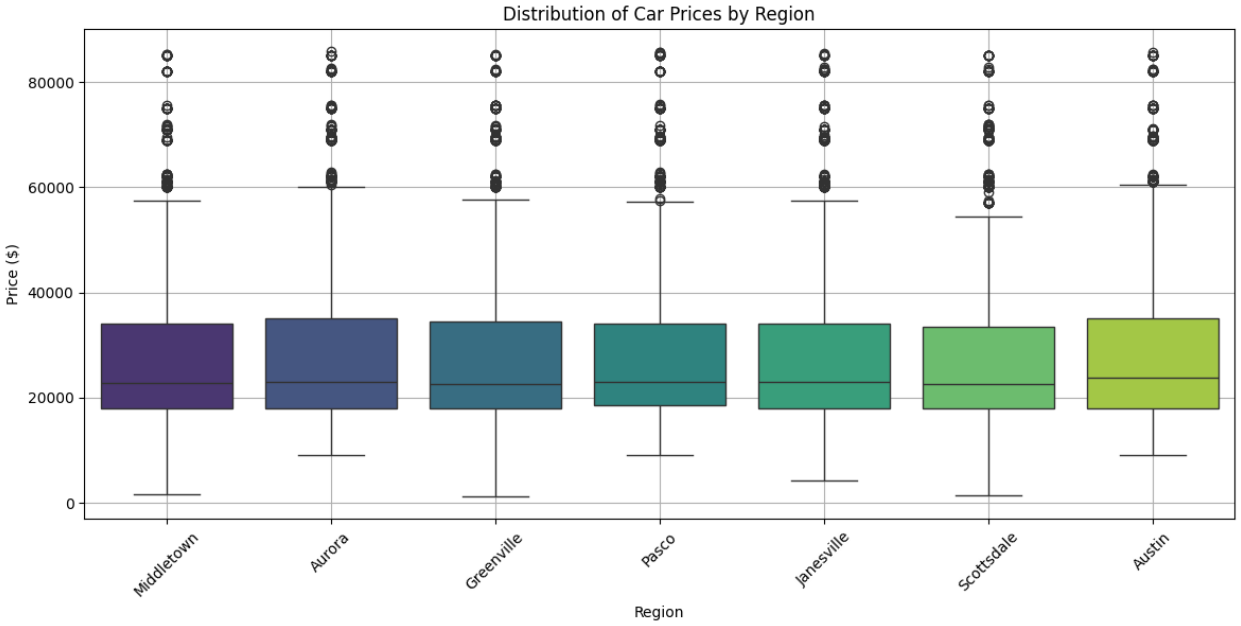

```
Average Price by Gender:
Gender
Female    28277.265270
Male      28039.429407
Name: Price ($), dtype: float64

Average Price by Gender and Income Group:
Income Group           Low  Lower-Mid           Mid     Upper-Mid  \
Gender
Female        28132.038732        NaN  28635.027119  28070.242135
Male          27816.302247    43000.0  28537.169450  27993.611332

Income Group          High
Gender
Female        28579.626947
Male          28105.557471
```

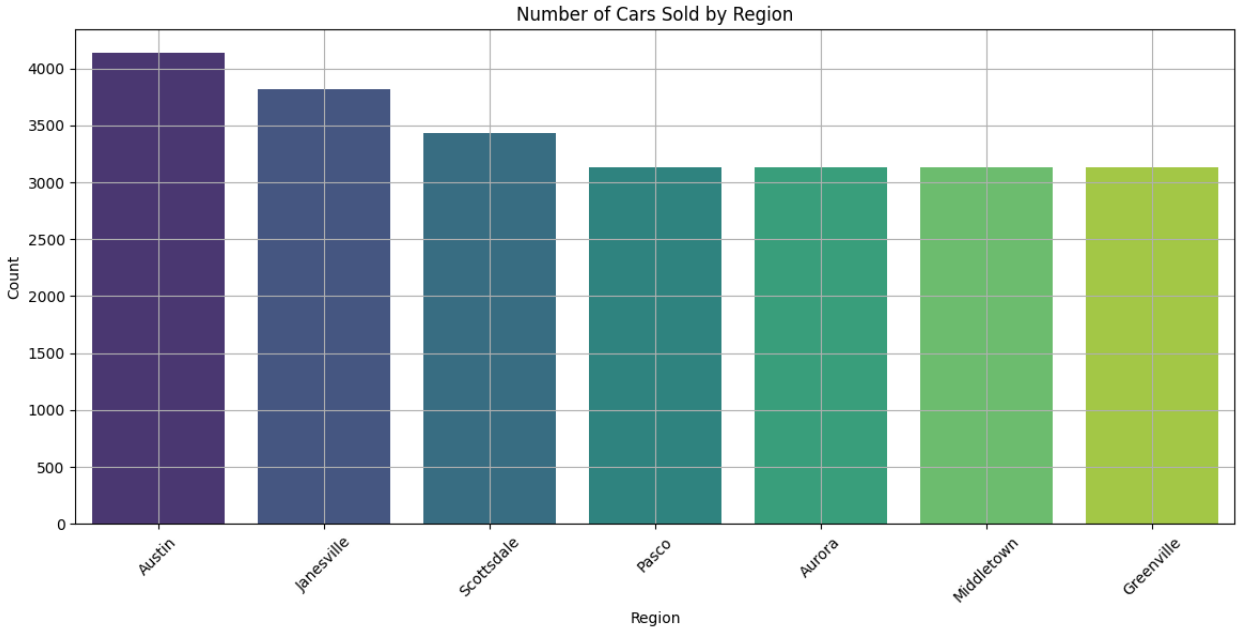


#Q7.What is the distribution of car prices by region, and how does the number of cars sold vary by region?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with your file path
data = pd.read_csv(file_path)

# 1. Boxplot for distribution of car prices by region
plt.figure(figsize=(14, 6))
sns.boxplot(x='Dealer_Region', y='Price ($)', data=data,
palette='viridis')
plt.title('Distribution of Car Prices by Region')
plt.xlabel('Region')
plt.ylabel('Price ($)')
plt.xticks(rotation=45)  # Rotate labels for readability
plt.grid(True)
plt.show()

# 2. Bar plot for number of cars sold by region
region_counts = data['Dealer_Region'].value_counts()

plt.figure(figsize=(14, 6))
```

```
sns.barplot(x=region_counts.index, y=region_counts.values,
palette='viridis')
plt.title('Number of Cars Sold by Region')
plt.xlabel('Region')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# Print counts of cars sold by region
print("Number of Cars Sold by Region:")
print(region_counts)
```






```
Number of Cars Sold by Region:
Dealer_Region
Austin        4135
Janesville    3821
Scottsdale    3433
Pasco         3131
Aurora        3130
Middletown    3128
Greenville    3128
Name: count, dtype: int64
```

#Q8. How does the average car price differ between cars with different engine sizes?

```
import pandas as pd
file_path = 'Car Sale.csv'  # Replace this with the actual path to
your CSV file
data = pd.read_csv(file_path)

# Group data by 'Engine' and calculate average price
engine_price = data.groupby('Engine')['Price
($)'].mean().reset_index()

# Sort results by average price in descending order
engine_price_sorted = engine_price.sort_values(by='Price ($)',
ascending=False)

# Display the result
print(engine_price_sorted)
```

```
                       Engine     Price ($)
0  DoubleÂ Overhead Camshaft  28248.525972
1           Overhead Camshaft  27914.710631
```

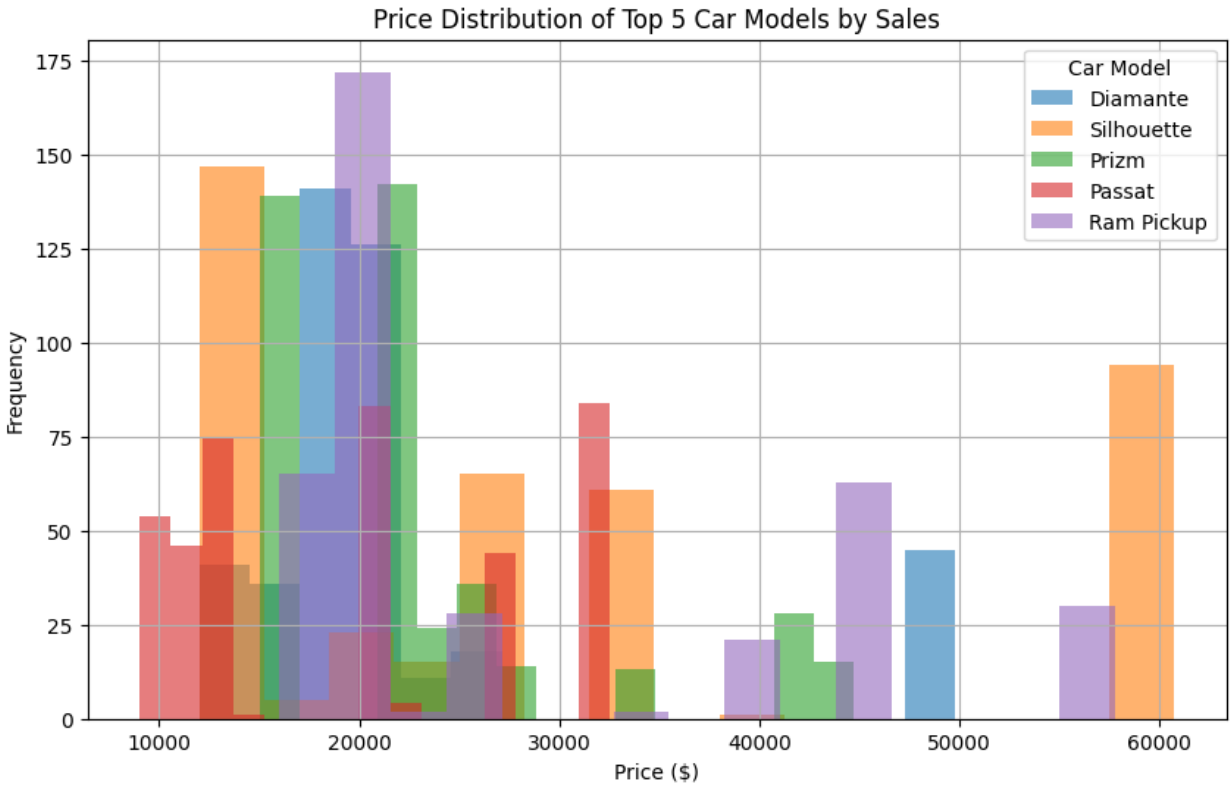
#Q9.How do car prices vary based on the customer's annual income bracket?

```
import pandas as pd
file_path = 'Car Sale.csv'  # Replace this with the correct path to
your file
data = pd.read_csv(file_path)

# Define income brackets
bins = [0, 25000, 50000, 100000, 500000, 1000000, float('inf')]
labels = ['0-25k', '25k-50k', '50k-100k', '100k-500k', '500k-1M',
'1M+']

# Create income brackets
data['Income Bracket'] = pd.cut(data['Annual Income'], bins=bins,
labels=labels)

# Group data by income bracket and calculate average car price
income_price = data.groupby('Income Bracket')['Price
($)'].mean().reset_index()

# Sort results by income bracket
income_price_sorted = income_price.sort_values(by='Income Bracket')

# Display the result
print(income_price_sorted)
```

```
  Income Bracket     Price ($)
0          0-25k  27884.297820
1        25k-50k           NaN
2       50k-100k  43000.000000
3      100k-500k  28563.329860
4        500k-1M  28011.726423
5            1M+  28186.202040

<ipython-input-2-a92f6b476637>:13: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  income_price = data.groupby('Income Bracket')['Price
($)'].mean().reset_index()
```

#Q10.What are the top 5 car models with the highest number of sales, and how does their price distribution look?

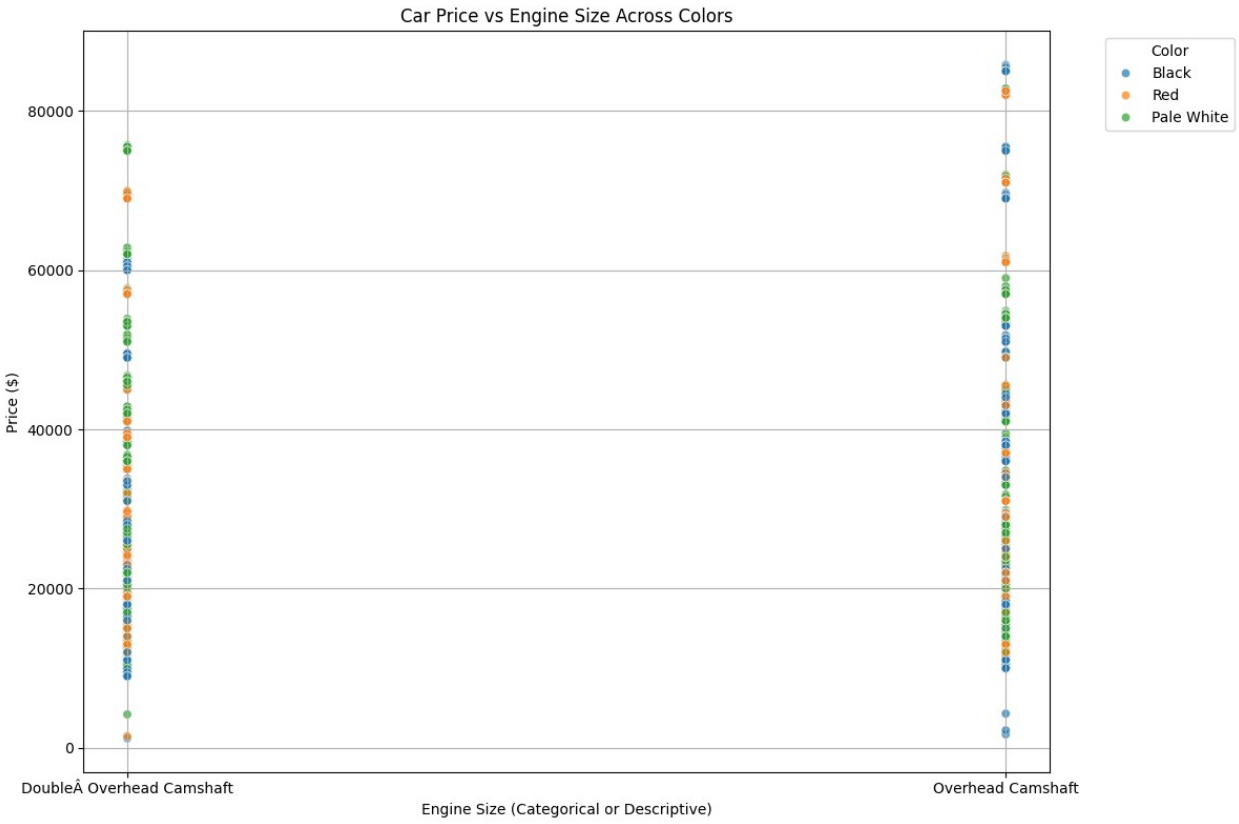
```python
import pandas as pd
import matplotlib.pyplot as plt
file_path = 'Car Sale.csv'  # Replace with the actual file path
data = pd.read_csv(file_path)

