world-cup-2023-prediction

December 11, 2023

0.1 ICC World Cup 2023

0.2 Import libraries

[1]:
```
# Importing necessary libraries for data analysis and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# Read the data from "World_cup_2023.csv" into the 'World_cup' DataFrame
World_cup = pd.read_csv("World_cup_2023.csv")

# Read the data from "results.csv" into the 'results' DataFrame
results = pd.read_csv(r"E:\Sem 3rd\ICC-2023-Worldcup\Datasets\results.csv")
```

[3]:
```
# Display the first few rows of the World_cup DataFrame.
World_cup
```

[3]:

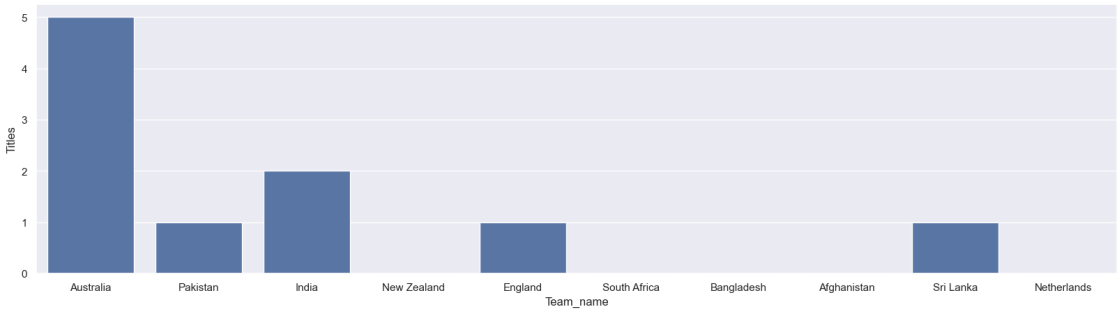
	Team_name	Team_ranking	Titles	Win_percentage_ODI	WC_matches	WC_match_won	Win_percent_WC	WC_match_loss	Loss_percent_WC	Tied
0	Australia	1	5	60.73	94	69	73.40	23	24.46	1
1	Pakistan	2	1	52.78	79	45	56.96	32	40.50	0
2	India	3	2	52.38	84	53	63.09	29	34.52	1
3	New Zealand	4	0	45.89	89	54	60.67	33	37.07	1
4	England	5	1	50.32	83	48	57.83	32	38.55	2
5	South Africa	6	0	61.00	64	38	59.37	23	35.93	2
6	Bangladesh	7	0	36.65	40					
7	Afghanistan	8	0	49.65	15					
8	Sri Lanka	9	1	45.74	80					
9	Netherlands	10	0	34.21	20					

```
6              14            35.00              25            62.50      0
7               1             6.66              14            93.33      0
8              38            47.50              39            48.75      1
9               2            10.00              18            90.00      0
```

```
   No_result World_cup_winner  Recent_points  Rating
0          1              Yes           2714     118
1          2              Yes           2316     116
2          1              Yes           3807     115
3          1               No           2806     104
4          1              Yes           2426     101
5          1               No           1910     101
6          1               No           2451      98
7          0               No           1361      91
8          2              Yes           2794      87
9          0               No           1044      37
```

0.3 Q(1) No.of titles won by each teams

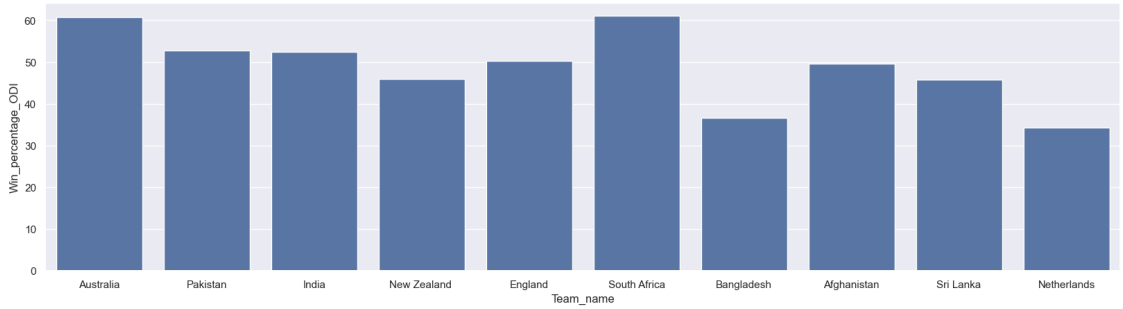
[4]:
```
# Set the figure size using sns.set
sns.set(rc={'figure.figsize':(20, 5)})

# Create a bar plot using sns.barplot to visualize team titles
sns.barplot(x='Team_name', y='Titles', data=World_cup)

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



0.4 Q(2) Win percentage in ODI by each team

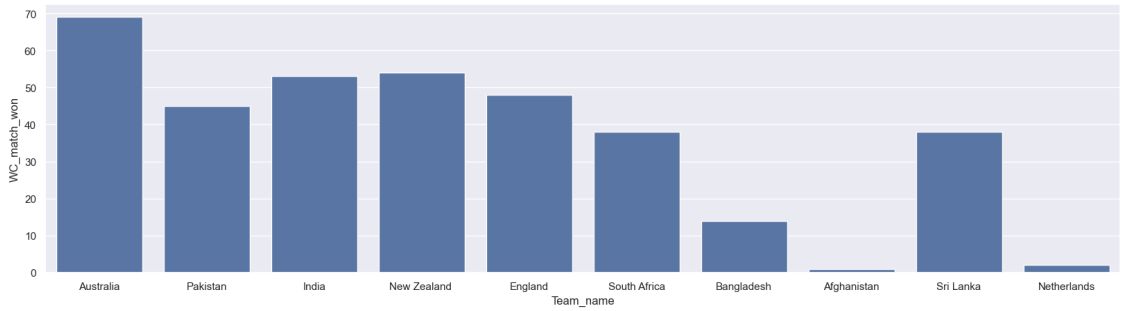
[5]:
```
# Set the figure size for the bar plot using Seaborn
sns.set(rc={'figure.figsize':(20, 5)})

# Create a bar plot using Seaborn
```

```
sns.barplot(x='Team_name', y='Win_percentage_ODI', data=World_cup)

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



0.5 Q(3) No.of matches won in world cup by each team

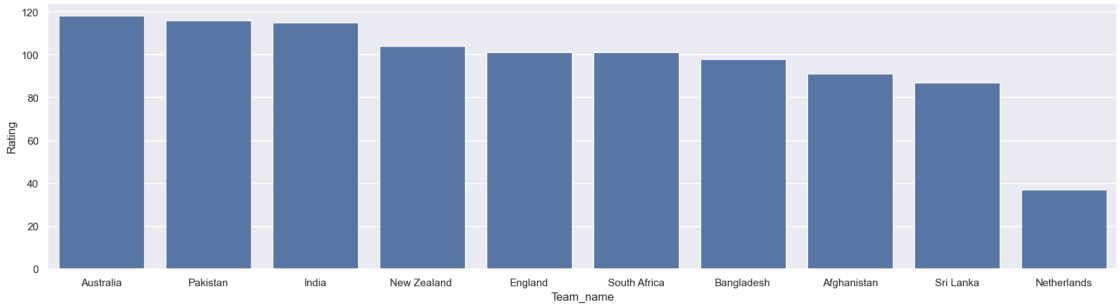
[6]:
```
# Set the figure size for the bar plot using Seaborn
sns.set(rc={'figure.figsize':(20, 5)})

# Create a bar plot using Seaborn
sns.barplot(x='Team_name', y='WC_match_won', data=World_cup)

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



0.6 Q(4) Recent ICC ODI rating

[7]:
```
# Set the figure size for the bar plot using Seaborn
sns.set(rc={'figure.figsize':(20, 5)})

# Create a bar plot using Seaborn to display recent ratings of teams
sns.barplot(x='Team_name', y='Rating', data=World_cup)
```

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



[8]:
```
# Displaying the first few rows of the results DataFrame.
#results.head()
```

[9]:
```
# Removing rows with 'Match abandoned' and 'No result' from the 'results'␣
 ↪DataFrame."
results.drop(results[(results['Winner'] == 'Match abandoned' )].index,␣
 ↪inplace=True)
results.drop(results[(results['Winner'] == 'No result' )].index, inplace=True)
```

0.7 Q(5) Number of wins for India against each team in the ODI World Cup

[9]:
```
# Number of wins against each team in the ODI world cup
# Out of the 84 ODI matches played by India in the ODI world cup, number of␣
 ↪matches won against the following teams
team_win_counts_wc_ind = {
    'Australia': 4,
    'New Zealand': 3,
    'South Africa ': 2,
    'Pakistan': 7,
    'Sri Lanka': 5,
    'Bangladesh': 3,
    'England': 3,
    'Netherlands': 2,
    'Afghanistan': 2
}

# Total matches played is calculated
total_matches_wc_ind = sum(team_win_counts_wc_ind.values())

# India's win percentages against each team is calculated
```

```
win_percentages_wc_ind = {team: (wins / total_matches_wc_ind) * 100 for team,␣
 ↪wins in team_win_counts_wc_ind.items()}

# Pie chart
plt.figure(figsize=(5, 5))
plt.pie(win_percentages_wc_ind.values(), labels=win_percentages_wc_ind.keys(),␣
 ↪autopct='%1.1f%%', startangle=140)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Title for the pie chart
plt.title('Win Percentage of India in the ODI world cup')

# Display the pie chart
plt.show()
```

[ ]:

ic-patient-medical-report-analysis

December 11, 2023

1 Project Title - Cardiac Patients Medical Report Analysis

1.0.1 Import requried libraries

[ ]:
```
# Import required libraries

import matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

[ ]:
```
# Create dataframe

heart_disease = pd.read_csv('/content/clean_data_heart.csv')
```

[ ]:
```
heart_disease.head(10)
```

[ ]:
```
# overall statistics

heart_disease.describe().round(2)
```

1.0.2 Q.1: What age group has show higher sign of heart attack?

- The `displot` shows that people of age 55-60 years old shows high distribution along the dataset.
- Much of patients are of 50-70 age group.
- Here, we just created a new column 'age_group', which gives a better picture of the patients age data.

[ ]:
```
# Create a dataframe with the total number of patients of all age groups

df1 = heart_disease.groupby(['age_group']).target.count().to_frame(name=None)

df1.reset_index(inplace=True) # convert the index column to column

df1.rename(columns={'target':'total_patients'}, inplace=True) # rename the␣
 ↪column name

df1
```

- 123 patients were in their 50s, followed by people in their 60s and 40s.

[ ]:
```
# Create another dataframe based on the target

df2 =heart_disease.groupby(['age_group', 'target']).target.count()#.
 ↪to_frame(name=None)
df2
```

[ ]:
```
df2 = df2.to_frame(name=None) # convert the series to dataframe
df2.rename(columns={'target':'targetwise_total'}, inplace=True) # Rename the␣
 ↪target column
```

[ ]:
```
df2
```

- From the above series we can say that
    1. Most of admitted patients are in there 50s.
    2. Heart issues starts from 30s, and after 40 it becomes severe.
    3. The `df2` data frame shows how many patients have heart risk.
    4. Most people at 40s and 50s are at high risk of heart attack.

1.0.3 Q.2 Which gender were at high risk of heart attack?

[ ]:
```
# Divide dataset to higher risk and lower risk dataset

high_risk_patients = heart_disease.loc[(heart_disease.target==1)]
low_risk_patients = heart_disease.loc[heart_disease.target==0]
```

[ ]:
```
# Patients data who are at higher risk of heart attack
high_risk_patients.head()
```

- We just divided the whole dataset to two parts as patients with higher risk and patients with lower risk.
- For that we used `.loc` magic method.
- So that we can find the distribution of chances of heart attack

[ ]:
```
# Histogram of Age distribution in higher risk patients of both genders

# sns.set(rc={'figure.figsize':(10,8)})
g = sns.FacetGrid(high_risk_patients, col='sex', margin_titles=True, height=6)
g.map(sns.histplot, 'age', color='#c04000')
g.add_legend()
g.fig.suptitle('Fig-4:Age distribution in higher risk patients')
plt.show()
```

In the above hist plot sex =0 : Female, and sex=1 : Male.

The above hist plot concludes that: - Females of age group 50 are at high risk of heart disease - In case of male, around 40 and near 60, heart disease is common.

[ ]:
```
# Replacing gender values with name in higher_risk dataframe

high_risk_patients.sex = high_risk_patients.sex.replace([0,1], ['Female',␣
 ↪'Male'])
high_risk_patients.rename(columns={'target':'high_risk'}, inplace=True)
high_risk_patients.head()
```

[ ]:
```
# Number of patients (genderwise) at high risk

d3_1 = high_risk_patients.groupby(['sex']).high_risk.count().to_frame(name=None)

d3_1.reset_index(inplace=True)
d3_1
```

[ ]:
```
# Group the higher risk patients based on gender

df4 = high_risk_patients.groupby(['sex','age_group']).high_risk.count().
 ↪to_frame(name=None)
df4 = df4.transpose()

print('Table showing number of male patients and female patient with high risk␣
 ↪of heart attack')
df4
```

- Most of female patients with higher risk of heart attack were in their 50s followed by 60s.
- 70+ year old females were more at risk than male of same group.