# Find the top 5 car models with the highest sales count
top_models = data['Model'].value_counts().nlargest(5).index

# Filter data for only the top 5 models
top_models_data = data[data['Model'].isin(top_models)]

# Plot price distribution for each top model
plt.figure(figsize=(10, 6))
for model in top_models:
    subset = top_models_data[top_models_data['Model'] == model]
    plt.hist(subset['Price ($)'], bins=15, alpha=0.6, label=model)

plt.title('Price Distribution of Top 5 Car Models by Sales')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')
plt.legend(title='Car Model')
plt.grid(True)
plt.show()
```



#Q11. How does car price vary with engine size across different car colors, and which colors have the highest price variation?

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with the actual file path
data = pd.read_csv(file_path)

# Group data by Color and calculate price variation (standard
deviation)
color_price_variation = data.groupby('Color')['Price
($)'].std().reset_index()
color_price_variation = color_price_variation.sort_values(by='Price
($)', ascending=False)

# Display the top colors with highest price variation
print("Top colors with highest price variation:")
print(color_price_variation.head(10))

# Plot price vs. engine size for different colors
plt.figure(figsize=(12, 8))
sns.scatterplot(data=data, x='Engine', y='Price ($)', hue='Color',
alpha=0.7)

plt.title('Car Price vs Engine Size Across Colors')
plt.xlabel('Engine Size (Categorical or Descriptive)')
plt.ylabel('Price ($)')
plt.legend(title='Color', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```

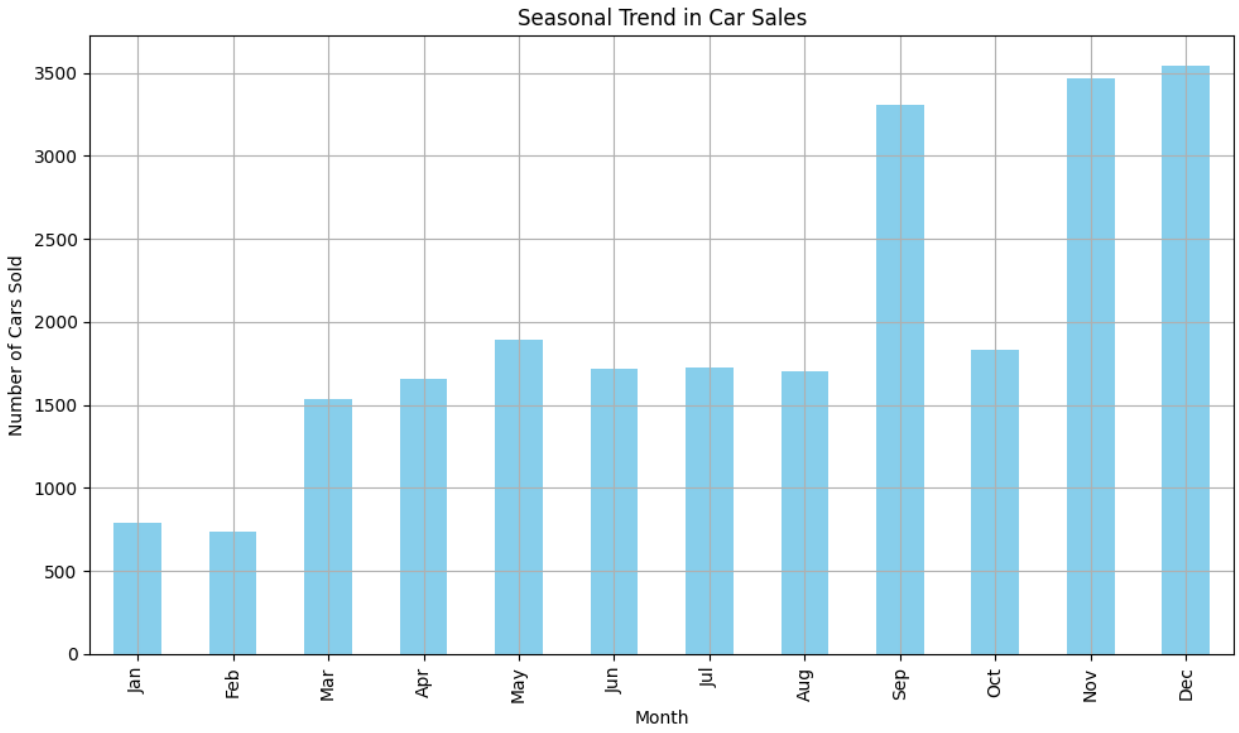
```
Top colors with highest price variation:
        Color     Price ($)
2         Red  15519.360962
0       Black  15286.065976
1  Pale White  14077.346859
```



#Q12.Is there any seasonal trend in car sales based on the date of sale?

```
import pandas as pd
import matplotlib.pyplot as plt
file_path = 'Car Sale.csv'  # Replace with the actual file path
data = pd.read_csv(file_path)

# Convert the 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'])

# Extract month and year
data['Month'] = data['Date'].dt.month
data['Year'] = data['Date'].dt.year

# Group data by month to analyze seasonal trends
monthly_sales = data.groupby('Month')['Car_id'].count()

# Plot seasonal trends
plt.figure(figsize=(10, 6))
monthly_sales.plot(kind='bar', color='skyblue')
plt.title('Seasonal Trend in Car Sales')
plt.xlabel('Month')
plt.ylabel('Number of Cars Sold')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May',
'Jun',
```

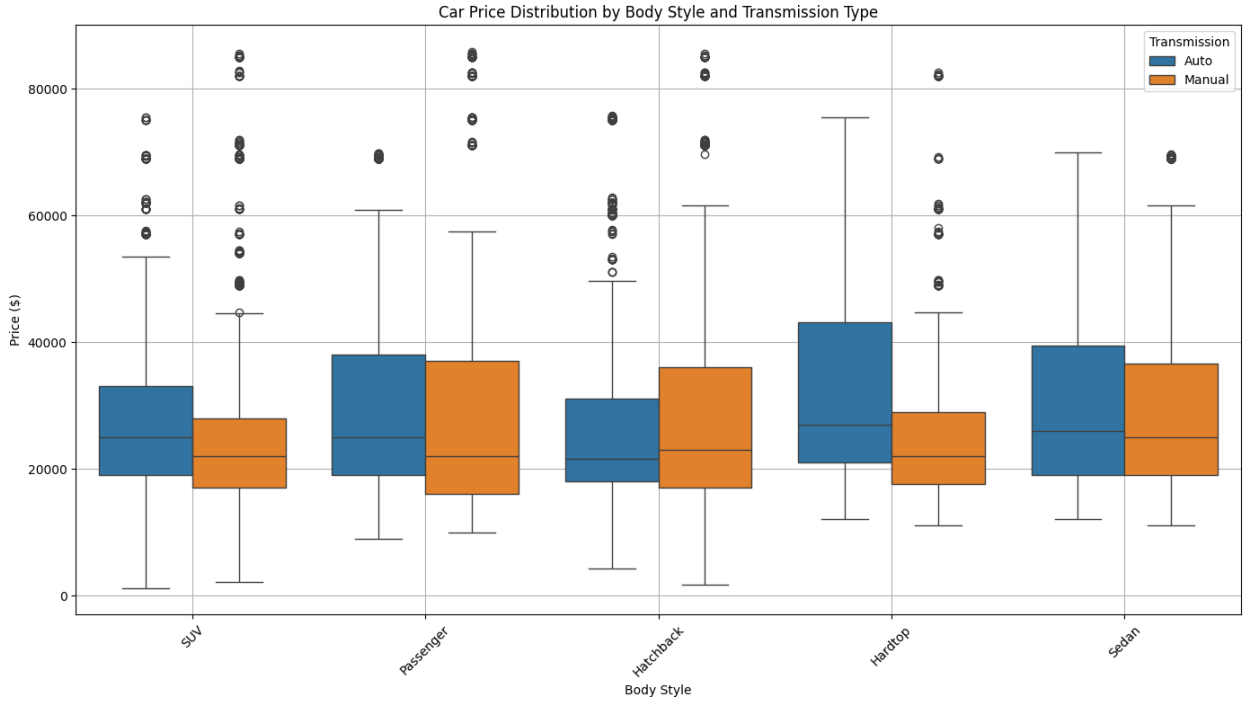
```
                                        'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
'Dec'])
plt.grid(True)
plt.tight_layout()
plt.show()
```



#Q13. How does the car price distribution change when considering different combinations of body style and transmission type?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with the actual file path
data = pd.read_csv(file_path)

# Create a grouped boxplot to visualize price distribution
plt.figure(figsize=(14, 8))
sns.boxplot(data=data, x='Body Style', y='Price ($)',
hue='Transmission')

plt.title('Car Price Distribution by Body Style and Transmission
Type')
plt.xlabel('Body Style')
plt.ylabel('Price ($)')
plt.xticks(rotation=45)  # Rotate labels for better readability
plt.legend(title='Transmission')
plt.grid(True)
```

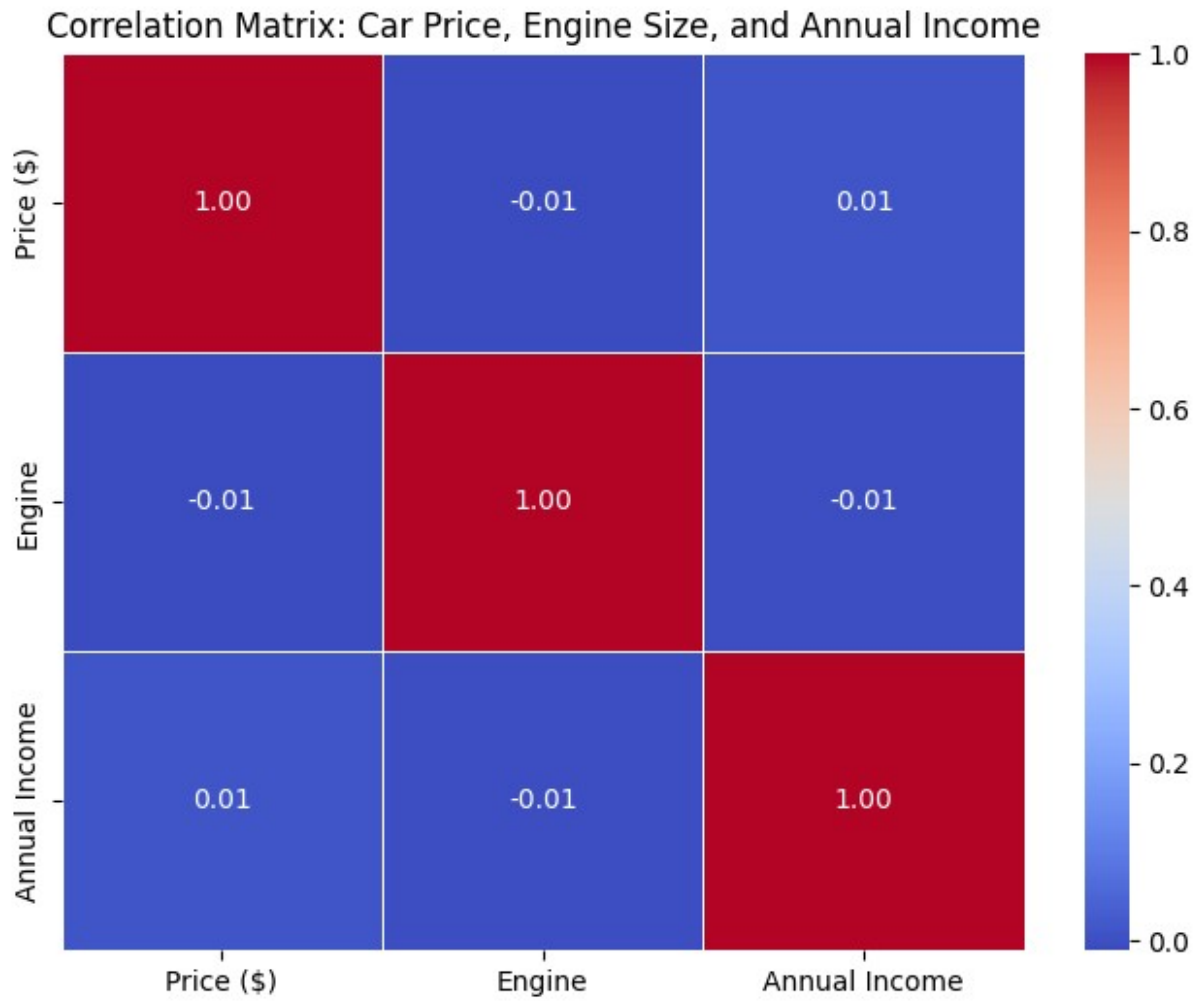
```
plt.tight_layout()
plt.show()
```

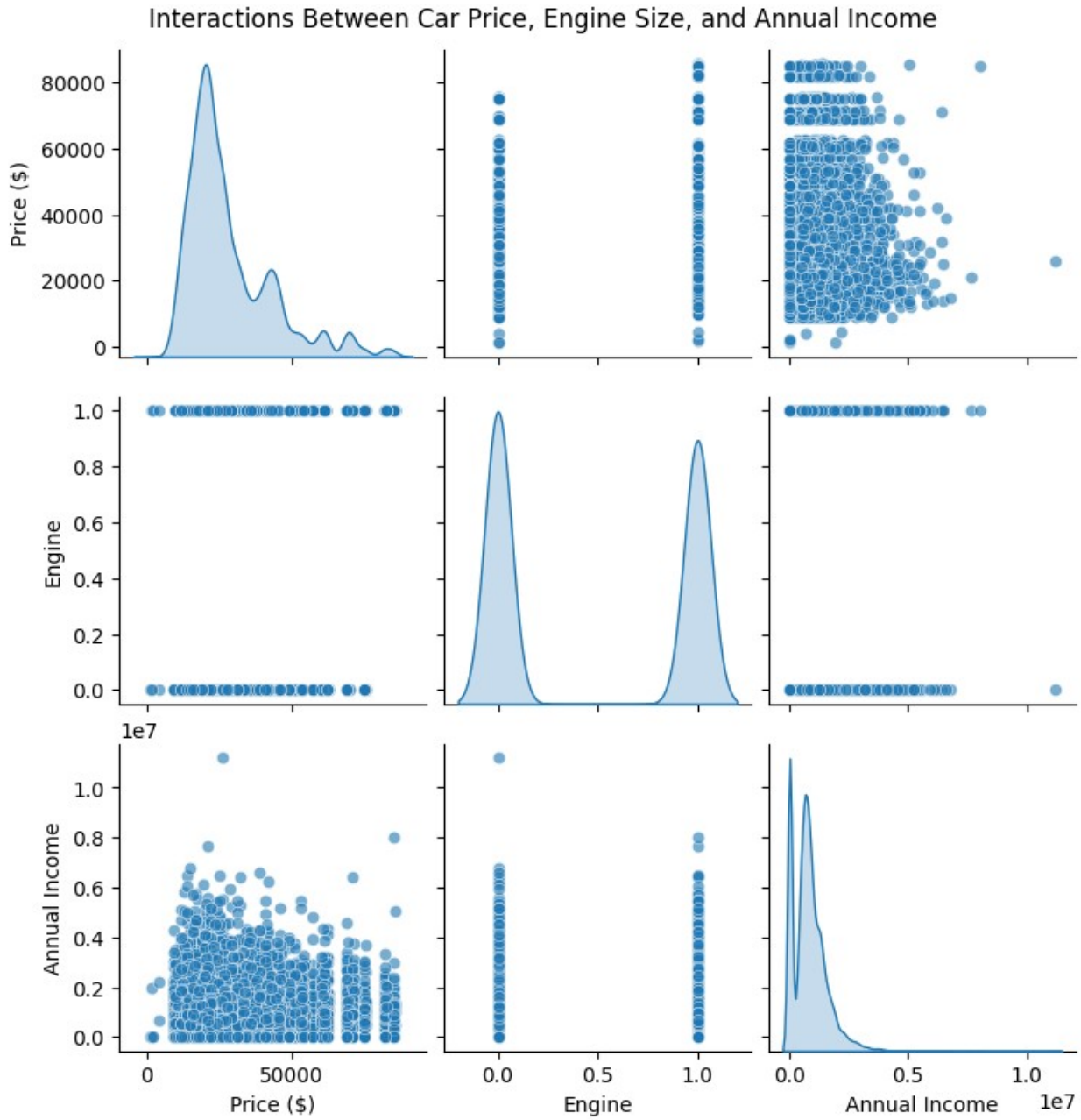


#Q14.What is the correlation between car price, engine size, and annual income of customers, and how do these features interact?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with the correct file path
data = pd.read_csv(file_path)

# Check and clean the dataset
data = data[['Price ($)', 'Engine', 'Annual Income']].dropna()  #
Select relevant columns and drop missing values

# Encode 'Engine' size if it is categorical
if data['Engine'].dtype == 'object':
    data['Engine'] = data['Engine'].astype('category').cat.codes  #
Convert to numeric codes if needed

# Compute correlation matrix
correlation = data.corr()

# Display correlation matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
```

```
plt.title('Correlation Matrix: Car Price, Engine Size, and Annual
Income')
plt.show()

# Pairplot for interaction visualization
sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.6})
plt.suptitle('Interactions Between Car Price, Engine Size, and Annual
Income', y=1.02)
plt.show()
```

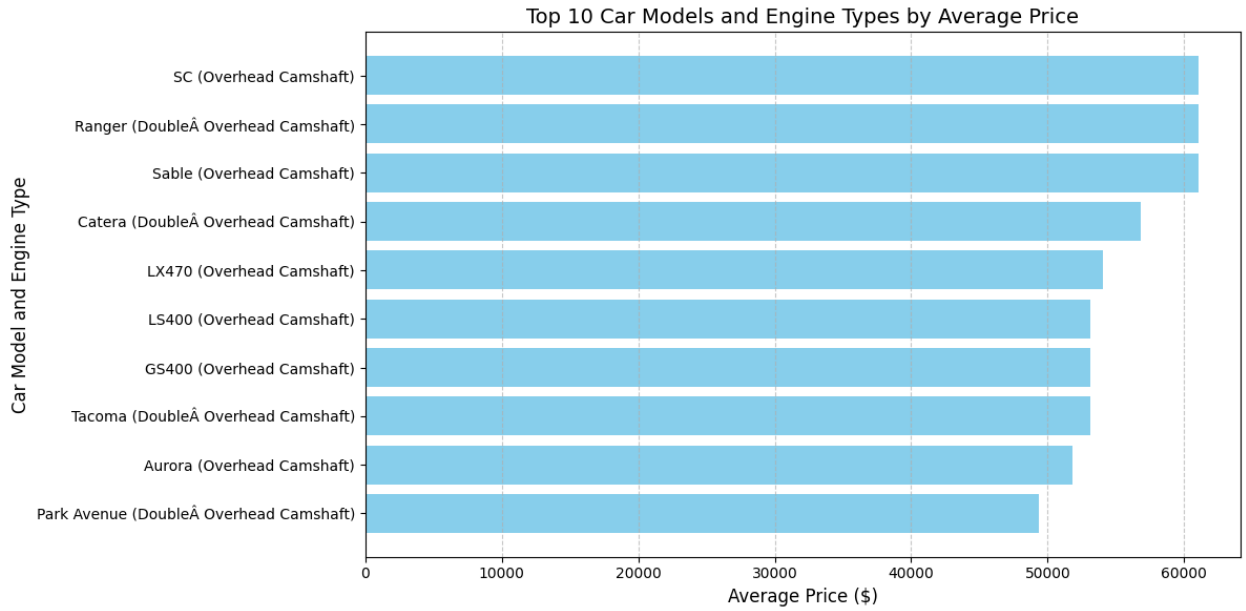




#Q15. How does the average car price vary across different car models and engine types?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
file_path = 'Car Sale.csv'  # Replace with the correct file path
data = pd.read_csv(file_path)

# Calculate average price for each model and engine type
avg_price_by_model_engine = data.groupby(['Model', 'Engine'])['Price
($)'].mean().reset_index()
```

```
# Sort by average price in descending order
avg_price_by_model_engine_sorted =
avg_price_by_model_engine.sort_values(by='Price ($)', ascending=False)

# Plot the top 10 car models and engine types by average price
top_avg_prices = avg_price_by_model_engine_sorted.head(10)

plt.figure(figsize=(12, 6))
plt.barh(top_avg_prices['Model'] + " (" + top_avg_prices['Engine'] +
")",
         top_avg_prices['Price ($)'], color='skyblue')

plt.xlabel('Average Price ($)', fontsize=12)
plt.ylabel('Car Model and Engine Type', fontsize=12)
plt.title('Top 10 Car Models and Engine Types by Average Price',
fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



EDA - 3 AMAZON SALES

```
df2=pd.read_csv('amazon.csv')
df2
```


```
{"summary":"{\n  \"name\": \"df2\",\n  \"rows\": 1465,\n  \"fields\":
[\n    {\n      \"column\": \"product_id\",\n      \"properties\": {\n
```

```
\"dtype\": \"string\",\n        \"num_unique_values\": 1351,\n
\"samples\": [\n          \"B09GFLXVH9\",\n          \"B0BC9BW512\",\n
\"B097JVLW3L\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"product_name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 1337,\n
\"samples\": [\n          \"Glun Multipurpose Portable Electronic
Digital Weighing Scale Weight Machine (10 Kg - with Back Light)\",\n
\"Akiara\\u00ae - Makes life easy Mini Sewing Machine with Table Set |
Tailoring Machine | Hand Sewing Machine with extension table, foot
pedal, adapter\",\n          \"TTK Prestige Limited Orion Mixer
Grinder 500 Watts, 3 Jars (1200ml, 1000ml, 500ml) (Red)\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 211,\n        \"samples\": [\n
\"Electronics|Mobiles&Accessories|MobileAccessories|
Photo&VideoAccessories|SelfieSticks\",\n          \"Home&Kitchen|
Kitchen&HomeAppliances|Coffee,Tea&Espresso|CoffeeGrinders|
ElectricGrinders\",\n          \"Computers&Accessories|
Accessories&Peripherals|HardDriveAccessories|Caddies\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"discounted_price\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 550,\n        \"samples\": [\n          \"\\
u20b920,999\",\n          \"\\u20b91,699\",\n          \"\\u20b9419\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"actual_price\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 449,\n
\"samples\": [\n          \"\\u20b93,210\",\n          \"\\
u20b91,129\",\n          \"\\u20b94,500\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"discount_percentage\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 92,\n        \"samples\": [\n
\"86%\",\n          \"72%\",\n          \"26%\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
28,\n        \"samples\": [\n          \"3.6\",\n          \"3\",\n
\"3.3\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_count\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 1143,\n
\"samples\": [\n          \"197\",\n          \"7,945\",\n
\"1,40,036\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"about_product\",\n      \"properties\": {\n        \"dtype\":
```

\"string\",\n   \"num_unique_values\": 1293,\n
\"samples\": [\n   \"Advanced Bluetooth calling: Upgrade to an effortless calling experience - attend/reject calls and dial numbers, from your wrist.;Digital crown: Navigate through the watch, adjust volume and change the watch face via the fully-functional crown.|1.72\\u2019\\u2019display: ColorFit Pro 4 features 1.72\\u2019\\u2019 TFT LCD with 25% more screen area than ColorFit Pro 3.;Vivid clarity: View information under the brightest sun, thanks to 311 PPI and 500 nits of brightness.|60Hz refresh rate: Get smoother scrolling & navigation experience.;100 sports modes: Take your pick from 100 sports modes and ace your game.|Noise Health Suite: Know how your body is doing with the battery of fitness features.;Productivity suite: Get more work done with quick reply options, stock market updates, alarm and disconnect with smart DND when you want to.; Water Resistance Level: Water Resistant|Item Type Name: Smartwatch; Connectivity Technology: Usb; Included Components: \\u200eSmartwatch, Magnetic Charger, User Manual, Warranty Card\",\n   \"Fire-Boltt is India' No 1 Wearable Watch Brand Q122 by IDC Worldwide quarterly wearable device tracker Q122.\\u30101.69\\u201d HD Large Touch Screen\\u3011- Fire-Boltt Ninja 3 comes with a 1.69\\u201d HD Full Touch Display for smooth swipes and clear vision;\\u3010SPO2/ Oxygen, Heart Rate\\u3011 - Fire-Boltt Ninja 3 Smartwatch comes with real time 24*7 SPO2 / Blood Oxygen tracking, Dynamic Heart Rate Monitoring (If a patient is suffering from Covid 19 please use a medical device prescribed by the Doctor)|\\u301060 workout modes\\u3011- This smartwatch consists of 60 sports mode to track. Keep a track of all your activities and compare history to analyse your performance. Count steps, distance, and calories burned.;\\u3010IP68 Water Resistant\\u3011- This smartwatch can withstand dust, spills, raindrops and is sweatproof too|\\u3010POWERFUL BATTERY\\u3011 - About 7 days battery life and a Standby Time of 25 Days \\u3010Multiple Watch Faces\\u3011- Unlimited Customized Built in Watch Faces and also multiple watch faces through the app;\\u3010Stay Social Stay Updated\\u3011 \\u2013 Inbuilt Social Media Notifications.|\\u3010All In One Smart Coach\\u3011 - Track your Daily Steps, Sleep, Fitness, Sports, Heart Rate and SPO2 \\u3010Enjoy Music And Camera Control\\u3011 \\u3010IP68 Water Resistant\\u3011- This smartwatch can withstand dust, spills, raindrops and is sweatproof too; Water Resistance Level: water_resistant|Connectivity Technology: Bluetooth; Clasp Type: Tang Buckle; Compatible Devices: Smartphonetablet; Human Interface Input: Touch Screenbuttons; Item Type Name: Smart Watch; Included Components: 1 Smartwatch, 1 Manual, 1 Magnetic Charger, 1 Warranty Card; Band Color: Green; Band Material Type: Silicone; Case Material Type: Plastic; Color Name: Green\",\n   \"Keyboard : Standard keyboard|Rupee key, Comfortable|Silent Durable keys|Mouse : Ergonomic design, Accurate optical sensor|High resolution enabling faster navigation\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"user_id\",\n   \"properties\": {\n   \"dtype\": \"string\",\