[ ]:
```
# Total number of patients

d3 = heart_disease.groupby(['sex']).sex.count().to_frame(name=None)
d3.rename(columns={'sex':'total'}, inplace=True) # change column name
d3.reset_index(inplace=True) # convert index_col to column
d3.sex.replace([0,1],['Female','Male'], inplace=True) # changing values
d3
```

- There were total 95 female patients and 205 male patients.

[ ]:
```
# Find percentage

percent_female = d3_1.high_risk.iloc[0]/d3.total.iloc[0] * 100 # female␣
 ↪percentage

percent_male = d3_1.high_risk.iloc[1]/d3.total.iloc[1]*100 # male percentage


print(f'There were {round(percent_female,2)}% female were at high risk of heart␣
 ↪attack.')
print(f'Also {round(percent_male,2)}% male were at high risk.')
```

[ ]:
```
# Bar graph of gender and target corelation

sns.countplot(x='sex', hue='target', data=heart_disease)
plt.xticks([1,0], ['Male', 'Female'])
plt.legend(labels=['No-Hert attack', 'Heart attack'])
plt.title('Gender Distributuon',loc='left')
plt.show();
```

The above **DataFrame and Barchart** shows that

- In comparision to male and female, female shows higher rate of risk of heart attack.
- The barplot here shows the count of male and female.
- There are 95 female with 205 male.
- 74.4% female are at risk of heart attack, i.e. 71 out of 95 female.
- 44.9% of male shows risk of heart attack, which is 92 out of 295 male.

1.0.4 Q3. Which chest pain results in heart attack?

[ ]:
```
# Types of chest pain

cp_type = heart_disease.groupby(['cp']).cp.count()
cp_type
```

[ ]:
```
# plot Types of chest pain

sns.countplot(heart_disease.cp)
plt.xticks([0,1,2,3,], ['Typical angina', 'Atypical angina', 'Non-angina',␣
 ↪'Asymptotic'])
plt.title('Common types of Chest Pain', loc='left')
plt.xlabel('Chest Pain')
plt.show()
```

- There are 4 types of chest pain: > 1. Typical angina: or Angina pectoris, a condition where heart does not get enough oxygen or blood due to coronary artery blockage. > 2.

Atypical angina: Chest pain but no symptoms of angina, happens when heart doesn't get enough oxygeneted blood. > 3. Non-anginal: or Non-cardiac chest pain (NCCP), feels like chest pain but rather happens due to acidity i.e. Gastroesophageal Reflux Disease (GERD). > 4. Asymptotic chest pain: or Silent Myocardial Infarction (SMI), happens due to stress, high cholesterol, high bp, diabetes, and other physical or mental issues. One of a reason of premature death in India. Mostly seen in middle aged people

- Here, 142 patients have typical angina, 85 have non-angina, 50 have atypical angina and rest 23 have asymptotic chest pain.

[ ]:
```
# Divide dataset based on chest pain types

typical_angina_patients = heart_disease.loc[heart_disease.cp==0]
atypical_angina_patients = heart_disease.loc[heart_disease.cp==1]
non_angina_patients = heart_disease.loc[heart_disease.cp==2]
asymptotic_patients = heart_disease.loc[heart_disease.cp==3]
```

[ ]:
```
typical_angina_patients.head(3)
```

[ ]:
```
typical_cp = typical_angina_patients.groupby(['target']).target.count()
typical_cp
```

[ ]:
```
# percent of typical angina high risk patients

cp0_highrisk_prct = typical_cp[1]/cp_type[0]*100

print(f'Among people with typical angina, only {round(cp0_highrisk_prct)}% are␣
 ↪at high risk of heart attack.')
```

[ ]:
```
# Similarly

atypical_cp = atypical_angina_patients.groupby(['target']).target.count()
non_anginal_cp = non_angina_patients.groupby(['target']).target.count()
asymptotic_cp = asymptotic_patients.groupby(['target']).target.count()


# percentage

cp1_risk = atypical_cp[1]/cp_type[0]*100
cp2_risk = non_anginal_cp[1]/cp_type[0]*100
cp3_risk = asymptotic_cp[1]/cp_type[0]*100

print('NOTE:')
print(f'Among patients with atypical angina {round(cp1_risk,2)}% were at a risk␣
 ↪of heart attack.')
print(f'Where as in case of non-anginal patients the risk is much higher at a␣
 ↪rate of {round(cp2_risk,2)}% of patients, and {round(cp3_risk,2)}% patients␣
 ↪with asymptotic angina are vulnerable')
```

[ ]:
```
# Bargraph to check heart attack risk due to different type chest pains

sns.countplot(x='cp',hue='target', data=heart_disease)
plt.legend(labels=['No-Heart attack', 'Heart attck'])
plt.xticks([0,1,2,3],['Typical angina', 'Atypical angina', 'Non-angina',␣
 ↪'Asymptotic'])
plt.title('Chest pain leading to Heart Attack', loc='left')
plt.xlabel('Chest pain')
plt.show()
```

- 142 patients have typical angina, among which only 27.5% are at risk of heart attack.
- People with atypical angina, 82% are at risk of heart attack.
- Whereas, among non-angina patients 78.8% are vulnerable to heart attack.
- Asymptotic chest pain rarely lead to heart attack which is 7.7%.

Note: - Most of chest pains are stress releated. However, NCCP and atypical are vulnerable conditions to heart attack.

1.0.5 Q4. How fasting blood sugar level is related to heart attack?

[ ]:
```
# Barchart to compare the fbs level

sns.countplot(x='fbs', hue='target', data=heart_disease)
plt.legend(labels=['No-Heart attack', 'Heart attack'])
plt.title('Relation ship between Fasting blood sugar level and Heart disease',␣
 ↪loc='left')
plt.xticks([0,1],['<120 mg/dL', '>120 mg/dL'])
plt.xlabel('Fasting blood sugar level (mg/dL)')
plt.show()
```

[ ]:
```
# catplot to show the relationship b/w fbs and age

sns.catplot(x='fbs',y='age', data=heart_disease)
plt.title('Fig-10: Fasting blood sugar level vs Age\n', loc='left')
plt.xlabel('Fasting blood sugar level (mg/dl)')
plt.xticks([0,1],['<120', '>120'])
plt.show();
```

[ ]:
```
# distribution plot of fbs on both the genders

sns.FacetGrid(heart_disease, hue='sex', aspect=4).map(sns.kdeplot,'fbs',␣
 ↪shade=True)
plt.legend(labels=['Male', 'Female'])
plt.title('Fig-11: Fasting blood sugar level based on gender\n', loc='left')
plt.show();
```

- The above plots shows that fasting blood sugar level is not much affected by sex and age.

- However, fbs level in female patients are higher than male (as per the `kdeplot`).

1.0.6 Q.5. What type of thalassemia leads to heart attack?

[ ]:
```
# Types of thalassemia

thal_types = heart_disease.groupby(['thal']).thal.count()
thal_types
```

[ ]:
```
# Countplot of thal vs target

sns.countplot(x='thal',hue='target', data=heart_disease)
plt.title('Fig-12: Types of Thalassemia vs risk of heart disease\n', loc='left')
plt.legend(["No-Heart attack", 'Heart attack'])
plt.xticks([0,1,2],['Normal', 'Fixed defect', 'Reversible defect'], rotation=70)
plt.xlabel('Thalassemia')
plt.show()
```

- 165 patients had fixed defect thalassemia.
- Whereas 117 had reversible defect, and only 18 had normal type.

[ ]:
```
# Thalassemia patients with high risk of heart attack

highrisk_thal = high_risk_patients.groupby(['thal']).thal.count()
highrisk_thal
```

[ ]:
```
# percent of thalassemia patients with high risk of heart attack

thal2_prcnt = highrisk_thal[2]/thal_types[2]*100
thal2_prcnt
```

[ ]:
```
# Gender wise thalassemia patients

genderwise_thaltypes = heart_disease.groupby(['thal', 'sex']).sex.count()
genderwise_thaltypes
```

[ ]:
```
# Number of patients with high risk of heart attack and thalassemia

genderbased_thal= high_risk_patients.groupby(['thal', 'sex']).sex.count()
genderbased_thal
```

[ ]:
```
# Percentage of male patients with higher risk of heart attack and thalassemia␣
↪fixed defect

thal2_prct_male = genderbased_thal[2][1]/genderwise_thaltypes[2][1]* 100
thal2_prct_male
```

[ ]:
```
# Percent of female patients with fixed defect thalassemia and at high risk

thal2_prct_female = genderbased_thal[2][0]/genderwise_thaltypes[2][0]*100
thal2_prct_female
```

[ ]:
```
# Percentage of female patients with revesible defect thalassemia

thal3_prct_female = genderbased_thal[3][0]/genderwise_thaltypes[3][0]*100
thal3_prct_female
```

[ ]:
```
# catplot of thalasemia and risk of heart attack based on gender

sns.catplot(x='sex', y='target', hue='thal', kind='bar', data=heart_disease,␣
 ↪height=6)
plt.xticks([0,1], ['Female', 'Male'])
plt.title('Fig-13: Heart disease in relationship to Thalassemia\n',loc='left')
plt.show();
```

restaurant-reviews

December 11, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from wordcloud import WordCloud
from collections import Counter
```

[2]:
```
file_path="Restaurant_Reviews.tsv"
df=pd.read_csv(file_path, delimiter = '\t', quoting = 3)
```

[3]:
```
df.head()
```

[3]:

	Review	Liked
0	Wow… Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of…	1
4	The selection on the menu was great and so wer…	1

[4]:
```
# Shape of dataframe
df.shape
```

[4]: (1000, 2)

Q1-> How to use PortStemmer in Preprocessing

0.1 Data Preprocessing

[5]:
```
nltk.download('stopwords')

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[6]:
```
corpus=[]

for i in range(0, 1000):
  review = re.sub('[^a-zA-Z]', ' ', df['Review'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
  review = ' '.join(review)
  corpus.append(review)
```

[7]:
```
corpus[500]
```

[7]:
```
'also tast mom multi grain pumpkin pancak pecan butter amaz fluffi delici'
```

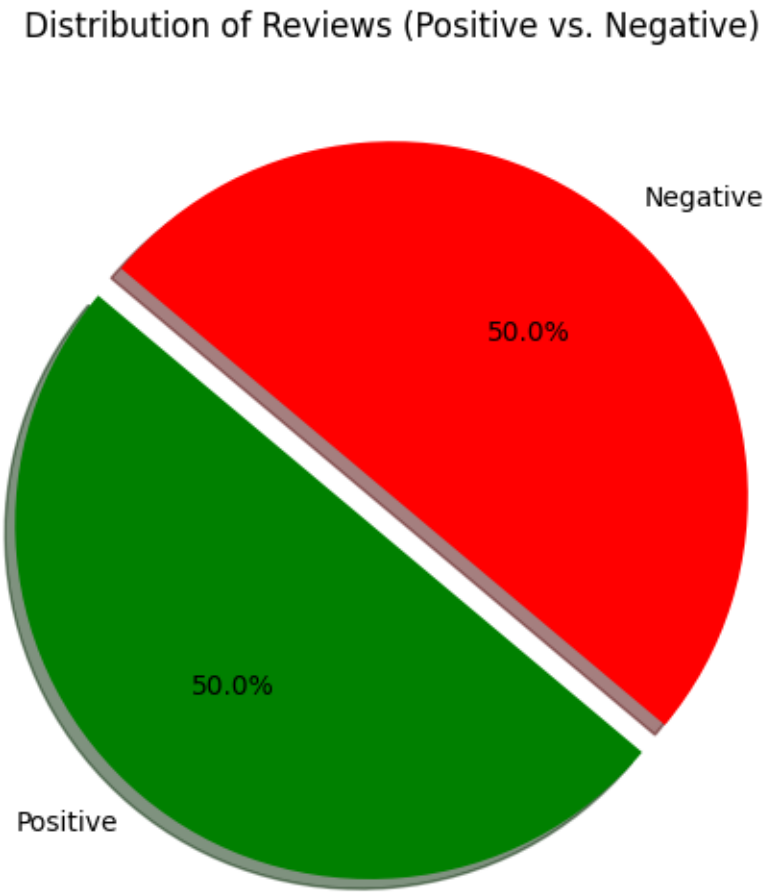
Q2-> Top 3 most commom words in our dataset

[8]:
```
# Combine all the preprocessed text into one string
all_text = ' '.join(corpus)

# Tokenize the combined text into words
words = all_text.split()

# Calculate word frequencies using Counter
word_freq = Counter(words)

# Get the top 50 most common words
top_three_words = word_freq.most_common(3)

# Create a dictionary from the top 50 words
word_dict = dict(top_three_words)

# Generate the word cloud from the dictionary
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate_from_frequencies(word_dict)
```

```
# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Word Cloud of Top 50 Words")
plt.show()
```

Q3-> In What Percentage our Positive and Negative Reviews Divided

[9]:
```
import matplotlib.pyplot as plt

# Count the number of positive and negative reviews
positive_reviews_count = (df['Liked'] == 1).sum()
negative_reviews_count = (df['Liked'] == 0).sum()

# Create a pie chart
labels = 'Positive', 'Negative'
sizes = [positive_reviews_count, negative_reviews_count]
colors = ['green', 'red']
explode = (0.1, 0)  # Explode the 'Positive' slice

plt.figure(figsize=(6, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.
↪1f%%', shadow=True, startangle=140)
plt.title("Distribution of Reviews (Positive vs. Negative)")
plt.show()
```