```
n        \"num_unique_values\": 1194,\n        \"samples\": [\n
\"AEWW4LJOVXD65UKE7QCBCHQZMG7A,AGVWB5YTQZC7GAIV4PCCOFF2U27A,AFUR2THG6B
YV6IRA5JV6LBQNG2AA,AFSG5TXKKCLHBK3FABKJABBBUHEQ,AF2DOUVTY5LHTVWGGVE6YH
W5KEGA,AFEX5OM5UOST6POIWTBW6TCEZ2YA,AFKZZOQ7J2SOXP3OHFBEDXNFINCQ,AFD6P
5IRXY6KWXUW4H7X6ECRMSLA\",\n
\"AG65C34LATM4J3ZFKJJPDNISZKUQ,AG76GICZHJGA7YVN4TORX36ONVYA,AHHIHCEKEY
DIRPJ5W7WXGLB3E66Q,AFYSF6635O2EAPR4GMVBH74FSIFQ,AHAVRPA7Z3PKTTWVBVUISC
KI7RYQ,AEDH674UH53A5FKLUZCCM5LVKUQQ,AEUK344UA4FNU4PR4AWSPKWX5PPQ,AGPAK
6ELVZPVKQ7GEZ7IUHNK2C3Q\",\n
\"AE3S2ZAEMH765KUJ57DR6HBZBB3Q,AHSIVUNTJMI5S5AJGFDE5EDQ355Q,AGQUDHVCMB
W7DYS2HT5HA3QCZIEA,AFNXZNINQLTHKVRFI37VQAAFFGOA,AGVKQNHNS7PQK63FIB6EVC
5GUAMQ,AEHZ4NOZ5SIDQLGODWS4UZ6RVQJA,AGBVAEUPMWYCDQIKNAD2DXEYWXZA,AEOJW
7OWUZROZ6Z66ZQU33Y2ZYLQ\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"user_name\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 1194,\n
\"samples\": [\n          \"Kindle Customer,Aryan,pooja reddy,Amazon
Customer,Meenakshi jasrotia,imamthulla,Anan,Sanjay Chavan\",\n
\"zain,Deepak,VIMAL,Shiv Sagar,Tamil selvan,Rakesh yadav,PAGOLA
SURESH,Olivia\",\n          \"Fardeen mujawar,Pavan,Danny,Siddhartha
Pratap,Rabindra Kumar Das,Amazon Customer,Rakesh Ranga Yadav,Nivedita
Chatterjee\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"review_id\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 1194,\n
\"samples\": [\n
\"R1T4TKPYU5EJCB,R1D38AX8G0RVNS,R1KHCRDEEREQG7,R396UL83OTSD8F,R3CY781P
K5CB8A,RBCCWRI4IUHH5,R2K7JYQMGQ31YJ,R3P0GJ4V5HPF2M\",\n
\"R2JCUKBR0BQ8ES,RNVX0V6SJF3CP,RW5MJG9LTX6QD,R37PSG13H70Z1F,R17RIHK0XX
QDH5,R2P187SBO4SEMH,R1V49G7PD8Y93G,RU78E5A4MW0PK\",\n
\"RD6OIJUG0R241,R3EUJ7A6LG8X7V,R1DWGT4USEVGYK,R187KH5XJBPS86,R2XYH31E9
NK0GU,RDYNZZPHU7SZK,R2MR0DYZVFN3HA,R3PV91U8ZYN5DU\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1194,\n        \"samples\": [\n
\"Clearly makes a difference,Good,Value for money,Good material,The
ink of parker is very lite,Good,Good,Very good\",\n          \"Good
product with less money,At this price ok ok.,Good product,Good mouse
at this price range,Good,Good for daily use ke liye,Good,Good\",\n
\"Ok,Like all other ball pens,Regular pen over priced,Nice,It is
fine.,Awful blue ink,Nice and my Favorite Pen,Reasonable price\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"review_content\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1212,\n        \"samples\": [\n
\"Reviewing just after a day of using this product. We made French
fries and chicken tikka and result is quiet impressive! The recipe
book and cooking tips from the given QR code is really helpful. Hope
```

```
it serves for a long time.Not to forget about the beautiful bottle
green and golden look of
it.,,https://m.media-amazon.com/images/I/81lT2gsd9sL._SY88.jpg,No
detailed user manual.. no idea about the cooking time. How to use is
not describe.,It works well and plastic quality is poor but it can
wothstand the temperature for sure. Its not a toy to look for high
quality plastic. And for the half the price that the other models, we
can ignore that.Functionality wise its perfect.Just buy it,Don't go
for costly products as it is available in a reasonable price and it
has so many great features. I'm happy with it,Pigeon never dissapointd
with their quality. The best way to have healthy, crispy food.,The
outcome of cooked is not up to the mark.  The recipe book was not
attached so,  it is difficult to know how to cook different dishes.  I
tried some but outcome was bad.  Definitely you will have to
compromise taste if you use this appliance.  I'm very much worried
about the current consumption.  Think before you buy these air
fryers.\",\n          \"Not a perfect fit for long usage,One problem
you may face if you use it continuously for a long time may be ear
ache can be start..,Although it's an HP product there's nothing to
write home about this headphone. It does what it is promised and
there's value for money as you can trust the brand but don't expect
anything 'extra'. No volume control or on/off button, comfortable to
wear but not designed for comfort, clear sound, and a mic but no
advanced features for either. Yet it's good for the price!,Product
would not meet my expectation and sound quality is poor.,Sound quality
is good , cancels the background noise.,Not a bad deal,Build quality
and sound quality was good,Base and noise cancelling is also good \\
u263a\\ufe0f\\ud83d\\ude0a,I bought it in october month but now is not
wprking properly\",\n          \"ABOUT
AMAZONBASICS:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxAmazon Basics was launched
in 2009 & is Amazon\\u2019s own inhouse brand for fast moving small
electronic consumer goods.Here amazon uses its massive collection of
sales data to launch products that are in huge demand & already exist
in the market- but at lower prices. Simply put up a similar replica
for something successful but at much affordable prices.If something
isn\\u2019t an immediate hit, Amazon pulls it and moves on.Amazon
otherwise is like an online marketplace where it provides a portal for
various sellers to sell their product BUT with amazonbasics - Amazon
is selling its own product at its own marketplace .Here it derives the
benefit of eliminating any intermediate distributors or retailers &
hence amazonbasics branded products are available for a lower price
attracting bulk customers online.As an additional benefit,
Amazonbasics products are delivered free to prime members & are
covered under amazon warranty for all and hence any claim or
replacement procedures are highly streamlined & immediately taken care
of.ABOUT OUR PRODUCT ( REQUIREMENT vs
ACTUAL):xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxI
recently purchased a new Qualcomm 3 Qbix car charger & was looking for
a cable with USB A to Micro B connector. I listed my priorities under
```

various heads to come up with a conclusion and let us compare the actual product based upon my initial requirements:1.) DATA EXCHANGE:**********Since it was to be primarily used for fast mobile devices charging in car, Data exchange capability was not much of my concern. Preference though would surely have been a USB 3 but it didn\\u2019t bother me if I could only get a USB 2.0 too.ACTUAL PRODUCT: I did try to copy a movie file just for the sake of testing data exchange and I found it to be pretty well. I did not capture any speed data but then we all know speed of data transfer also varies with the type of data being transferred. The more variety of data being transferred simultaneously the lower will be the speed.2.) LENGTH OF THE CABLE :**********Again since I could not afford to have a lengthy loop of cable bunched around my gear knob, I preferred to keep it short & simple hence my only lookout was upto around a meter or below.ACTUAL PRODUCT: The cable came nicely packed in a paper packet and was precisely 0.9 meters or approximately 3 feet long. The length was sufficient for me to plug-in any of the mobile devices to my car charger at the drivers or the side passenger\\u2019s seat.3.) TANGLE FREE/ FLEXIBILITY/ STRENGTH STANDARDS:**********I am not particularly a fan of those stubborn braided wires which are so hard that they retain the shape in which they are bent. I wanted something that was thick yet flexible enough to acquire a circular shape when bunched.ACTUAL PRODUCT: The cable received looked exactly as shown over the site with good flexibility , reasonable thickness & a sturdy intermediate cable. The whole construction of the cable due it\\u2019s cable size & flexibility is almost tangle-free.The associated cable was not exactly thick but can\\u2019t be termed as thin or delicate too. It\\u2019s not the thickest I\\u2019ve seen but then thickest doesn\\u2019t always means most durable. Given my application it\\u2019s more than just suitable.The overall built & quality of the cable & insulation looks promising enough to last few years. Even if used for other than car charging it looks durable enough to last long.I had further shortlisted mansaa & an amkette cable for the same purpose but they were too long for my requirement.4.) COMPACT MOULDED CONNECTORS:**********Had an inclination towards moulded connectors to avoid any issues where the connectors break open exposing the terminal PCBs.ACTUAL PRODUCT : There are no complaints regarding the connectors of the actual cable. The connectors are perfectly moulded without any joints or risks of splitting open. The connector casings are further quite compact at terminals to fit in comfortably at scarce spaces.The connector ports are sturdy enough both at USB A & micro B ports. The micro B port pins lock securely onto the charging mobile devices which is quite good.No signs of loose construction.Being Gold-plated is more of a misleading & fancy term(in this case) as most of the metal ports designed today already have a corrosion resistance & nobody is going to use them in

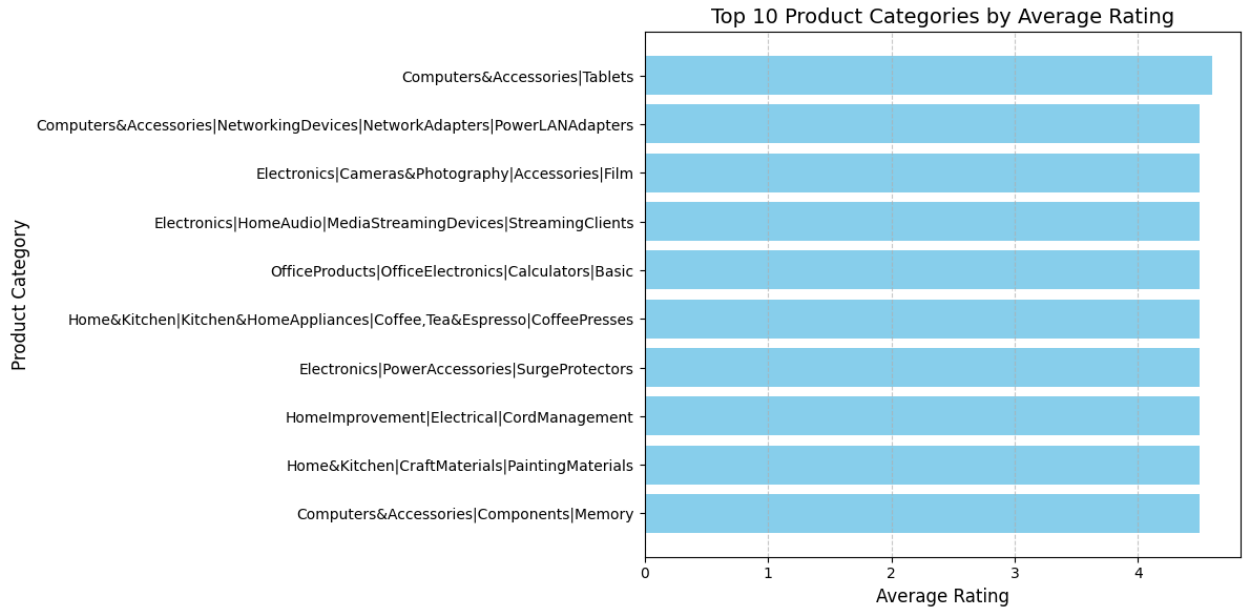
saline sea water anyways.5.) AVAILABILITY OF TIES/VELCRO STRAPS :**********As per my intended use in a car where compactness was of paramount importance, I expected an included cable tie or a Velcro strap would be a nice add-on to properly adjust & arrange the cable as per requirement.ACTUAL PRODUCT: This I miss the most in the provided actual cable, there is no provision of an included strap or cable tie through which I could adjust my required cable length easily.6.) DECENT CURRENT HANDLING CAPABILITY:**********On the newest QC 3.0 certified chargers the current transmission can go up to 3.4 amps in certain cases hence the cable needed to have a decent current handling ability.ACTUAL PRODUCT : The actual cable has a nameplate rating of handling up to 2.1 amps against demands at new QC 3.0 chargers that could go up to 3.4 amps. Here, let me clear out that the latest QC chargers vary voltage to current ratios to achieve desired fast charging & thus it is not like a continuous flow of 3.4 amps is there, it keeps on reducing hence cable ratings designed for a continuous current handling of anything above 2.0 amps would sufficiently work with QC 3.0 chargers.Have tried it safely multiple times charging my Samsung S7 from 10 % without the cable getting hot.7.) WARRANTY:**********Comes with a 1 year warranty as expected. Not the best in the industry but reasonable.OTHER OBSERVATIONS:**********Apart from the above listed features, the other details of the actual product received worth mentioning are:8.) The cable is manufactured in China & imported by amazon warehouse dealers under the brand name & philosophy of amazonbasics.9.) The cable has a manufacturing date of Oct 2017 & it was imported to India in Dec 2017.10.) The cable has a MRP tag of 495 bucks however I purchased it online for 269 bucks.being a prime member shipping was free.11.) A customer care toll free number & email is also printed over the label for registration of any consumer grievances.FINAL VERDICT:xxxxxxxxxxxxxxxxxxIt has been few days since I have received & been using this product. So far everything from construction to performance seems to be convincing enough to recommend it and for a price of around 260 I suppose, its worth it.Will surely update if any malfunction is observed.,Worth for buy!,The quality that amazon basics at times give at cheap prices is beyond imagination. simply superb, goverment shouldnt hinder amazon products , amazon products rather gives competition to local qualityless products which consumers are forced to buy beacause they have no quality competition. Make in india is good, but if the make in india products are simple cheap copies of branded products without any investment in R and D, without R and D make in india would never be successfull and ousting companies like amazon will only lead to loss for consumers , govt should infact encourage such competition.,Amazon basics provides one of the best cables available for charging your phone or connecting devices.As an past customer of many cables from Amazon this cable

```
doesn't also disappoint,Supports fast charging for all my Samsung phones.I use Samsung a9pro 2016, Samsung s8plus 2017, which this cable is compatible with .Very sturdy, thick and very long. 6 ftVery affordable pricing. Thanks AmazonI also use a USB c cable for my Samsung s20fe.,Super,Product charging is ok.. however it's mere 1 foot in length..  the vendor could have mentioned  correct product description.. there is no need to mislead.. too early to say performance as I have received it today.,Good,I have bought many cheap chinese micro usb cable in Rs 50 and Rs 100 of ubon and of many other chinese local companies,and none of them worked properly.Finall i decided to go for this.And it is charging as well transferring data, without any issue and i am very happy with my purchaseMy advice : Dont buy, cheap chinese local cables of.You will have to throw them in dustbean after some time.Better buy this one.\"\n        ],\n \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"img_link\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 1412,\n        \"samples\": [\n          \"https://m.media-amazon.com/images/W/WEBP_402378-T2/images/I/51esjcOy79L._SY300_SX300_QL70_FMwebp_.jpg\",\n \"https://m.media-amazon.com/images/I/41nRBNNDnNL._SX300_SY300_QL70_FMwebp_.jpg\",\n          \"https://m.media-amazon.com/images/I/31-hWNXDxiL._SX300_SY300_QL70_ML2_.jpg\"\n        ],\n \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"product_link\",\n \"properties\": {\n        \"dtype\": \"string\",\n \"num_unique_values\": 1465,\n        \"samples\": [\n \"https://www.amazon.in/Snapdragon-Resolution-Refresh-27-81Cm-Display/dp/B09XXZXQC1/ref=sr_1_437?qid=1672903017&s=computers&sr=1-437\",\n          \"https://www.amazon.in/Skadioo-Accessories-Receiver-Compatible-dongle/dp/B09LHXNZLR/ref=sr_1_195?qid=1672909134&s=electronics&sr=1-195\",\n \"https://www.amazon.in/LOHAYA-Assistant-Compatible-Xstream-Function/dp/B09LV13JFB/ref=sr_1_408?qid=1672909144&s=electronics&sr=1-408\"\n ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n }\n    }\n  ]\n}","type":"dataframe","variable_name":"df2"}
```


#Q1.What is the average rating for each product category?

```
import pandas as pd
import matplotlib.pyplot as plt

df2 = pd.read_csv('amazon.csv')

# Convert the 'rating' column to numeric, handling errors
df2['rating'] = pd.to_numeric(df2['rating'], errors='coerce')

# Calculate the average rating for each product category
avg_rating_by_category = df2.groupby('category')
```

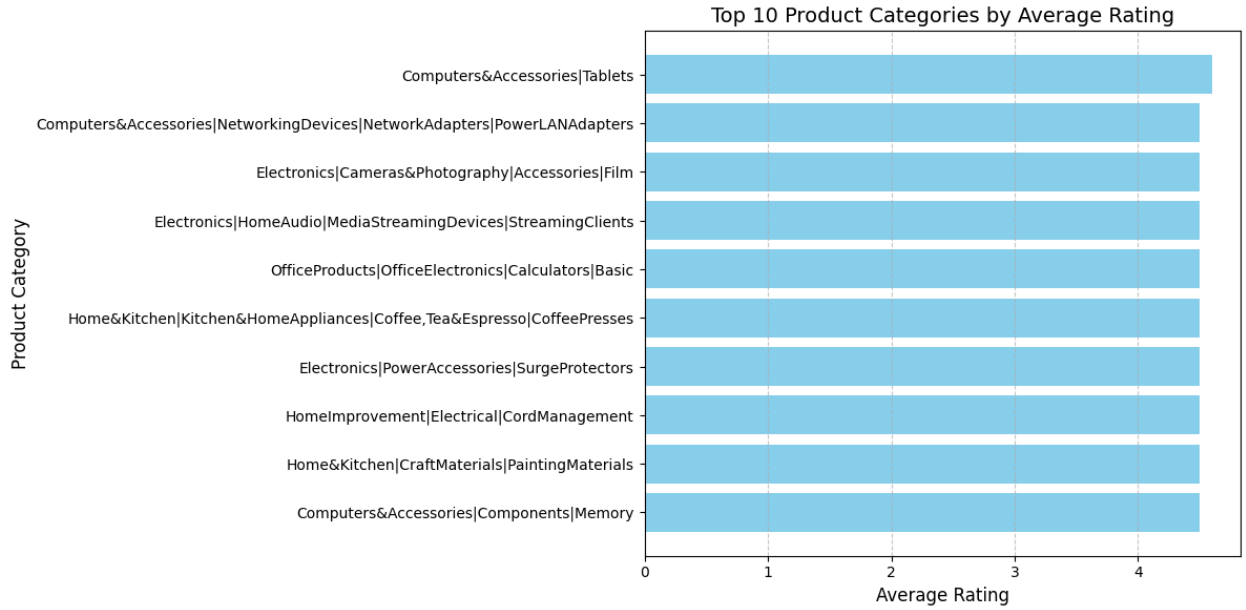
```
['rating'].mean().reset_index()

# Sort the categories by average rating in descending order
avg_rating_by_category_sorted =
avg_rating_by_category.sort_values(by=['rating'], ascending=False)

# Select top 10 categories by average rating for visualization
top_categories = avg_rating_by_category_sorted.head(10)

# Plot the data
plt.figure(figsize=(12, 6))
plt.barh(top_categories['category'], top_categories['rating'],
color='skyblue')

# Customize the plot
plt.xlabel('Average Rating', fontsize=12)
plt.ylabel('Product Category', fontsize=12)
plt.title('Top 10 Product Categories by Average Rating', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```



#Q2.What are the top rating_count products by category?

```
import matplotlib.pyplot as plt

# Select top 10 categories by average rating for visualization
top_categories = avg_rating_by_category_sorted.head(10)
```

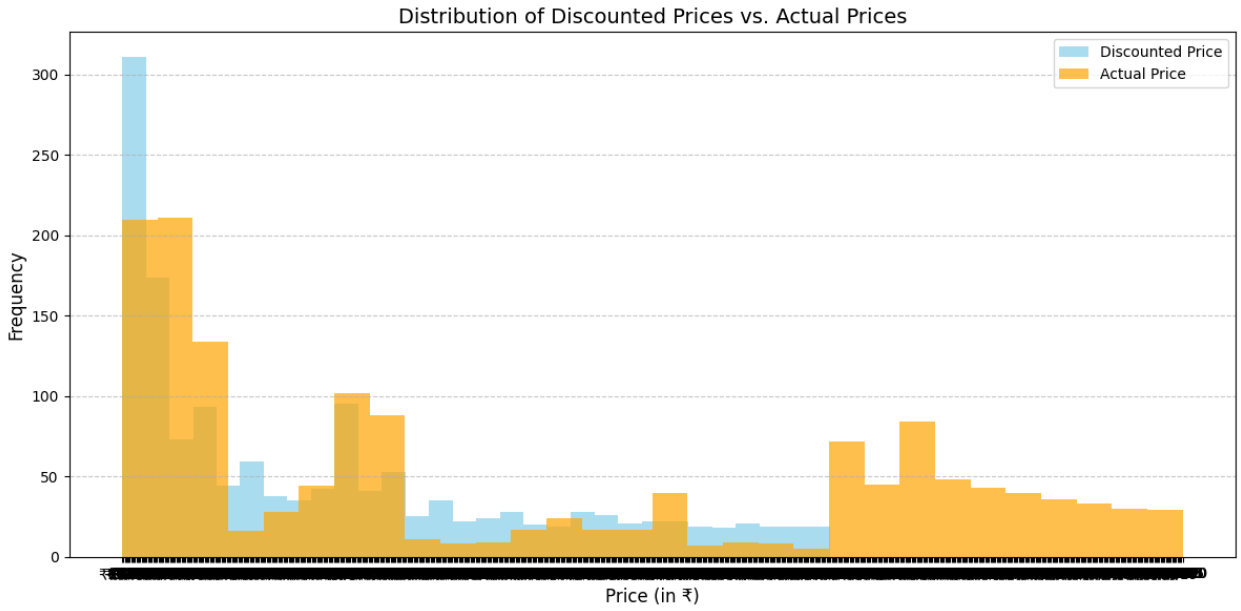
```
# Plot the data
plt.figure(figsize=(12, 6))
plt.barh(top_categories['category'], top_categories['rating'],
color='skyblue')

# Customize the plot
plt.xlabel('Average Rating', fontsize=12)
plt.ylabel('Product Category', fontsize=12)
plt.title('Top 10 Product Categories by Average Rating', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```



#Q3. What is the distribution of discounted prices vs. actual prices?

```
# Plot the distributions of discounted and actual prices
plt.figure(figsize=(12, 6))

# Assuming 'df2' from previous cells contains the Amazon data:
plt.hist(df2['discounted_price'], bins=30, alpha=0.7,
label='Discounted Price', color='skyblue')
plt.hist(df2['actual_price'], bins=30, alpha=0.7, label='Actual
Price', color='orange')

# Customize the plot
plt.xlabel('Price (in ₹)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
```

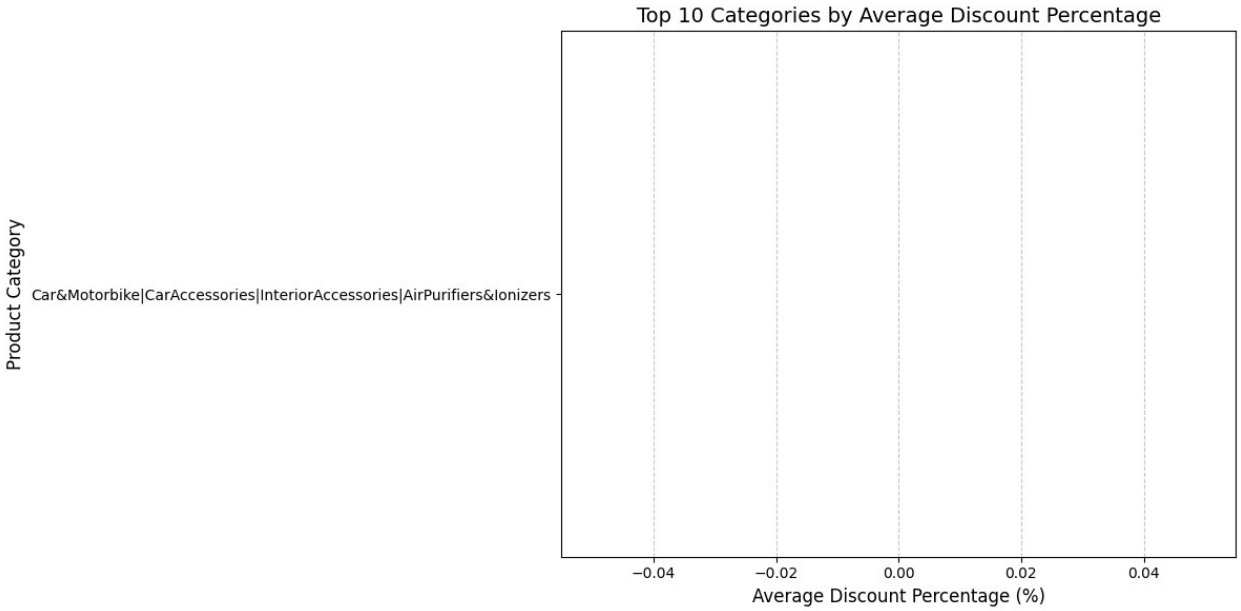
```
plt.title('Distribution of Discounted Prices vs. Actual Prices',
fontsize=14)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```



#Q4.How does the average discount percentage vary across categories?