0.2 Data transformation

Q4->Used TF-IDF vectorization with max_features

[10]:
```
# TF-IDF vectorization
tfidf = TfidfVectorizer(max_features=1420)
X = tfidf.fit_transform(corpus).toarray()
y = df.iloc[:, -1].values
```

Q5-> Used random_state with train test split

0.3 Dividing dataset into training and test set

[11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,␣
↪random_state = 0)
```

health-care-data-analysis

December 11, 2023

1 Data Science Lab by Danish Kamal

1.0.1 Importing libraries like `pandas`, `numpy` and `matplotlib`.

Imported Pandas as `pd` for data manipulation and Matplotlib as `plt` for data visualization.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.0.2 Data Loading and Inspection

`read_csv` is inbuilt function in python for read dataset in the form of csv.

[2]:
```
file_path = 'healthcare_dataset.csv'
df = pd.read_csv(file_path)
```

1.0.3 Display the first 5 rows of the DataFrame

[3]:
```
df.head()
```

[3]:

	Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type
0	Tiffany Ramirez	81	Female	O-	Diabetes	2022-11-17	Patrick Parker	Wallace-Hamilton	Medicare	37490.983364	146	Elective
1	Ruben Burns	35	Male	O+	Asthma	2023-06-01	Diane Jackson	Burke, Griffin and Cooper				
2	Chad Byrd	61	Male	B-	Obesity	2019-01-09	Paul Baker	Walton LLC				
3	Antonio Frederick	49	Male	B-	Asthma	2020-05-02	Brian Chandler	Garcia Ltd				
4	Mrs. Brandy Flowers	51	Male	O-	Arthritis	2021-07-09	Dustin Griffin	Jones, Brown and Murray				

```
1   UnitedHealthcare    47304.064845          404      Emergency
2            Medicare    36874.896997          292      Emergency
3            Medicare    23303.322092          480         Urgent
4   UnitedHealthcare    18086.344184          477         Urgent

  Discharge Date   Medication  Test Results
0     2022-12-01      Aspirin  Inconclusive
1     2023-06-15      Lipitor        Normal
2     2019-02-08      Lipitor        Normal
3     2020-05-03   Penicillin      Abnormal
4     2021-08-02  Paracetamol        Normal
```

1.0.4 Display the last 5 rows of the DataFrame

[6]:
```
df.tail()
```

[6]:
```
                     Name  Age  Gender Blood Type Medical Condition  \
9995            James Hood   83    Male         A+            Obesity
9996       Stephanie Evans   47  Female        AB+          Arthritis
9997  Christopher Martinez   54    Male         B-          Arthritis
9998          Amanda Duke   84    Male         A+          Arthritis
9999            Eric King   20    Male         B-          Arthritis

     Date of Admission             Doctor                    Hospital  \
9995        2022-07-29       Samuel Moody   Wood, Martin and Simmons
9996        2022-01-06  Christopher Yates               Nash-Krueger
9997        2022-07-01   Robert Nicholson            Larson and Sons
9998        2020-02-06        Jamie Lewis               Wilson-Lyons
9999        2023-03-22        Tasha Avila  Torres, Young and Stewart

     Insurance Provider  Billing Amount  Room Number Admission Type  \
9995   UnitedHealthcare    39606.840083          110       Elective
9996         Blue Cross     5995.717488          244      Emergency
9997         Blue Cross    49559.202905          312       Elective
9998   UnitedHealthcare    25236.344761          420         Urgent
9999              Aetna    37223.965865          290      Emergency

     Discharge Date  Medication Test Results
9995     2022-08-02   Ibuprofen     Abnormal
9996     2022-01-29   Ibuprofen       Normal
9997     2022-07-15   Ibuprofen       Normal
9998     2020-02-26  Penicillin       Normal
9999     2023-04-15  Penicillin     Abnormal
```

1.0.5 Data Inspection and Exploration

Display the basic information about the DataFrame

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                10000 non-null  object
 1   Age                 10000 non-null  int64
 2   Gender              10000 non-null  object
 3   Blood Type          10000 non-null  object
 4   Medical Condition   10000 non-null  object
 5   Date of Admission   10000 non-null  object
 6   Doctor              10000 non-null  object
 7   Hospital            10000 non-null  object
 8   Insurance Provider  10000 non-null  object
 9   Billing Amount      10000 non-null  float64
 10  Room Number         10000 non-null  int64
 11  Admission Type      10000 non-null  object
 12  Discharge Date      10000 non-null  object
 13  Medication          10000 non-null  object
 14  Test Results        10000 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 1.1+ MB
```

1.0.6 This will show all column name in an array

[8]:
```
df.columns.tolist()
```

[8]:
```
['Name',
 'Age',
 'Gender',
 'Blood Type',
 'Medical Condition',
 'Date of Admission',
 'Doctor',
 'Hospital',
 'Insurance Provider',
 'Billing Amount',
 'Room Number',
 'Admission Type',
 'Discharge Date',
 'Medication',
 'Test Results']
```

1.0.7 Shape give info that how much row and column in dataset.

[9]:
```
df.shape
```

[9]:
```
(10000, 15)
```

1.1 Check for missing values

[13]:
```
missingValue =  df.isnull().sum()
missingValue
```

[13]:
```
Name                  0
Age                   0
Gender                0
Blood Type            0
Medical Condition     0
Date of Admission     0
Doctor                0
Hospital              0
Insurance Provider    0
Billing Amount        0
Room Number           0
Admission Type        0
Discharge Date        0
Medication            0
Test Results          0
dtype: int64
```

[16]:
```
MedUniqueValue = df['Medical Condition'].unique()
InsUniqueValue = df['Insurance Provider'].unique()
MedUniqueValue, InsUniqueValue
```

[16]:
```
(array(['Diabetes', 'Asthma', 'Obesity', 'Arthritis', 'Hypertension',
        'Cancer'], dtype=object),
 array(['Medicare', 'UnitedHealthcare', 'Aetna', 'Cigna', 'Blue Cross'],
       dtype=object))
```

2 Data Analysis

2.0.1 Question1: Calculate average billing amount and total amount.

[21]:
```
average_billing_amount = df['Billing Amount'].mean()
total_billing_amount = df['Billing Amount'].sum()

print(f"Average Billing Amount: {average_billing_amount}")
print(f"Total Billing Amount: {total_billing_amount}")
```

```
Average Billing Amount: 25516.8067777384
Total Billing Amount: 255168067.77738398
```

2.0.2 Question2: How to cacuclate medical condition from dataset and plot barchart.

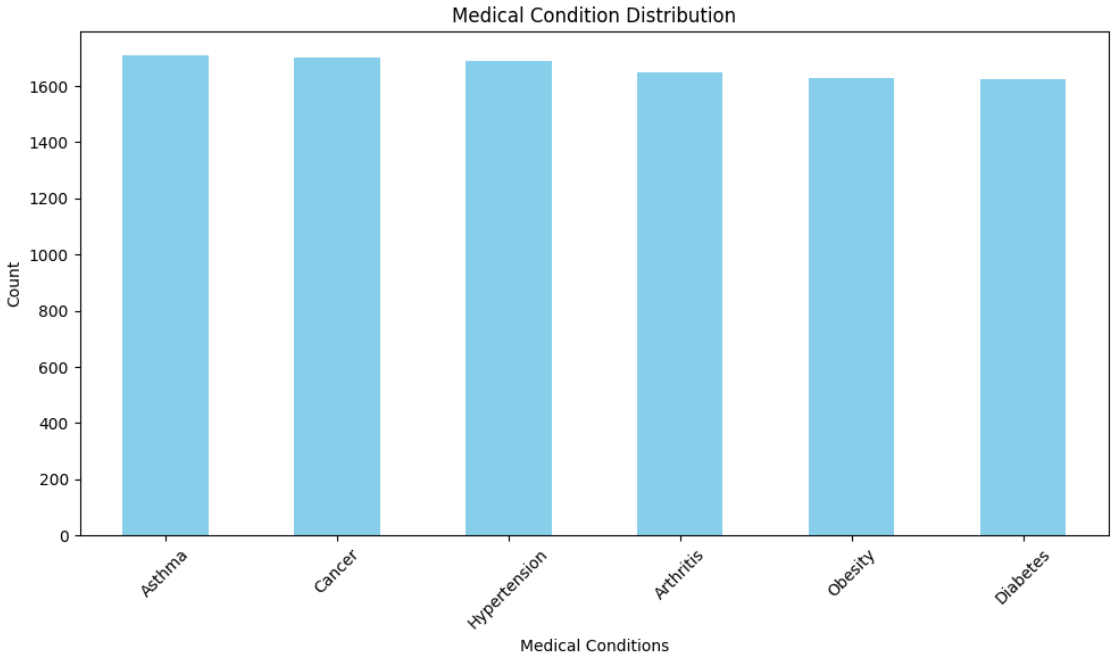
[22]:
```
condition_count = df['Medical Condition'].value_counts()
```

[23]:
```
condition_count
```

[23]:
```
Medical Condition
Asthma          1708
Cancer          1703
Hypertension    1688
Arthritis       1650
Obesity         1628
Diabetes        1623
Name: count, dtype: int64
```

2.0.3 Plotted a pie chart of medical condition.

[24]:
```
plt.figure(figsize=(10, 6))
condition_count.plot(kind='bar', color='skyblue')
plt.title('Medical Condition Distribution')
plt.xlabel('Medical Conditions')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



2.1 Data Exploratory

2.1.1 Question3: How to see all column for data exploratory and find the percentage of male and female patient?

[25]:
```
df.columns
```

[25]:
```
Index(['Name', 'Age', 'Gender', 'Blood Type', 'Medical Condition',
       'Date of Admission', 'Doctor', 'Hospital', 'Insurance Provider',
       'Billing Amount', 'Room Number', 'Admission Type', 'Discharge Date',
       'Medication', 'Test Results'],
      dtype='object')
```

2.1.2 Percentage of male and female patient.

[4]:
```
gender_counts = df['Gender'].value_counts()
gender_counts
```

[4]:
```
Female    5075
Male      4925
Name: Gender, dtype: int64
```

[27]:
```
total_count = gender_counts.sum()
total_count
```

[27]:
```
10000
```

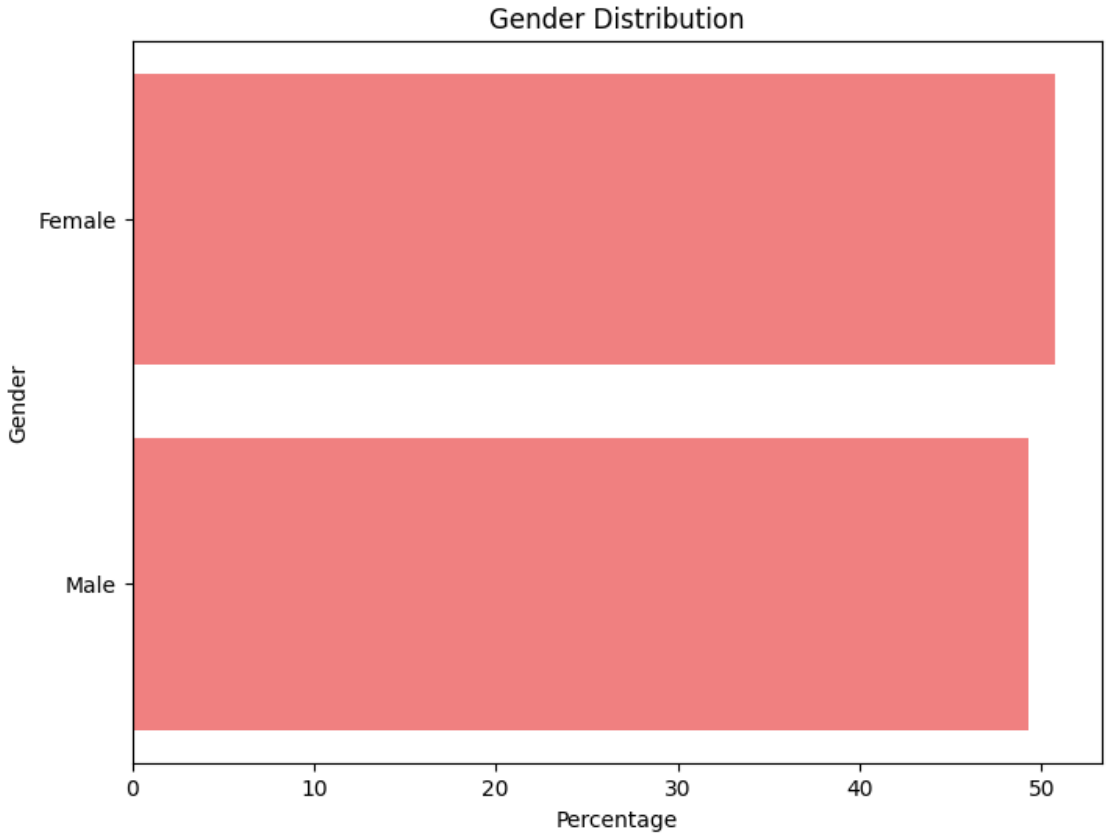
[28]:
```
percentage_male = (gender_counts['Male'] / total_count) * 100
percentage_female = (gender_counts['Female'] / total_count) * 100

print("Pecentage of male:", percentage_male)
print("Pecentage of female:", percentage_female)
```

```
Pecentage of male: 49.25
Pecentage of female: 50.74999999999999
```

2.1.3 Question4: Plotted barh chart for gender distribution for male and female.

[29]:
```
values = [percentage_male, percentage_female]
labels = ['Male', 'Female']

plt.figure(figsize=(8, 6))
plt.barh(labels, values, color='lightcoral')  # Using barh for horizontal bar␣
↪chart
```

```
plt.title('Gender Distribution')
plt.xlabel('Percentage')
plt.ylabel('Gender')
plt.show()
```



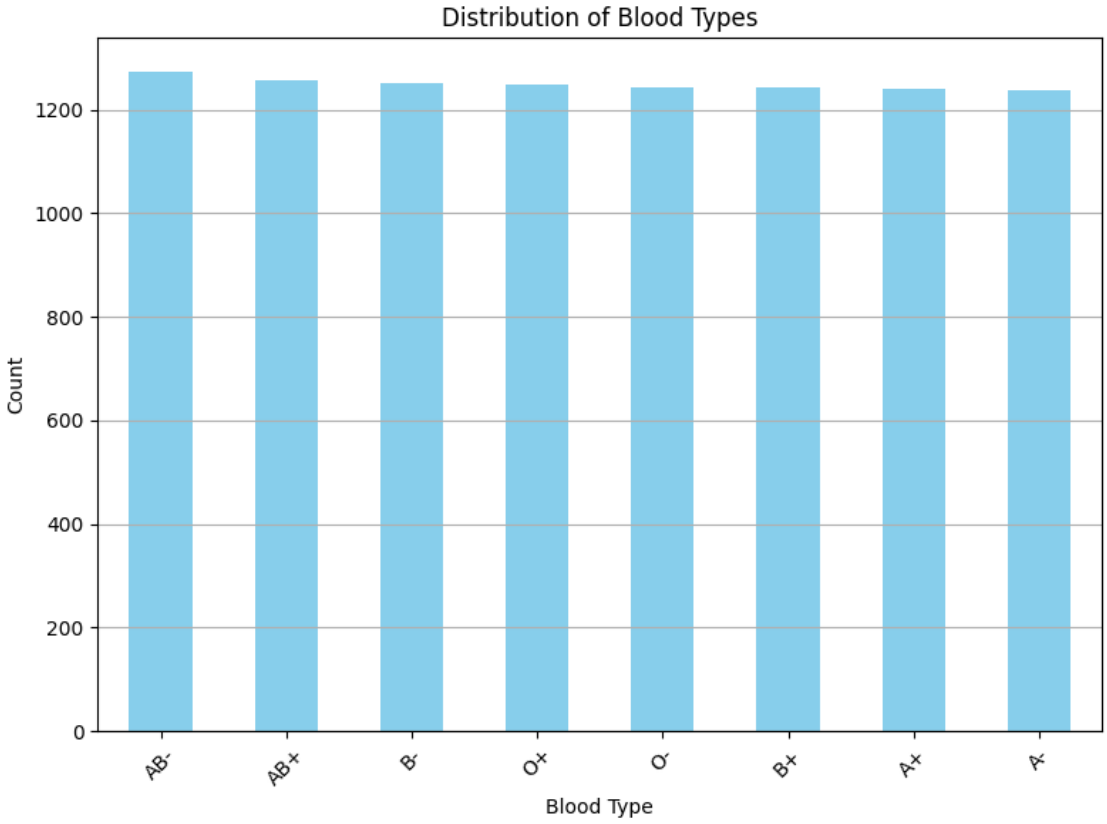
2.1.4 Questions5: Find the Categorical data for Blood type group and visualization with barchart.

[30]:
```
blood_group = df['Blood Type'].value_counts()
blood_group
```

[30]:
```
Blood Type
AB-    1275
AB+    1258
B-     1252
O+     1248
O-     1244
B+     1244
A+     1241
A-     1238
```

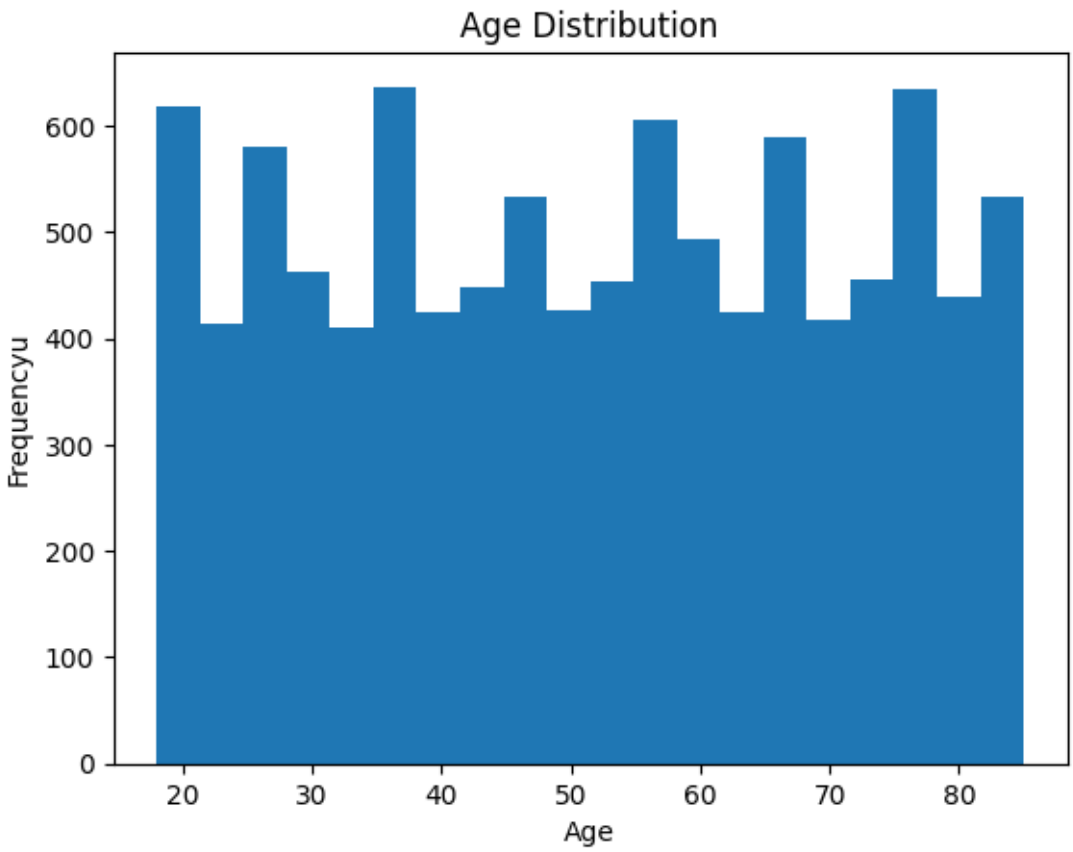
```
Name: count, dtype: int64
```

[31]:
```
plt.figure(figsize=(8, 6))
blood_group.plot(kind='bar', color='skyblue')
plt.title('Distribution of Blood Types')
plt.xlabel('Blood Type')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.grid(axis='y')  # Add gridlines on y-axis
plt.tight_layout()
plt.show()
```



2.1.5 Questions6: Find the frequency of the maximum age with histogram.

[6]:
```
plt.hist(df['Age'], bins=20)
plt.xlabel('Age')
plt.ylabel('Frequencyu')
plt.title('Age Distribution')
plt.show()
```


Age Distribution
Frequencyu
0
100
200
300
400
500
600
20
30
40
50
60
70
80
Age

diwali-sales-analysis

December 11, 2023

[1]:
```
# import python libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
```

[2]:
```
# import csv file
df = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')
```

[3]:
```
df.shape
```

[3]:
```
(11251, 15)
```

[4]:
```
df.head()
```

[4]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	unnamed1
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23952.0	NaN	NaN
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23934.0	NaN	NaN
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924.0	NaN	NaN
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	23912.0	NaN	NaN
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877.0	NaN	NaN

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11251 non-null  int64  
 1   Cust_name         11251 non-null  object 
 2   Product_ID        11251 non-null  object 
 3   Gender            11251 non-null  object 
 4   Age Group         11251 non-null  object 
 5   Age               11251 non-null  int64  
 6   Marital_Status    11251 non-null  int64  
 7   State             11251 non-null  object 
 8   Zone              11251 non-null  object 
 9   Occupation        11251 non-null  object 
 10  Product_Category  11251 non-null  object 
 11  Orders            11251 non-null  int64  
 12  Amount            11239 non-null  float64
 13  Status            0 non-null      float64
 14  unnamed1          0 non-null      float64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

[6]:
```
#drop unrelated/blank columns
df.drop(['Status', 'unnamed1'], axis=1, inplace=True)
```

[7]:
```
#check for null values
pd.isnull(df).sum()
```

[7]:
```
User_ID             0
Cust_name           0
Product_ID          0
Gender              0
Age Group           0
Age                 0
Marital_Status      0
State               0
Zone                0
Occupation          0
Product_Category    0
Orders              0
Amount             12
dtype: int64
```

[8]:
```
# drop null values
df.dropna(inplace=True)
```

[9]:
```
# change data type
df['Amount'] = df['Amount'].astype('int')
```

[10]:
```
df['Amount'].dtypes
```

[10]:
```
dtype('int32')
```

[11]:
```
df.columns
```

[11]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[12]:
```
#rename column
df.rename(columns= {'Marital_Status':'Shaadi'})
```

[12]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shaadi	State	Zone	Occupation	Product_Category	Orders	Amount
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	
…	…	…	…	…	…	…	…	…	…	…	…	…	
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Western	Chemical	Office	4	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	3	
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Central	Textile	Office	4	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southern	Agriculture	Office	3	
11250	1002744	Brumley	P00281742	F	18-25	19	0	Maharashtra	Western	Healthcare	Office	3	

```
0        23952
1        23934
2        23924
3        23912
4        23877
…          …
11246      370
11247      367
11248      213
11249      206
11250      188

[11239 rows x 13 columns]
```

[13]:
```
# describe() method returns description of the data in the DataFrame (i.e.␣
 ↪count, mean, std, etc)
df.describe()
```

[13]:

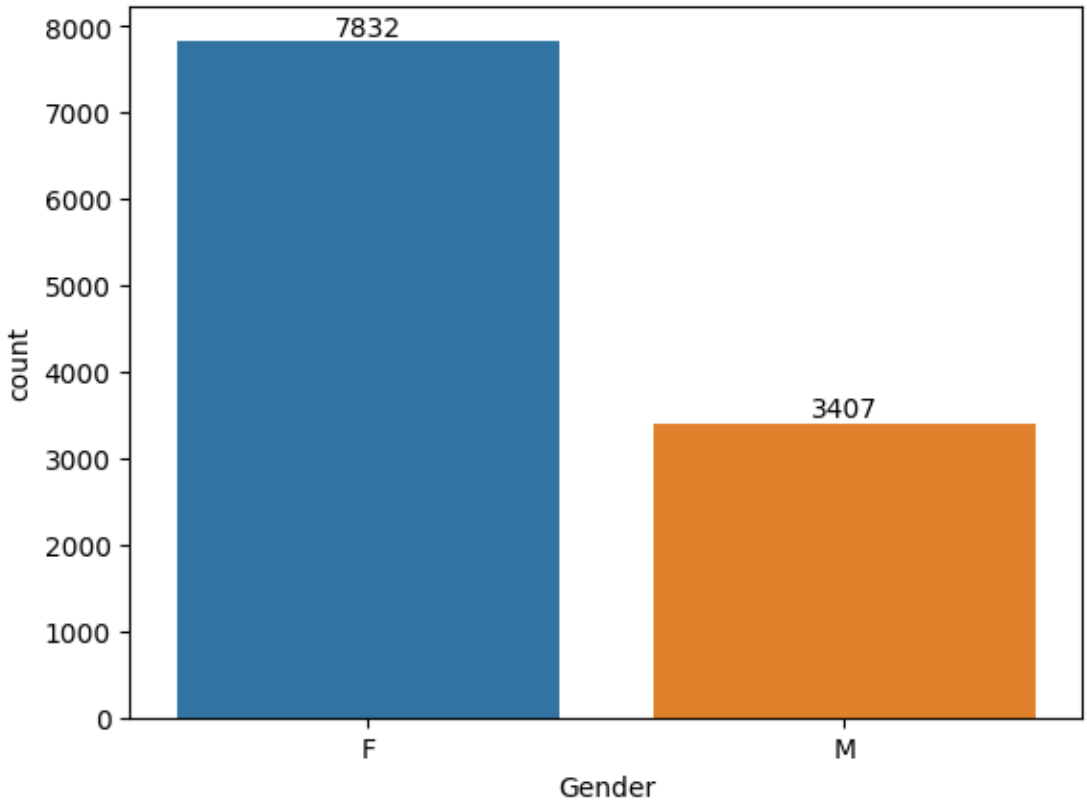
	User_ID	Age	Marital_Status	Orders	Amount
count	1.123900e+04	11239.000000	11239.000000	11239.000000	11239.000000
mean	1.003004e+06	35.410357	0.420055	2.489634	9453.610553
std	1.716039e+03	12.753866	0.493589	1.114967	5222.355168
min	1.000001e+06	12.000000	0.000000	1.000000	188.000000
25%	1.001492e+06	27.000000	0.000000	2.000000	5443.000000
50%	1.003064e+06	33.000000	0.000000	2.000000	8109.000000
75%	1.004426e+06	43.000000	1.000000	3.000000	12675.000000
max	1.006040e+06	92.000000	1.000000	4.000000	23952.000000

[14]:
```
# use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