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df2 contains the Amazon data

# Calculate discount percentage
df2['discount_percentage'] = ((df2['actual_price'] -
df2['discounted_price']) / df2['actual_price']) * 100

# Calculate average discount percentage by category
avg_discount_by_category = df2.groupby('category')
['discount_percentage'].mean().reset_index()

# Sort categories by average discount percentage in descending order
avg_discount_by_category_sorted =
avg_discount_by_category.sort_values(by=['discount_percentage'],
ascending=False)

# Select the top 10 categories with the highest average discount
percentage
```

```
top_discount_categories = avg_discount_by_category_sorted.head(10)

# Plot the data
plt.figure(figsize=(12, 6))
plt.barh(top_discount_categories['category'],
top_discount_categories['discount_percentage'], color='skyblue')

# Customize the plot
plt.xlabel('Average Discount Percentage (%)', fontsize=12)
plt.ylabel('Product Category', fontsize=12)
plt.title('Top 10 Categories by Average Discount Percentage',
fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```

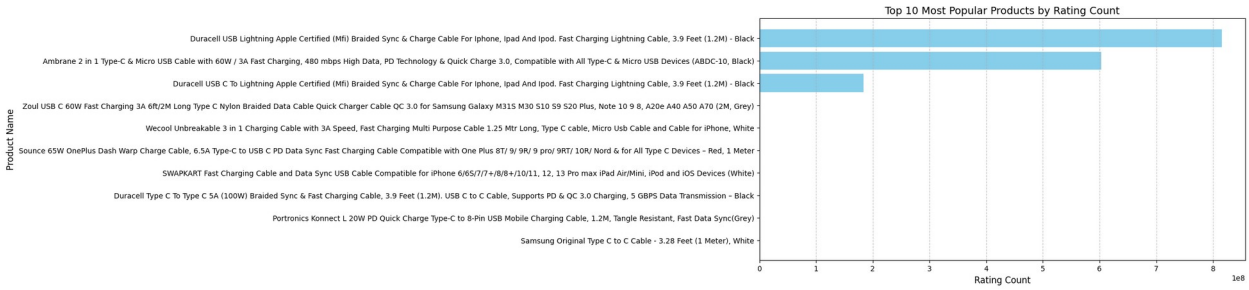


#Q5.What are the most popular product names?

```
import matplotlib.pyplot as plt
import pandas as pd  # Import pandas if not already imported

# Assuming 'df2' from previous cells contains the Amazon data:
# Calculate the rating count for each product
most_popular_products = df2.groupby('product_name')
['rating_count'].sum().reset_index()

# Convert 'rating_count' to numeric before sorting
most_popular_products['rating_count'] =
```

```
pd.to_numeric(most_popular_products['rating_count'], errors='coerce')

# Sort products by rating count in descending order
most_popular_products =
most_popular_products.sort_values(by=['rating_count'],
ascending=False)

# Select the top 10 most popular products
top_popular_products = most_popular_products.head(10)

# Plot the data
plt.figure(figsize=(12, 6))
plt.barh(top_popular_products['product_name'],
top_popular_products['rating_count'], color='skyblue')

# Customize the plot
plt.xlabel('Rating Count', fontsize=12)
plt.ylabel('Product Name', fontsize=12)
plt.title('Top 10 Most Popular Products by Rating Count', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```

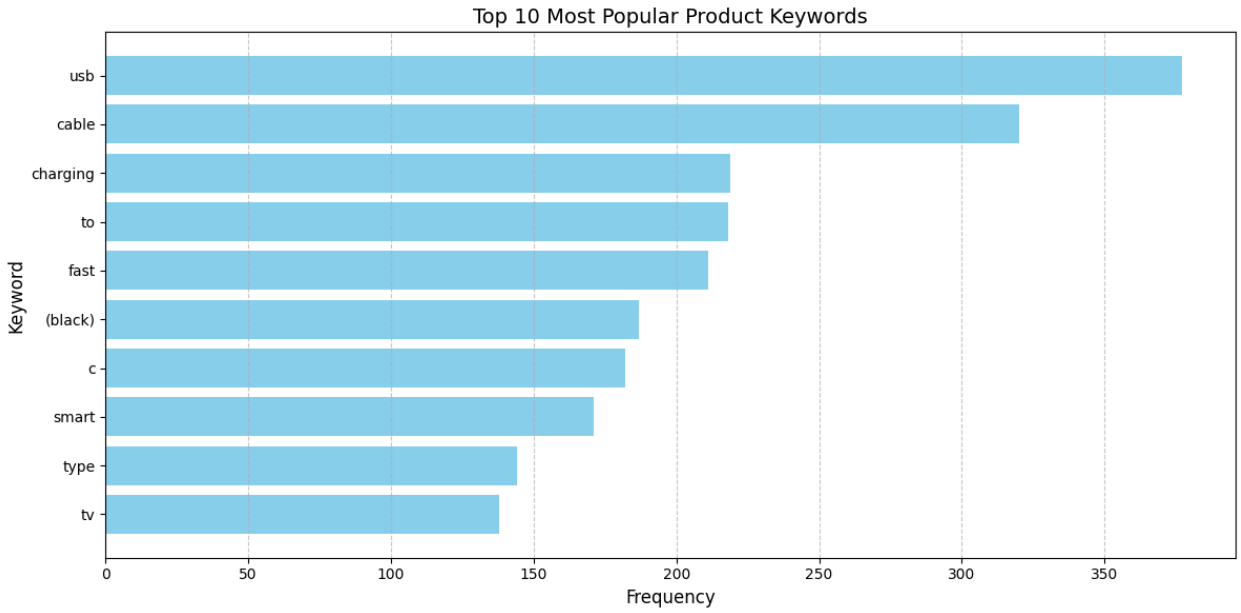
```
<ipython-input-27-79e71cceddbc>:27: UserWarning: Tight layout not
applied. The left and right margins cannot be made large enough to
accommodate all axes decorations.
  plt.tight_layout()
```



#Q6.What are the most popular product keywords?

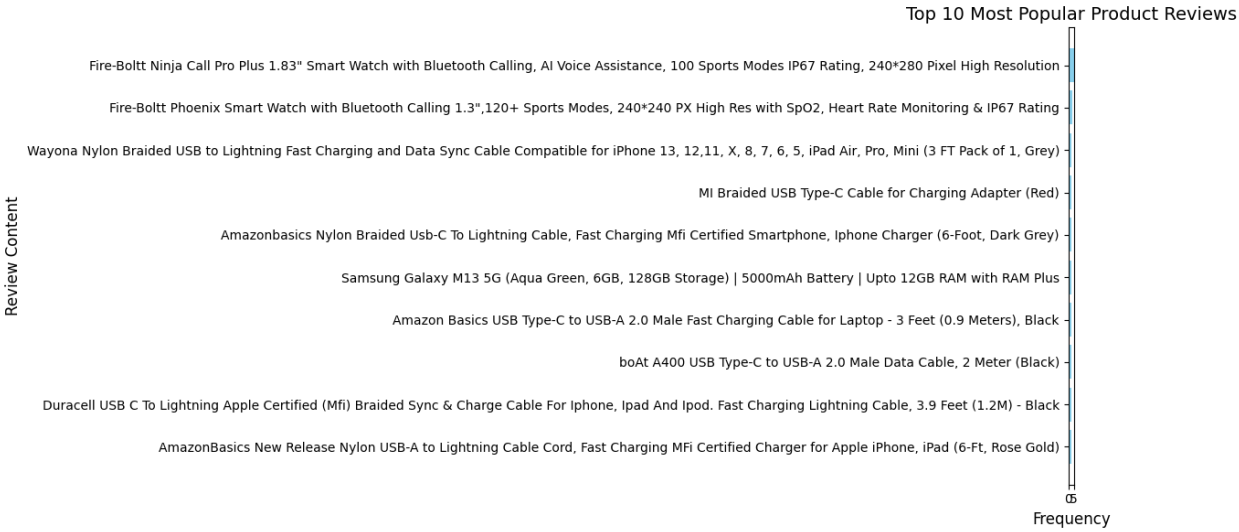
```
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df2 contains the Amazon data

# 1. Extract keywords from product names
df2['keywords'] = df2['product_name'].str.lower().str.split()  # Split
product names into keywords
```

```
all_keywords = [keyword for sublist in df2['keywords'] for keyword in
sublist]  # Combine all keywords

# 2. Calculate keyword frequency
keyword_df = pd.DataFrame(all_keywords, columns=['Keyword'])
keyword_df =
keyword_df.groupby('Keyword').size().reset_index(name='Frequency')

# 3. Sort keywords by frequency in descending order
keyword_df_sorted = keyword_df.sort_values(by=['Frequency'],
ascending=False)

# 4. Exclude generic words and select the top 10 most popular keywords
common_words = ['with', 'for', 'and', '|', '&', '-']
filtered_keywords =
keyword_df_sorted[~keyword_df_sorted['Keyword'].isin(common_words)].he
ad(10)

# 5. Plot the data
plt.figure(figsize=(12, 6))
plt.barh(filtered_keywords['Keyword'], filtered_keywords['Frequency'],
color='skyblue')

# Customize the plot
plt.xlabel('Frequency', fontsize=12)
plt.ylabel('Keyword', fontsize=12)
plt.title('Top 10 Most Popular Product Keywords', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```



#Q7.What are the most popular product reviews?

```
import matplotlib.pyplot as plt
import pandas as pd
review_frequency = df2['product_name'].value_counts().reset_index()  #
Assuming 'product_name' is the reviews column
review_frequency.columns = ['Review', 'Frequency']

# 2. Create the most_popular_reviews_new DataFrame
most_popular_reviews_new = review_frequency  # Or rename if desired

# 3. Select the top 10 most popular reviews
top_reviews = most_popular_reviews_new.head(10)

# 4. Plot the data
plt.figure(figsize=(12, 6))
plt.barh(top_reviews['Review'], top_reviews['Frequency'],
color='skyblue')

# Customize the plot
plt.xlabel('Frequency', fontsize=12)
plt.ylabel('Review Content', fontsize=12)
plt.title('Top 10 Most Popular Product Reviews', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.tight_layout()

# Show the plot
plt.show()
```

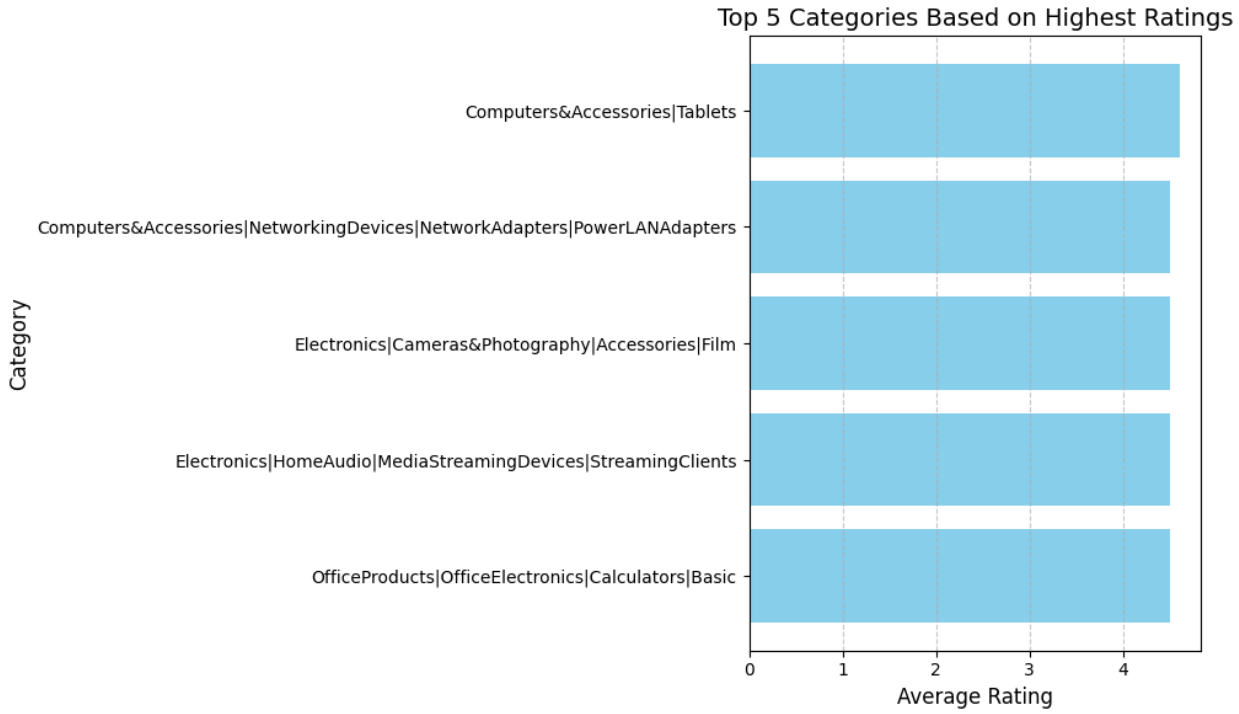


#Q8.What is the correlation between discounted_price and rating?

```
import matplotlib.pyplot as plt

# Assuming 'df2' from your previous cells contains the data
cleaned_data = df2  # If you don't need to clean data, use df2
directly

# Scatter plot of discounted_price vs. rating
plt.figure(figsize=(10, 6))
plt.scatter(cleaned_data['discounted_price'], cleaned_data['rating'],
alpha=0.5, color='skyblue')

# Customize the plot
plt.xlabel('Discounted Price (₹)', fontsize=12)
plt.ylabel('Rating', fontsize=12)
plt.title('Scatter Plot: Discounted Price vs. Rating', fontsize=14)
plt.grid(alpha=0.5)

# Show the plot
plt.tight_layout()
plt.show()
```

#Q9. What are the Top 5 categories based on the highest ratings?

```
import matplotlib.pyplot as plt

# Assuming 'avg_rating_by_category_sorted' from previous cells
contains the data
# Assign the top 5 categories to 'top_categories_by_rating'
top_categories_by_rating = avg_rating_by_category_sorted.head(5)  #
Select the top 5

# Plot the top 5 categories by average ratings
plt.figure(figsize=(10, 6))
plt.barh(top_categories_by_rating['category'],
top_categories_by_rating['rating'], color='skyblue')

# Customize the plot
plt.xlabel('Average Rating', fontsize=12)
plt.ylabel('Category', fontsize=12)
plt.title('Top 5 Categories Based on Highest Ratings', fontsize=14)
plt.gca().invert_yaxis()  # Invert y-axis for better readability
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

# Show the plot
plt.show()
```



#Q10. Identify any potential areas for improvement or optimization based on the data analysis.?

```
# 1. Analyze categories with the lowest average ratings
# Assuming 'df2' contains the Amazon data and 'rating' is numeric:
average_rating_by_category = df2.groupby('category')
['rating'].mean().reset_index()
low_rated_categories =
average_rating_by_category.sort_values(by='rating',
ascending=True).head(5)

# 2. Identify categories with high discounts but low ratings
cleaned_data['discount_percentage'] = ((cleaned_data['actual_price'] -
cleaned_data['discounted_price']) / cleaned_data['actual_price']) *
100
high_discount_low_rating = cleaned_data.groupby('category').agg(
    avg_discount=('discount_percentage', 'mean'),
    avg_rating=('rating', 'mean')
).reset_index()
high_discount_low_rating_filtered =
high_discount_low_rating[high_discount_low_rating['avg_rating'] <
3].sort_values(by='avg_discount', ascending=False).head(5)

# 3. Analyze categories with low sales (rating count)
low_sales_categories = cleaned_data.groupby('category')
['rating_count'].sum().reset_index()
low_sales_categories_sorted =
low_sales_categories.sort_values(by='rating_count',
ascending=True).head(5)
```

```
# Display results
low_rated_categories, high_discount_low_rating_filtered,
low_sales_categories_sorted
```

```
(                                                 category  rating
 146  Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea...     3.3
 14   Computers&Accessories|Accessories&Peripherals|...     3.4
 2    Computers&Accessories|Accessories&Peripherals|...     3.5
 88   Electronics|HomeTheater,TV&Video|Accessories|3...     3.5
 56   Computers&Accessories|Printers,Inks&Accessorie...     3.6,
 Empty DataFrame
 Columns: [category, avg_discount, avg_rating]
 Index: [],
                                                 category
rating_count
 184  Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cle...
1,017170297
 148  Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea...
1,065
 153  Home&Kitchen|Kitchen&HomeAppliances|SewingMach...
1,06713,2512,4492,283
 0    Car&Motorbike|CarAccessories|InteriorAccessori...
1,118
 109  Electronics|Mobiles&Accessories|MobileAccessor...
1,193)
```

EDA - 4

```
df3=pd.read_csv('spotify.csv')
df3
```


```
{"summary":"{\n  \"name\": \"df3\",\n  \"rows\": 440,\n  \"fields\":
[\n    {\n      \"column\": \"Artist\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 115,\n
\"samples\": [\n          \"Playboi Carti\",\n          \"Nicki
Minaj\",\n          \"NEIKED\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Track Name\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 412,\n
\"samples\": [\n          \"Shoota (feat. Lil Uzi Vert)\",\n
\"PUFFIN ON ZOOTIEZ\",\n          \"ROCKSTAR (feat. Roddy Ricch)\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Popularity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9,\n        \"min\": 29,\n        \"max\": 97,\n
\"num_unique_values\": 51,\n        \"samples\": [\n          35,\n
54,\n          52\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Duration (ms)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 53576,\n        \"min\": 81666,\n
\"max\": 501648,\n        \"num_unique_values\": 410,\n
\"samples\": [\n          203894,\n          225905,\n
213593\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Track ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 413,\n        \"samples\":
[\n          \"50ceCGZ3oD3U5caQV5bP6f\",\n
\"4LwOrnuxJwR7C5Sw4liY4Z\",\n          \"1c7MITQmNJTrvfbDSzWT6x\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df3"}
```


#Q1. Read the dataframe, check null value if present then do the needful, check duplicate row , if present then dothe needful?

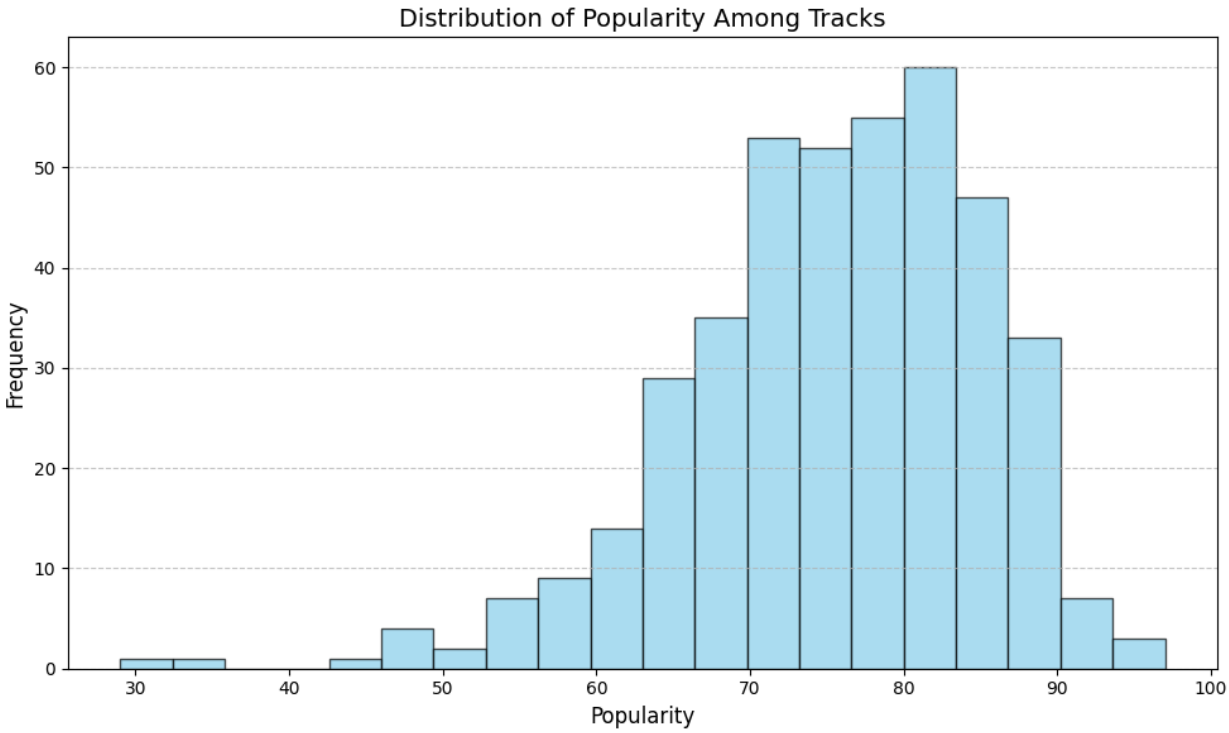
```
import pandas as pd
file_path_spotify = '/mnt/data/spotify.csv'
spotify_data = pd.read_csv('spotify.csv')

# Step 1: Check for null values
null_values = spotify_data.isnull().sum()

# Step 2: Drop duplicate rows
duplicate_rows = spotify_data[spotify_data.duplicated()]
spotify_data_cleaned = spotify_data.drop_duplicates()

# Step 3: Display results
print("Null Values in Each Column:")
print(null_values)

print("\nNumber of Duplicate Rows Removed:", len(duplicate_rows))
print("\nCleaned Dataset Info:")
spotify_data_cleaned.info()
```

```
Null Values in Each Column:
Artist          0
Track Name      0
Popularity      0
Duration (ms)   0
Track ID        0
dtype: int64

Number of Duplicate Rows Removed: 27

Cleaned Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 413 entries, 0 to 438
Data columns (total 5 columns):
```

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Artist          413 non-null    object
 1   Track Name      413 non-null    object
 2   Popularity      413 non-null    int64
 3   Duration (ms)   413 non-null    int64
 4   Track ID        413 non-null    object
dtypes: int64(2), object(3)
memory usage: 19.4+ KB
```

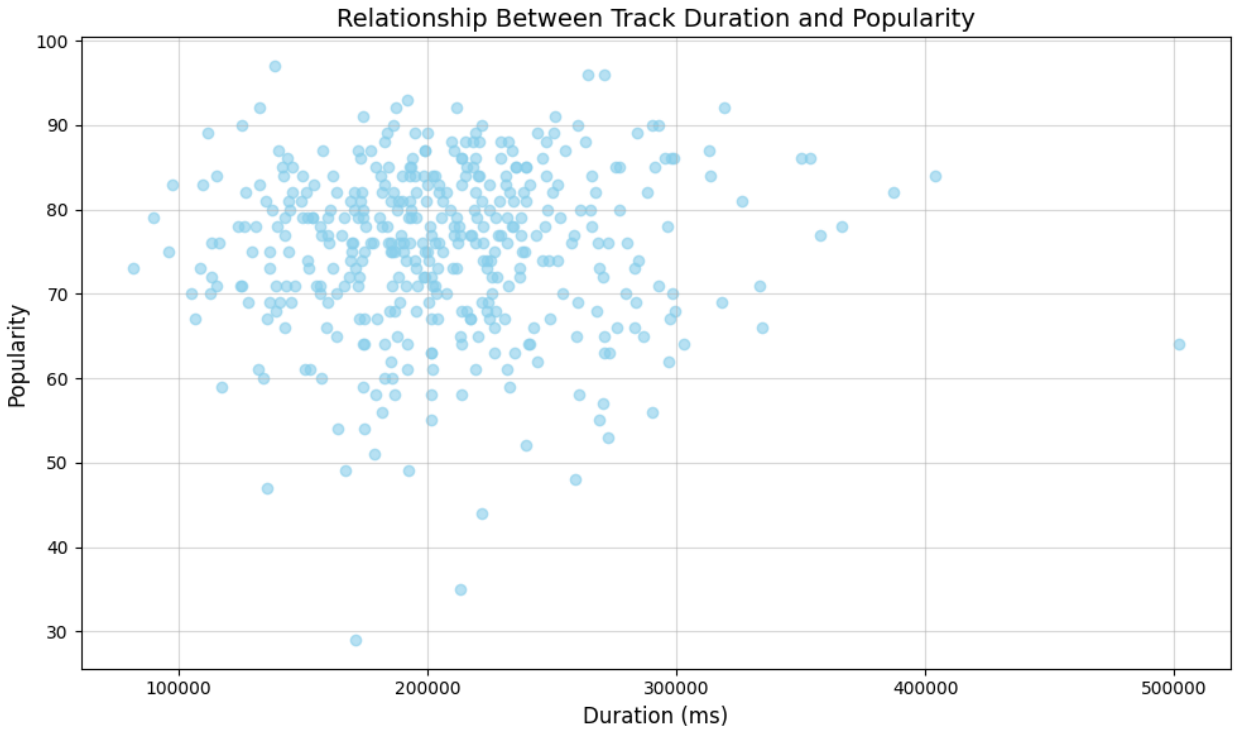
#Q2.What is the distribution of popularity among the tracks in the dataset? Visualize it using a histogram?

```
import matplotlib.pyplot as plt

# Plot a histogram of the 'Popularity' column
plt.figure(figsize=(10, 6))
plt.hist(spotify_data_cleaned['Popularity'], bins=20, color='skyblue',
edgecolor='black', alpha=0.7)

# Customize the plot
plt.title('Distribution of Popularity Among Tracks', fontsize=14)
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()
plt.show()
```



#Q3.Is there any relationship between the popularity and the duration of tracks? Explore this using a scatter plot?

```
import matplotlib.pyplot as plt

# Scatter plot of Duration vs. Popularity
plt.figure(figsize=(10, 6))
plt.scatter(spotify_data_cleaned['Duration (ms)'],
spotify_data_cleaned['Popularity'], alpha=0.6, color='skyblue')

# Customize the plot
plt.title('Relationship Between Track Duration and Popularity',
fontsize=14)
plt.xlabel('Duration (ms)', fontsize=12)
plt.ylabel('Popularity', fontsize=12)
plt.grid(alpha=0.5)

# Show the plot
plt.tight_layout()
plt.show()
```

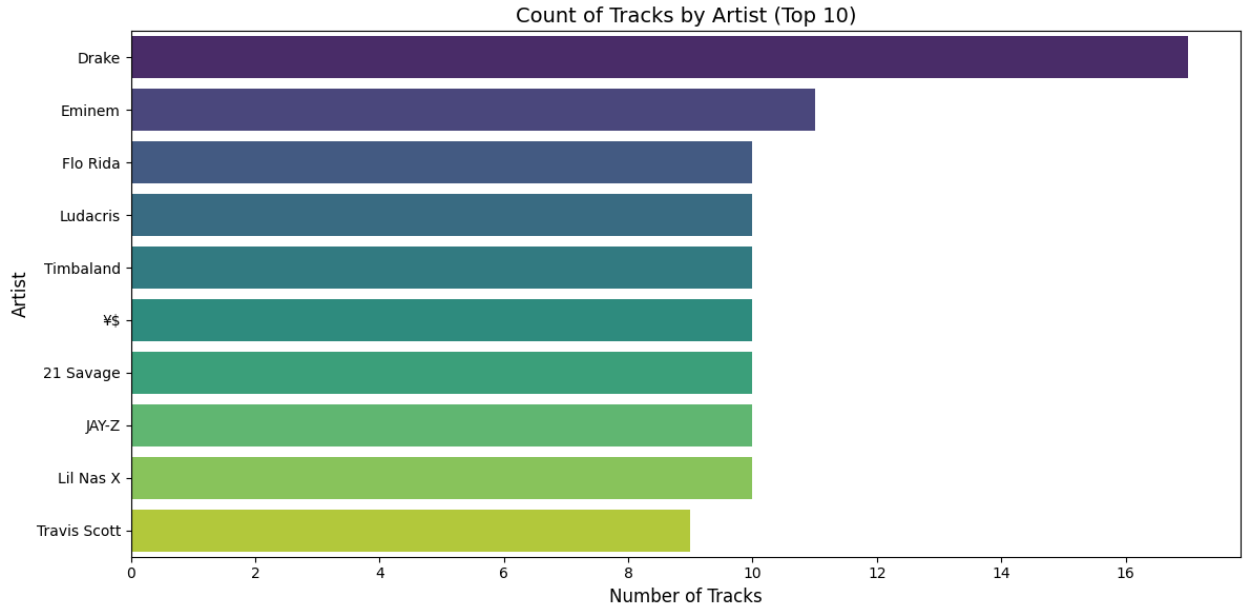


#Q4.Which artist has the highest number of tracks in the dataset? Display the count of tracks for each artist using a countplot?