[14]:

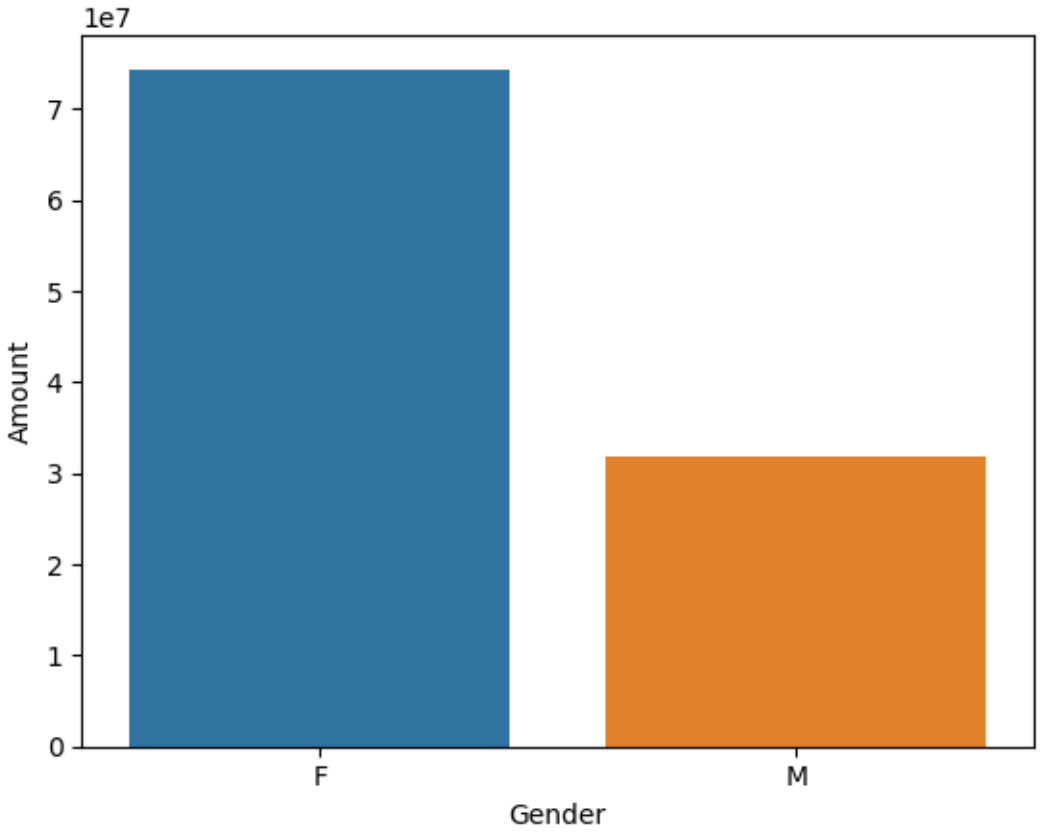
	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

[15]:
```
# plotting a bar chart for Gender and it's count

ax = sns.countplot(x = 'Gender',data = df)

for bars in ax.containers:
    ax.bar_label(bars)
```



[16]:
```
# plotting a bar chart for gender vs total amount

sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

sns.barplot(x = 'Gender',y= 'Amount' ,data = sales_gen)
```

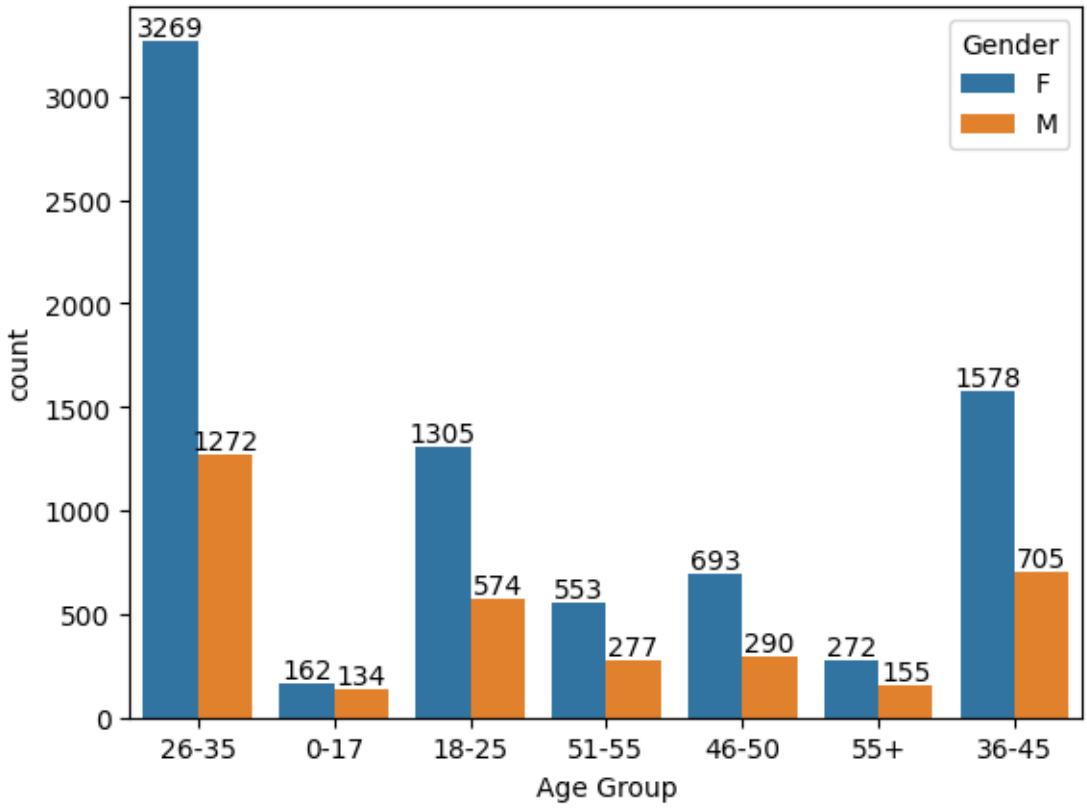
[16]:
```
<Axes: xlabel='Gender', ylabel='Amount'>
```



From above graphs we can see that most of the buyers are females and even the purchasing power of females are greater than men

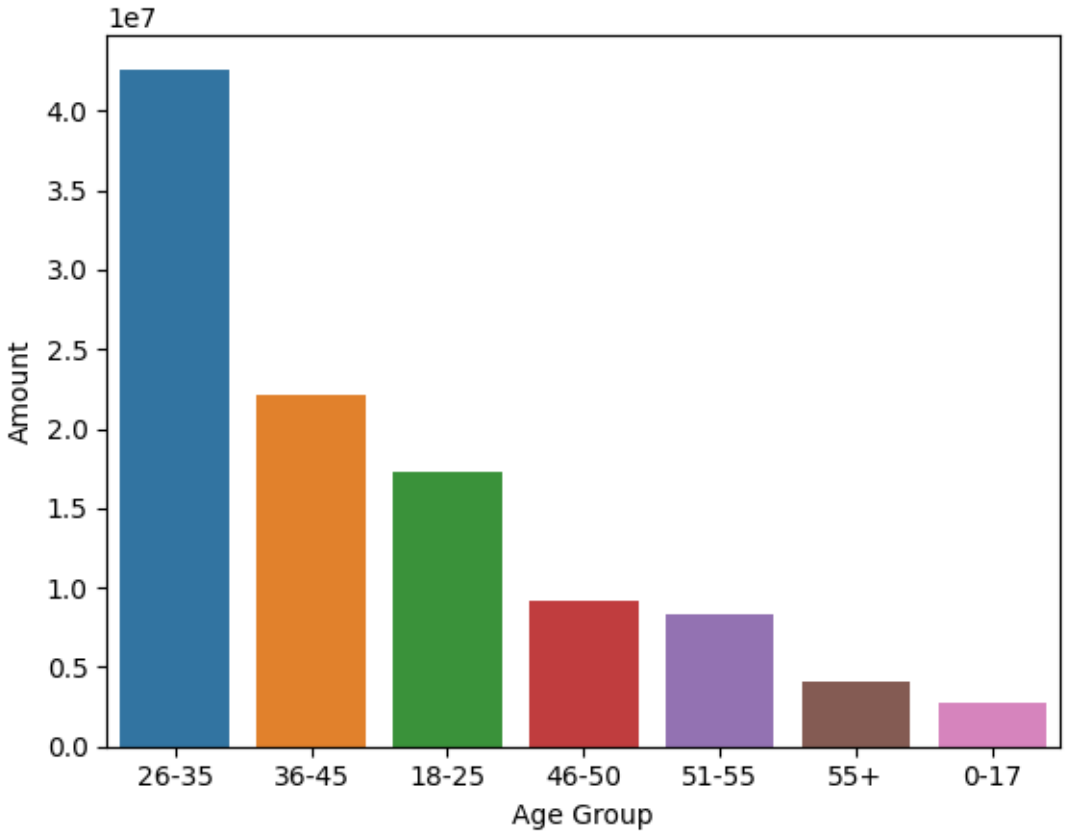
[ ]:
```
### Age
```

[17]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')

for bars in ax.containers:
    ax.bar_label(bars)
```



[18]:
```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