```
import seaborn as sns
import matplotlib.pyplot as plt

# Find the artist with the highest number of tracks
artist_track_counts = spotify_data_cleaned['Artist'].value_counts()

# Display the artist with the most tracks
most_tracks_artist = artist_track_counts.idxmax()
most_tracks_count = artist_track_counts.max()

# Plot the count of tracks for each artist (top 10 for clarity)
plt.figure(figsize=(12, 6))
sns.countplot(
    y=spotify_data_cleaned['Artist'],
    order=artist_track_counts.head(10).index,
    palette="viridis"
)

# Customize the plot
plt.title('Count of Tracks by Artist (Top 10)', fontsize=14)
plt.xlabel('Number of Tracks', fontsize=12)
plt.ylabel('Artist', fontsize=12)
plt.tight_layout()
```

```
# Show the plot
plt.show()

# Print the artist with the most tracks
f"Artist with the most tracks: {most_tracks_artist}
({most_tracks_count} tracks)"
```

```
<ipython-input-45-ad3238f72a58>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(
```




```
{"type":"string"}
```

#Q5.What are the top 5 least popular tracks in the dataset? Provide the artist name and track name for each?

```
# Sort the dataset by the 'Popularity' column in ascending order to
find the least popular tracks
least_popular_tracks =
spotify_data_cleaned.sort_values(by='Popularity',
ascending=True).head(5)

# Select the artist name and track name for the least popular tracks
least_popular_tracks_info = least_popular_tracks[['Artist', 'Track
Name', 'Popularity']]
```

```
least_popular_tracks_info
```


```
{"summary":"{\n  \"name\": \"least_popular_tracks_info\",\n  \"rows\":
5,\n  \"fields\": [\n    {\n      \"column\": \"Artist\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"Justin
Bieber\",\n          \"Wyclef Jean\",\n          \"French Montana\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Track Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Intentions\",\n          \"911 (feat. Mary J. Blige)\",\n
\"Splash Brothers\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Popularity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8,\n        \"min\": 29,\n
\"max\": 48,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          35,\n          48,\n          44\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"least_popular_tracks_info"}
```


#Q6.Among the top 5 most popular artists, which artist has the highest popularity on average? Calculate and display the average popularity for each artist?

```
# Calculate the average popularity for each artist
artist_avg_popularity = spotify_data_cleaned.groupby('Artist')
['Popularity'].mean().reset_index()

# Sort by average popularity in descending order to find the top 5
most popular artists
top_5_artists = artist_avg_popularity.sort_values(by='Popularity',
ascending=False).head(5)

# Identify the artist with the highest average popularity
most_popular_artist = top_5_artists.iloc[0]

# Display the results
top_5_artists, most_popular_artist
```

```
(          Artist  Popularity
 113         cassö   92.000000
 104        Trueno   89.000000
 24   David Guetta   87.000000
 103  Travis Scott   86.555556
 114            ¥$   85.100000,
 Artist        cassö
 Popularity     92.0
 Name: 113, dtype: object)
```

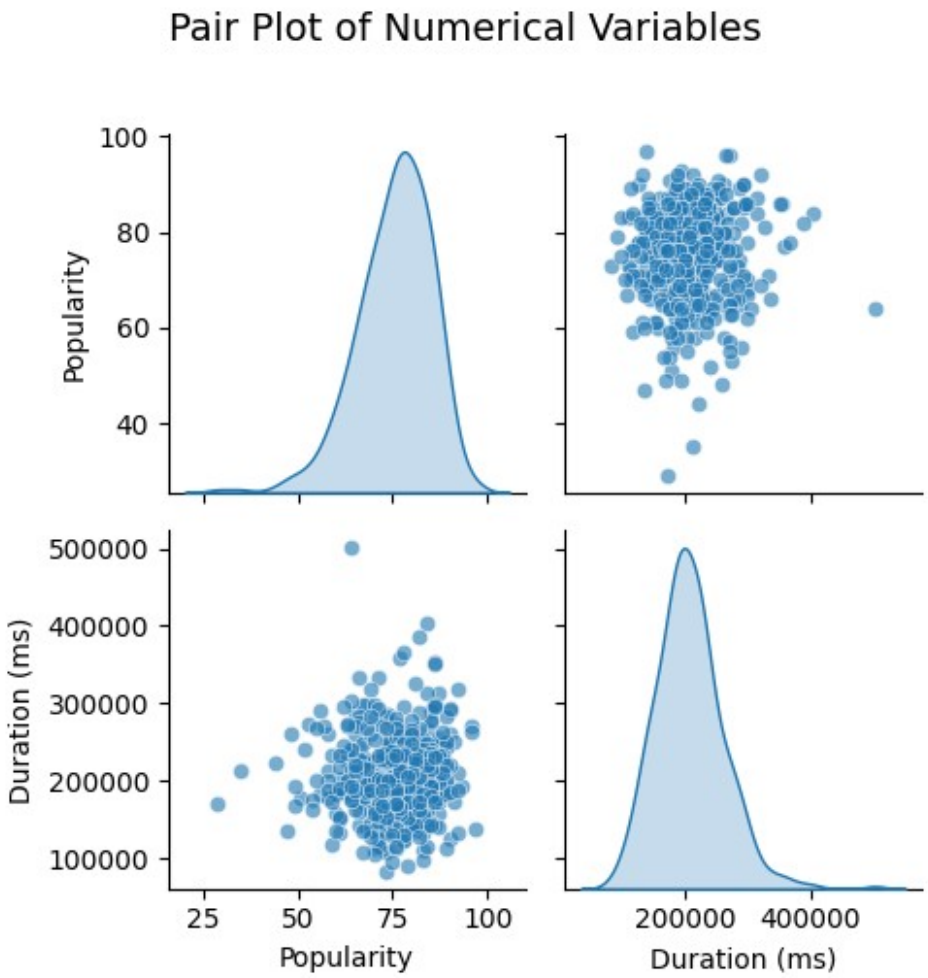
#Q7.For the top 5 most popular artists, what are their most popular tracks? List the track name for each artist?

```
# Calculate the average popularity for each artist
artist_avg_popularity = spotify_data_cleaned.groupby('Artist')
['Popularity'].mean().reset_index()

# Sort by average popularity in descending order to find the top 5
most popular artists
top_5_artists = artist_avg_popularity.sort_values(by='Popularity',
ascending=False).head(5)['Artist']

# Find the most popular track for each of the top 5 artists
most_popular_tracks = []
for artist in top_5_artists:
    artist_tracks =
spotify_data_cleaned[spotify_data_cleaned['Artist'] == artist]
    most_popular_track = artist_tracks.sort_values(by='Popularity',
ascending=False).iloc[0]
    most_popular_tracks.append({
        'Artist': artist,
        'Track Name': most_popular_track['Track Name'],
        'Popularity': most_popular_track['Popularity']
    })

# Convert to a DataFrame for display
most_popular_tracks_df = pd.DataFrame(most_popular_tracks)

most_popular_tracks_df
```


```
{"summary":"{\n  \"name\": \"most_popular_tracks_df\",\n  \"rows\":
5,\n  \"fields\": [\n    {\n      \"column\": \"Artist\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Trueno\",\n          \"\\u00a5$\",\n          \"David Guetta\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Track Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Mamichula - con Nicki Nicole\",\n          \"CARNIVAL\",\n
\"Baby Don't Hurt Me\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Popularity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3,\n        \"min\": 87,\n
\"max\": 96,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          89,\n          96,\n          87\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"most_popular_tracks_df"}
```

#Q8.Visualize relationships between multiple numerical variables simultaneously using a pair plot?

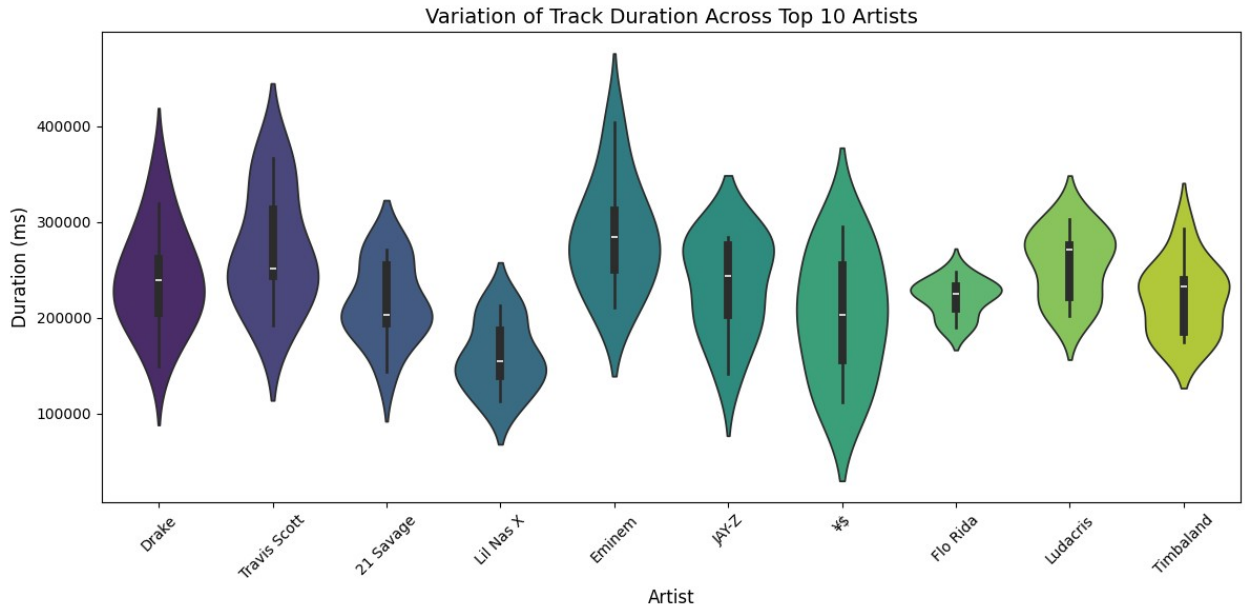
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select numerical columns for the pair plot
numerical_data = spotify_data_cleaned[['Popularity', 'Duration (ms)']]

# Create a pair plot
sns.pairplot(numerical_data, diag_kind='kde', kind='scatter',
plot_kws={'alpha': 0.6})

# Customize and show the plot
plt.suptitle('Pair Plot of Numerical Variables', y=1.02, fontsize=14)
plt.tight_layout()
plt.show()
```



#Q9. Does the duration of tracks vary significantly across different artists? Explore this visually using a box plot or violin plot?

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select the top 10 artists with the most tracks for better
visualization
top_10_artists =
spotify_data_cleaned['Artist'].value_counts().head(10).index
filtered_data =
spotify_data_cleaned[spotify_data_cleaned['Artist'].isin(top_10_artist
s)]

# Create a violin plot for track duration by artist
plt.figure(figsize=(12, 6))
sns.violinplot(x='Artist', y='Duration (ms)', data=filtered_data,
palette='viridis')

# Customize the plot
plt.title('Variation of Track Duration Across Top 10 Artists',
fontsize=14)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Duration (ms)', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()

# Show the plot
plt.show()
```

```
<ipython-input-50-806001ac9843>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.violinplot(x='Artist', y='Duration (ms)', data=filtered_data,
palette='viridis')
```



#Q10. How does the distribution of track popularity vary for different artists? Visualize this using a swarm plot or a violin plot.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select the top 10 artists with the most tracks for better
visualization
top_10_artists =
spotify_data_cleaned['Artist'].value_counts().head(10).index
filtered_data =
spotify_data_cleaned[spotify_data_cleaned['Artist'].isin(top_10_artist
s)]

# Create a violin plot for track popularity by artist
plt.figure(figsize=(12, 6))
sns.violinplot(x='Artist', y='Popularity', data=filtered_data,
palette='muted')

# Customize the plot
plt.title('Distribution of Track Popularity Across Top 10 Artists',
fontsize=14)
plt.xlabel('Artist', fontsize=12)
plt.ylabel('Popularity', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()

# Show the plot
plt.show()
```

```
<ipython-input-51-55ee558394e4>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(x='Artist', y='Popularity', data=filtered_data, palette='muted')
```