sns.barplot(x = 'Age Group',y= 'Amount' ,data = sales_age)
```

[18]: <Axes: xlabel='Age Group', ylabel='Amount'>

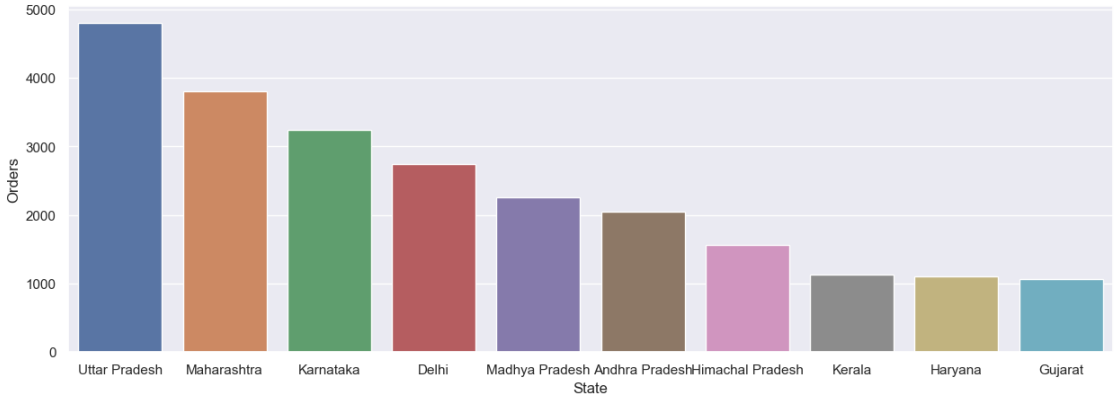


From above graphs we can see that most of the buyers are of age group between 26-35 yrs female

[ ]:
```
### State
```

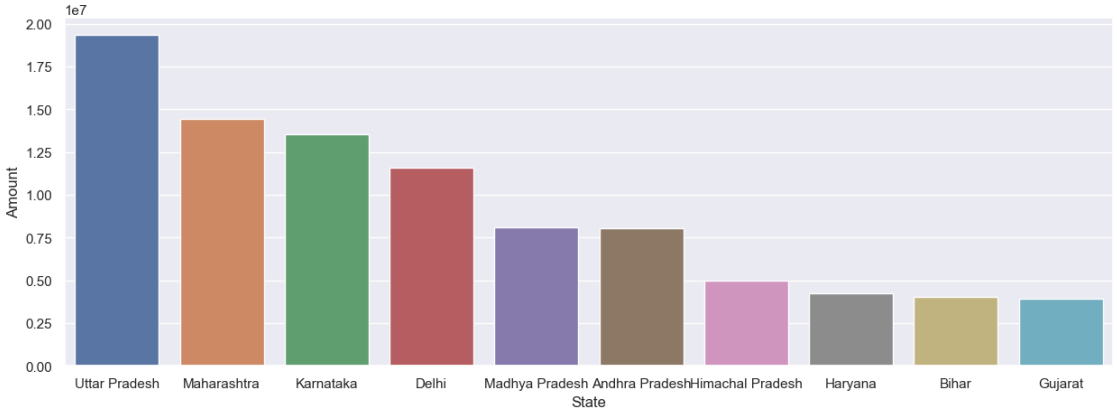
[19]:
```
# total number of orders from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Orders')
```

[19]:
```
<Axes: xlabel='State', ylabel='Orders'>
```



[20]:
```
# total amount/sales from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Amount')
```

[20]:
```
<Axes: xlabel='State', ylabel='Amount'>
```

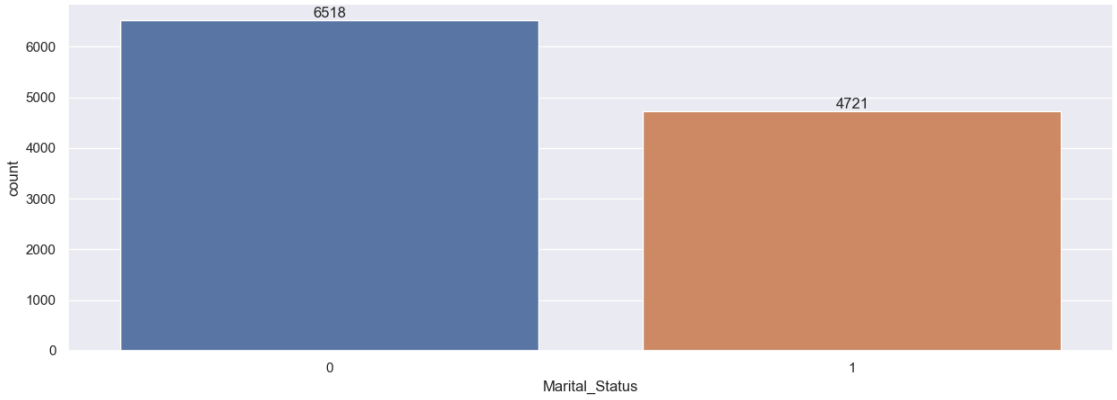


From above graphs we can see that most of the orders & total sales/amount are from Uttar Pradesh, Maharashtra and Karnataka respectively

[ ]:
```
### Marital Status
```

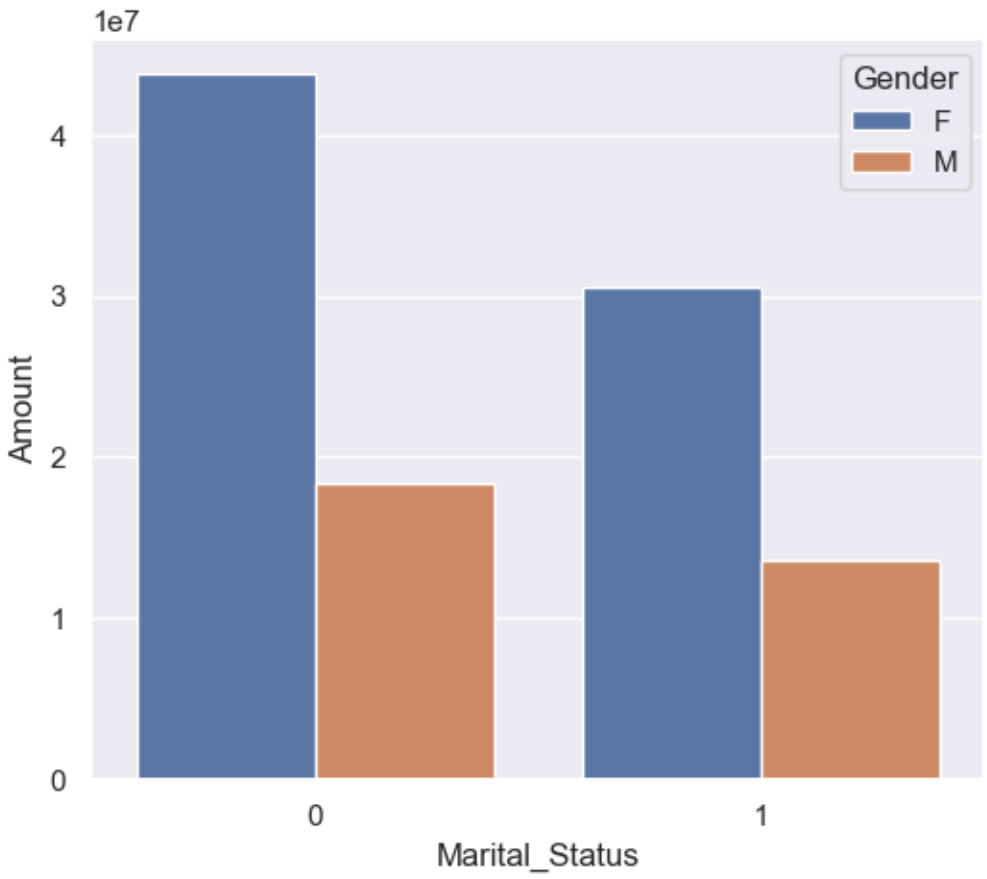
[21]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
```

```
for bars in ax.containers:
    ax.bar_label(bars)
```



[22]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'],␣
↪as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data = sales_state, x = 'Marital_Status',y= 'Amount', hue='Gender')
```

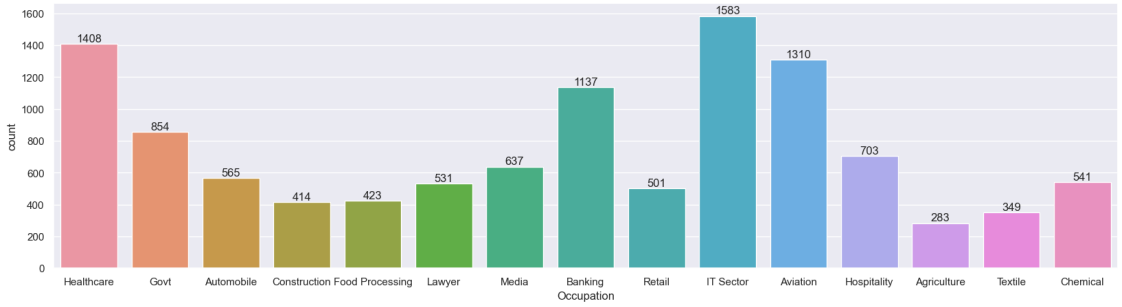
[22]:
```
<Axes: xlabel='Marital_Status', ylabel='Amount'>
```



From above graphs we can see that most of the buyers are married (women) and they have high purchasing power

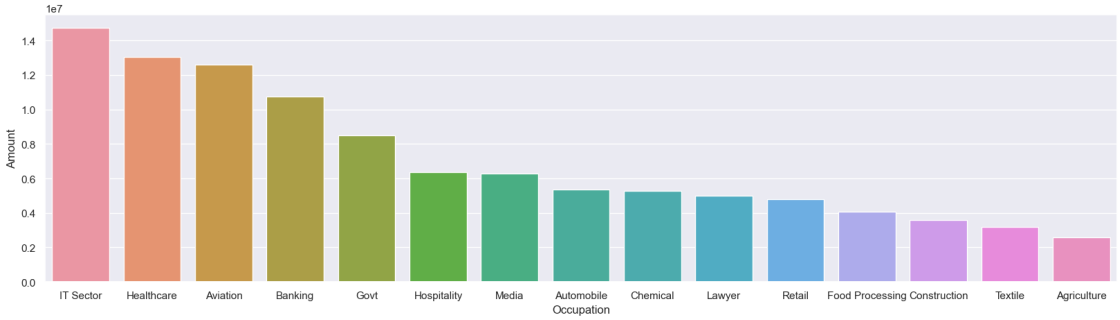
0.0.1 Occupation

[23]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

for bars in ax.containers:
    ax.bar_label(bars)
```



[24]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Occupation',y= 'Amount')
```

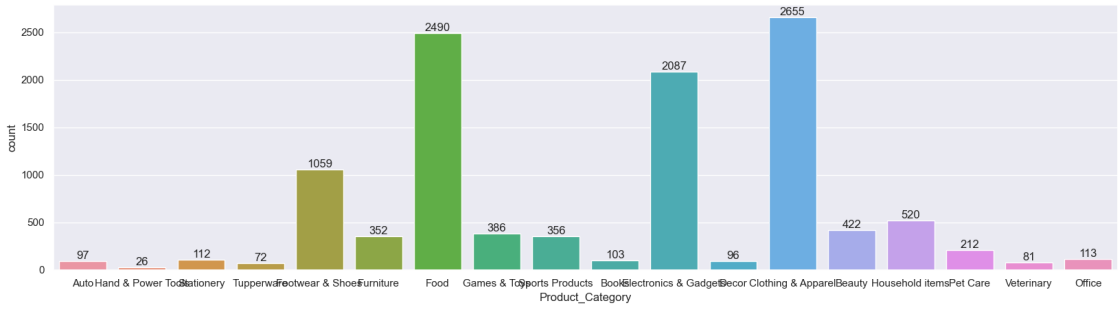
[24]:
```
<Axes: xlabel='Occupation', ylabel='Amount'>
```



From above graphs we can see that most of the buyers are working in IT, Healthcare and Aviation sector

[ ]:
```
### Product Category
```

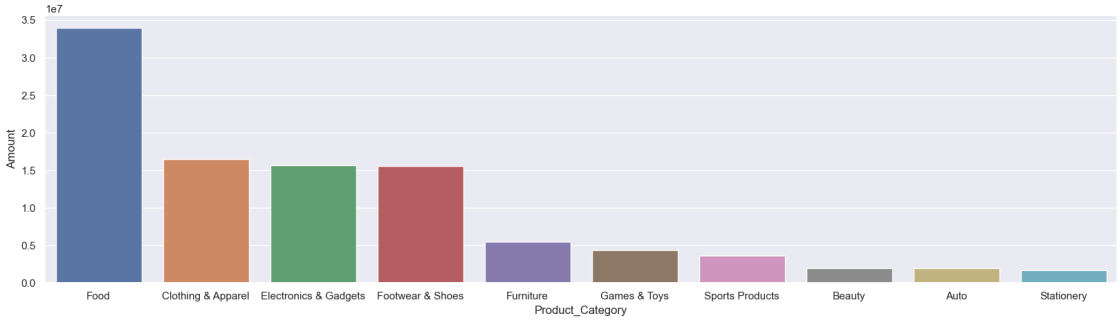
[25]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

for bars in ax.containers:
    ax.bar_label(bars)
```



[26]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)
```

```
sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_Category',y= 'Amount')
```

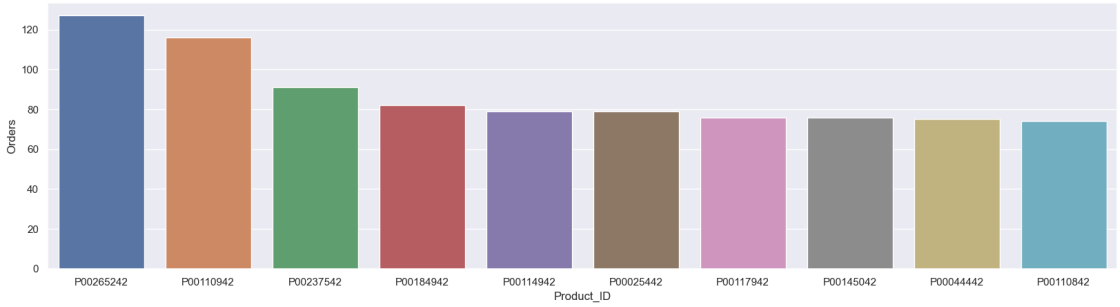
[26]: <Axes: xlabel='Product_Category', ylabel='Amount'>



From above graphs we can see that most of the sold products are from Food, Clothing and Electronics category

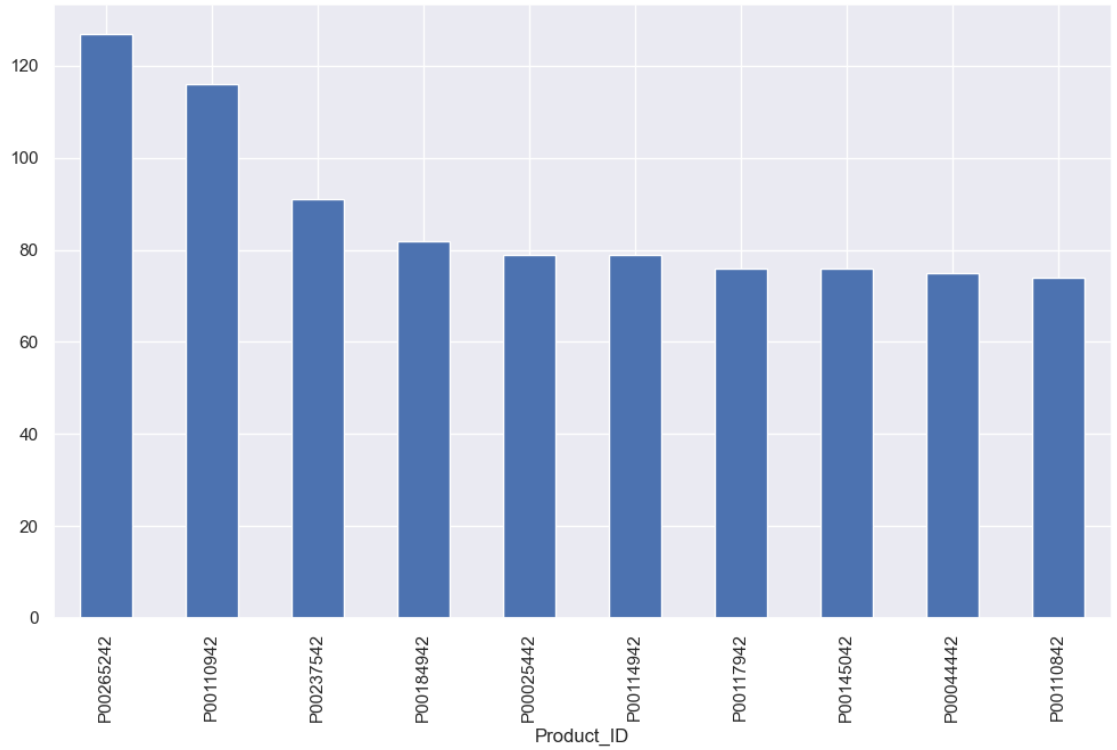
[27]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_ID',y= 'Orders')
```

[27]: <Axes: xlabel='Product_ID', ylabel='Orders'>



[28]:
```
# top 10 most sold products (same thing as above)

fig1, ax1 = plt.subplots(figsize=(12,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).
 ↪sort_values(ascending=False).plot(kind='bar')
```

[28]: <Axes: xlabel='Product_ID'>



[ ]:
```
## Conclusion:

###
```

Married women age group 26-35 yrs from UP, Maharastra and Karnataka working in IT, Healthcare and Aviation are more likely to buy products from Food, Clothing and Electronics category

Thank you!

bike-demand-prediction

December 11, 2023

1 Bike Demand Prediction

[1]:
```
# importing requried library

import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sn
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from math import sqrt
import time
import random
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
pd.options.mode.chained_assignment = None

%matplotlib inline
```

[2]:
```
# Reading dataset
raw_data = pd.read_csv("day.csv")
```

[3]:
```
raw_data.cnt.describe()
```

[3]:
```
count     731.000000
mean     4504.348837
std      1937.211452
min        22.000000
25%      3152.000000
50%      4548.000000
75%      5956.000000
```

```
max      8714.000000
Name: cnt, dtype: float64
```

[4]:
```
raw_data.head(3)
```

[4]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349

[5]:
```
raw_data.shape
```

[5]:
```
(731, 16)
```

We have 731 observations, 15 predictors and 1 target variable. Cnt is our target variable. Next examining variable types

[6]:
```
raw_data.dtypes
```

[6]:
```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

In the dataset season, yr, mnth, holiday, weekday, workingday, weathersit predictors should be

categorical type, but they are int64. In the next step mapping and categorical transformatin will be performed.

[8]:
```
#converting to categorical variable

categorical_variable =␣
 ↪["season","yr","mnth","holiday","weekday","workingday","weathersit"]

for var in categorical_variable:
    raw_data[var] = raw_data[var].astype("category")
```

[9]:
```
raw_data.head(2)
```

[9]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	\
0	1	2011-01-01	1	0	1	0	6	0	2	
1	2	2011-01-02	1	0	1	0	0	0	2	

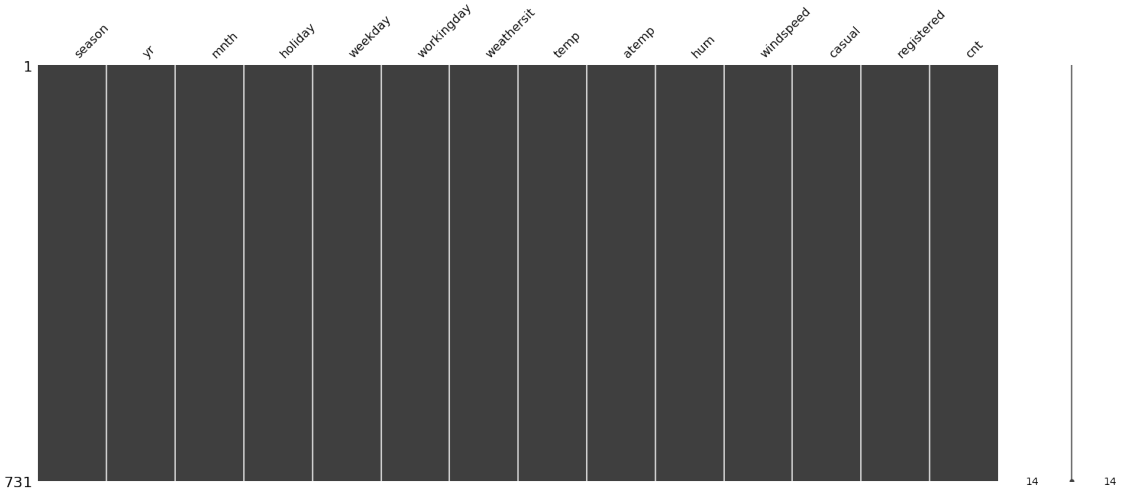
	temp	atemp	hum	windspeed	casual	registered	cnt
0	0.344167	0.363625	0.805833	0.160446	331	654	985
1	0.363478	0.353739	0.696087	0.248539	131	670	801

Droping variables which are not requried. 1. instant - index number 2. dteday- all the requried like month week day all ready present 3. casual and resgistered - their sum is equal to cnt ie. to the target variable

[10]:
```
raw_data = raw_data.drop(["instant","dteday"],axis = 1)
```

1.1 Missing value Analysis

[11]:
```
# We will perform missing value andlysis using missingno package
msno.matrix(raw_data)
```

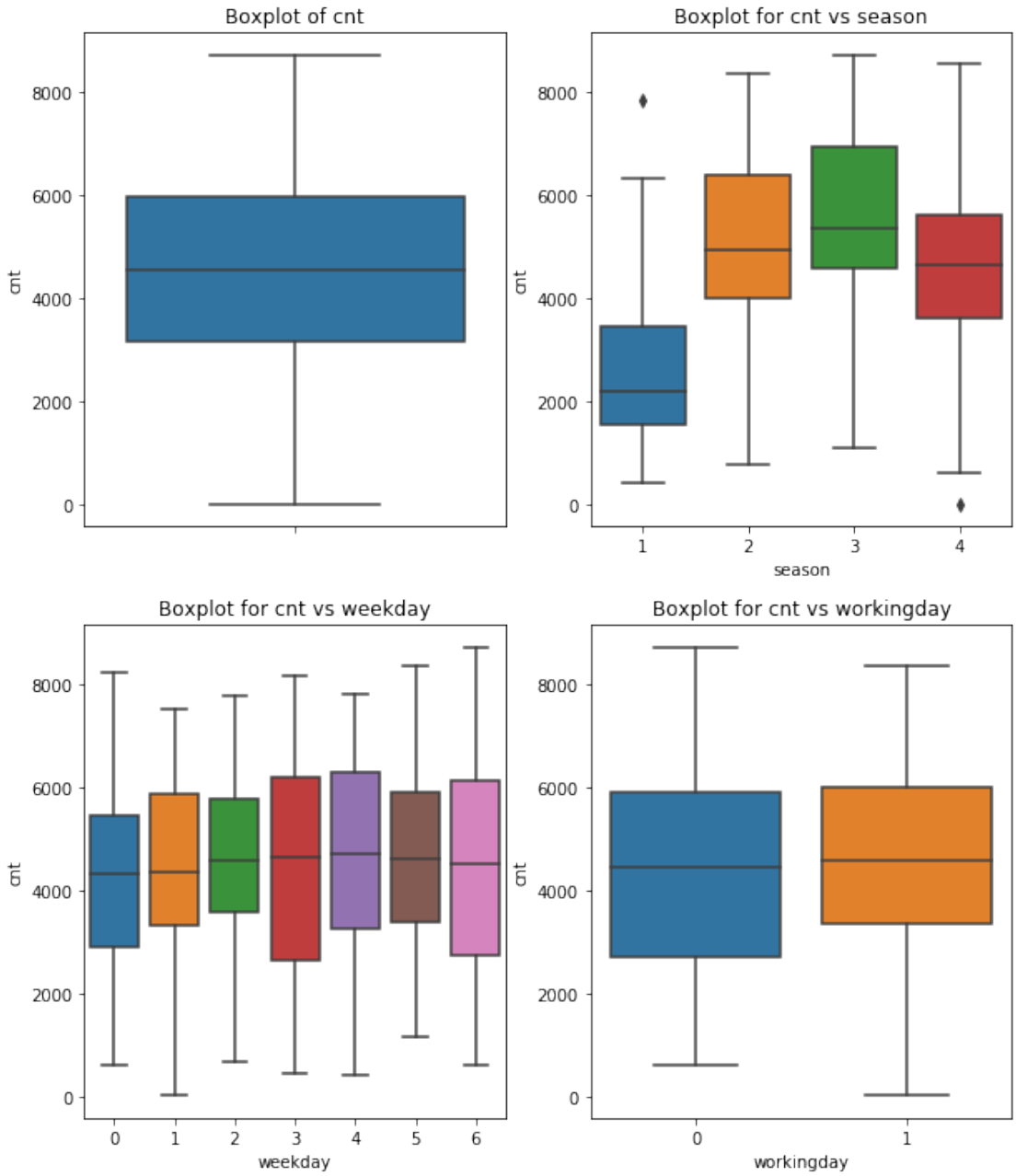


There are no missing values present in the dataset

Outliers Analysis

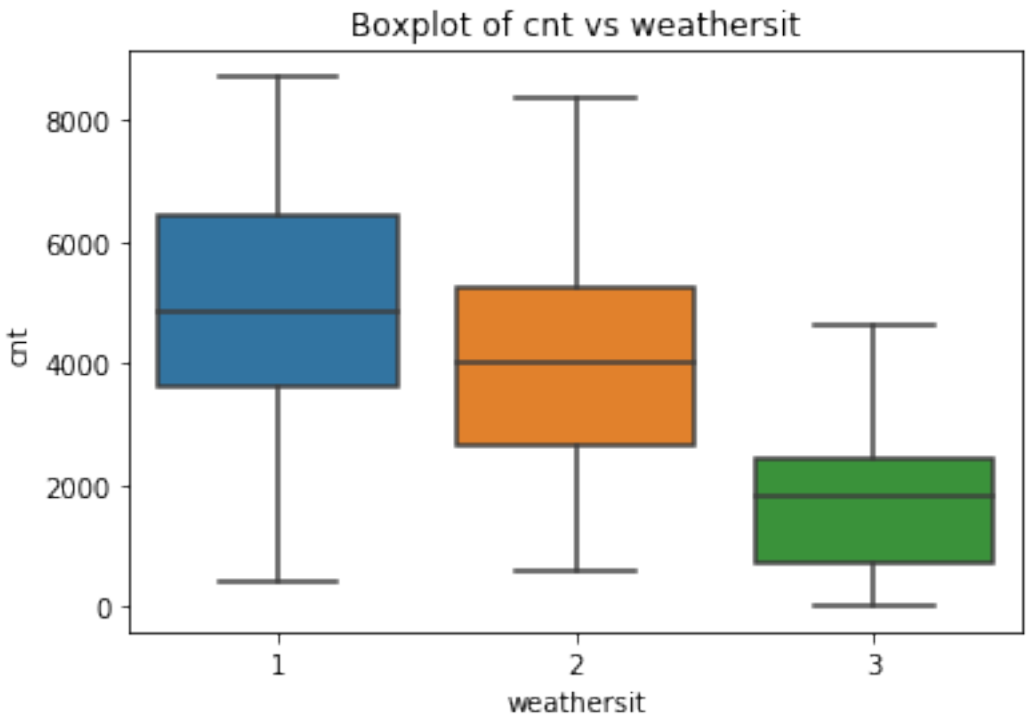
[12]:
```
fig, axes = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(10,12)
sn.boxplot(data=raw_data,y="cnt",orient='v',ax=axes[0][0])
sn.boxplot(data=raw_data,y="cnt",x="season",orient='v',ax=axes[0][1])
sn.boxplot(data=raw_data,y="cnt",x="weekday",orient="v",ax=axes[1][0])
sn.boxplot(data=raw_data,y="cnt",x="workingday",orient="v",ax=axes[1][1])

axes[0][0].set(ylabel='cnt',title = "Boxplot of cnt")
axes[0][1].set(xlabel="season",ylabel="cnt",title="Boxplot for cnt vs season")
axes[1][0].set(xlabel="weekday", ylabel="cnt",title="Boxplot for cnt vs␣
 ↪weekday")
axes[1][1].set(xlabel="workingday",ylabel="cnt",title="Boxplot for cnt vs␣
 ↪workingday")
```

[12]:
```
[Text(0,0.5,'cnt'),
 Text(0.5,0,'workingday'),
 Text(0.5,1,'Boxplot for cnt vs workingday')]
```



[13]:
```
fig.set_size_inches(8,12)
sn.boxplot(data=raw_data, x="weathersit",y="cnt").set_title("Boxplot of cnt vs␣
↪weathersit")
```

[13]: Text(0.5,1,'Boxplot of cnt vs weathersit')



From the above boxplots, it is evident that there are no outliers present in the cnt. Two things are clear. 1. Cnt is very low in spring season. 2. Cnt is maximum when weather is good and its minimum weather is bab.

1.2 Correlation Analysis

[14]:
```
churn_corr = raw_data.corr()
cmap = cmap=sn.diverging_palette(15, 250, as_cmap=True)

def magnify():
    return [dict(selector="th",
                 props=[("font-size", "7pt")]),
            dict(selector="td",
                 props=[('padding', "0em 0em")]),
            dict(selector="th:hover",
                 props=[("font-size", "12pt")]),
            dict(selector="tr:hover td:hover",
                 props=[('max-width', '200px'),
                        ('font-size', '12pt')])
]

churn_corr.style.background_gradient(cmap, axis=1)\
    .set_properties(**{'max-width': '90px', 'font-size': '15pt'})\
    .set_caption("Correlation matrix")\
```

```
    .set_precision(2)\
    .set_table_styles(magnify())
```

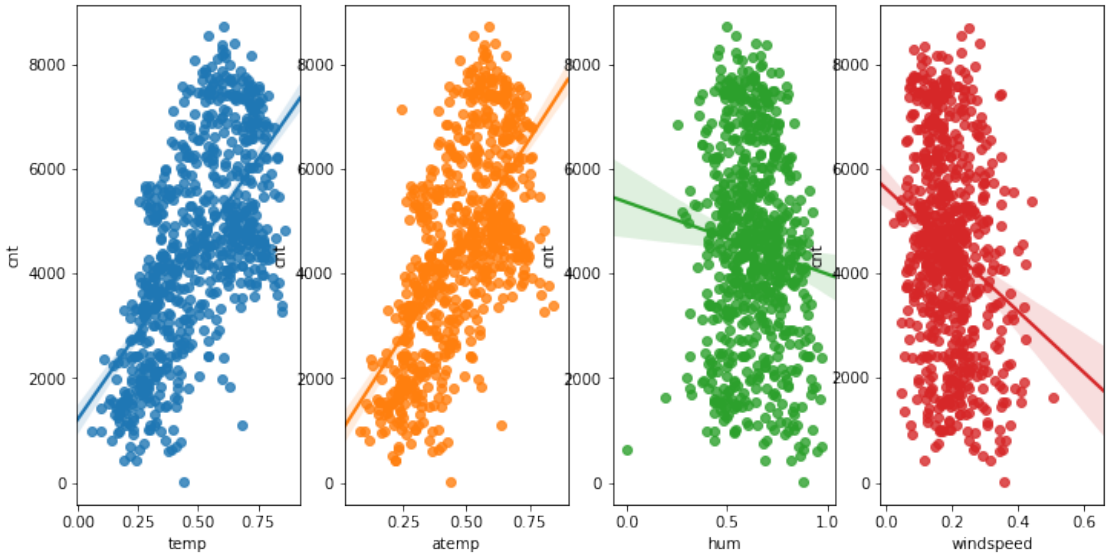
[14]: <pandas.io.formats.style.Styler at 0x1f63d96f438>

Finding of correlation analysis - 1. temp and atemp are highly correlated. 2. temp and atemp have positive and strong coorelation with cnt. 3. hum and windspeed have negative and weak correlation with cnt.

1.3 Bivariate analysis

[15]:
```
# Bivariate analysis of cnt and continous predictor

fig,(ax1,ax2,ax3,ax4) = plt.subplots(ncols=4)
fig.set_size_inches(12,6)

sn.regplot(x="temp",y="cnt",data=raw_data,ax=ax1)
sn.regplot(x="atemp",y="cnt",data=raw_data,ax=ax2)
sn.regplot(x="hum",y="cnt",data=raw_data,ax=ax3)
sn.regplot(x="windspeed",y="cnt",data=raw_data,ax=ax4)
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1f64070ec88>



From the above plot, it is evident that cnt has a positive linear relationship with temp and atemp. On the other hand, cnt has a negative linear relationship with windspeed. Humidity(hum) has a little negative linear relationship with cnt.

1.4 Distribution of target Variable

[16]:
```
fig,(ax1,ax2) = plt.subplots(ncols=2)
fig.set_size_inches(9,5)
sn.distplot(raw_data["cnt"],ax=ax1)
stats.probplot(raw_data["cnt"], dist='norm', fit=True, plot=ax2)
```

```
C:\Users\Rohit\AppData\Local\conda\conda\envs\ggpy36\lib\site-
packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is
deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

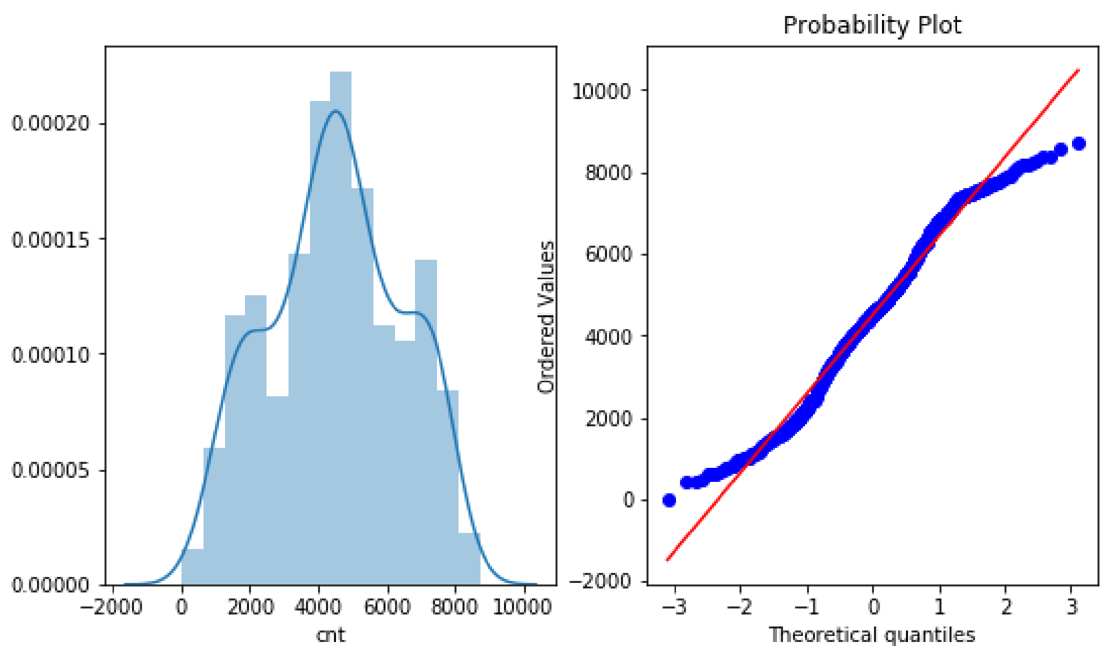
[16]:
```
((array([-3.10612952, -2.83371839, -2.68121219, -2.57340905, -2.48915191,
         -2.41955673, -2.36001798, -2.30782877, -2.26125818, -2.21912992,
         -2.18060696, -2.14507173, -2.11205508, -2.08119197, -2.05219258,
         -2.0248228 , -1.99889075, -1.97423711, -1.95072808, -1.92825019,
         -1.90670633, -1.88601273, -1.8660966 , -1.84689427, -1.82834975,
         -1.81041348, -1.79304141, -1.77619419, -1.75983653, -1.74393663,
         -1.72846577, -1.71339788, -1.69870925, -1.68437825, -1.67038506,
         -1.6567115 , -1.64334086, -1.63025771, -1.6174478 , -1.60489794,
         -1.59259587, -1.58053022, -1.56869036, -1.55706641, -1.54564912,
         -1.53442983, -1.52340042, -1.51255328, -1.50188124, -1.49137757,
         -1.4810359 , -1.47085025, -1.46081495, -1.45092464, -1.44117426,
         -1.431559  , -1.4220743 , -1.41271583, -1.40347947, -1.39436132,
         -1.38535765, -1.3764649 , -1.36767969, -1.35899879, -1.35041911,
         -1.3419377 , -1.33355173, -1.32525852, -1.31705546, -1.30894008,
         -1.30091001, -1.29296295, -1.28509673, -1.27730922, -1.26959842,
         -1.26196238, -1.25439922, -1.24690714, -1.2394844 , -1.23212934,
         -1.22484033, -1.21761582, -1.21045431, -1.20335435, -1.19631454,
         -1.18933352, -1.18240998, -1.17554267, -1.16873035, -1.16197185,
         -1.155266  , -1.14861171, -1.14200789, -1.13545351, -1.12894754,
         -1.12248901, -1.11607697, -1.10971049, -1.10338867, -1.09711064,
         -1.09087556, -1.08468261, -1.07853098, -1.07241989, -1.06634859,
         -1.06031635, -1.05432244, -1.04836618, -1.04244688, -1.03656388,
         -1.03071654, -1.02490423, -1.01912634, -1.01338227, -1.00767144,
         -1.0019933 , -0.99634727, -0.99073283, -0.98514945, -0.9795966 ,
         -0.97407381, -0.96858056, -0.96311639, -0.95768082, -0.9522734 ,
         -0.94689368, -0.94154123, -0.93621562, -0.93091643, -0.92564325,
         -0.92039569, -0.91517335, -0.90997585, -0.90480282, -0.89965388,
         -0.89452869, -0.88942689, -0.88434814, -0.87929209, -0.87425842,
         -0.86924681, -0.86425694, -0.85928849, -0.85434116, -0.84941466,
         -0.84450869, -0.83962296, -0.83475719, -0.8299111 , -0.82508442,
         -0.8202769 , -0.81548825, -0.81071823, -0.80596659, -0.80123308,
         -0.79651745, -0.79181947, -0.78713889, -0.7824755 , -0.77782907,
         -0.77319937, -0.76858618, -0.76398929, -0.75940848, -0.75484356,
         -0.75029432, -0.74576055, -0.74124205, -0.73673864, -0.73225012,
         -0.72777631, -0.72331702, -0.71887206, -0.71444126, -0.71002444,
         -0.70562143, -0.70123206, -0.69685616, -0.69249356, -0.6881441 ,
```

```
-0.68380762, -0.67948396, -0.67517297, -0.67087448, -0.66658836,
-0.66231445, -0.6580526 , -0.65380267, -0.64956452, -0.64533801,
-0.64112299, -0.63691932, -0.63272689, -0.62854555, -0.62437516,
-0.62021561, -0.61606676, -0.61192849, -0.60780067, -0.60368318,
-0.59957591, -0.59547872, -0.5913915 , -0.58731414, -0.58324652,
-0.57918853, -0.57514005, -0.57110098, -0.5670712 , -0.56305061,
-0.5590391 , -0.55503657, -0.55104291, -0.54705802, -0.5430818 ,
-0.53911414, -0.53515496, -0.53120414, -0.5272616 , -0.52332724,
-0.51940096, -0.51548267, -0.51157228, -0.5076697 , -0.50377484,
-0.4998876 , -0.4960079 , -0.49213565, -0.48827077, -0.48441317,
-0.48056276, -0.47671946, -0.4728832 , -0.46905388, -0.46523142,
-0.46141575, -0.45760679, -0.45380445, -0.45000867, -0.44621936,
-0.44243644, -0.43865984, -0.43488949, -0.43112532, -0.42736724,
-0.42361519, -0.41986909, -0.41612887, -0.41239447, -0.40866581,
-0.40494282, -0.40122544, -0.39751359, -0.39380721, -0.39010623,
-0.38641059, -0.38272022, -0.37903506, -0.37535503, -0.37168008,
-0.36801015, -0.36434516, -0.36068506, -0.35702979, -0.35337928,
-0.34973347, -0.34609231, -0.34245573, -0.33882367, -0.33519607,
-0.33157289, -0.32795405, -0.3243395 , -0.32072918, -0.31712304,
-0.31352101, -0.30992305, -0.3063291 , -0.3027391 , -0.299153  ,
-0.29557074, -0.29199227, -0.28841753, -0.28484648, -0.28127906,
-0.27771521, -0.27415488, -0.27059803, -0.2670446 , -0.26349453,
-0.25994779, -0.25640431, -0.25286405, -0.24932695, -0.24579297,
-0.24226206, -0.23873416, -0.23520924, -0.23168723, -0.2281681 ,
-0.22465178, -0.22113825, -0.21762744, -0.21411931, -0.21061382,
-0.20711091, -0.20361054, -0.20011267, -0.19661724, -0.19312421,
-0.18963354, -0.18614518, -0.18265908, -0.17917519, -0.17569349,
-0.17221391, -0.16873641, -0.16526095, -0.16178749, -0.15831598,
-0.15484638, -0.15137863, -0.14791271, -0.14444857, -0.14098615,
-0.13752543, -0.13406635, -0.13060888, -0.12715297, -0.12369857,
-0.12024565, -0.11679416, -0.11334407, -0.10989532, -0.10644788,
-0.1030017 , -0.09955675, -0.09611298, -0.09267034, -0.08922881,
-0.08578833, -0.08234887, -0.07891038, -0.07547282, -0.07203616,
-0.06860034, -0.06516534, -0.0617311 , -0.05829759, -0.05486477,
-0.0514326 , -0.04800103, -0.04457003, -0.04113955, -0.03770955,
-0.03428   , -0.03085085, -0.02742207, -0.0239936 , -0.02056542,
-0.01713748, -0.01370974, -0.01028217, -0.00685471, -0.00342734,
 0.        ,  0.00342734,  0.00685471,  0.01028217,  0.01370974,
 0.01713748,  0.02056542,  0.0239936 ,  0.02742207,  0.03085085,
 0.03428   ,  0.03770955,  0.04113955,  0.04457003,  0.04800103,
 0.0514326 ,  0.05486477,  0.05829759,  0.0617311 ,  0.06516534,
 0.06860034,  0.07203616,  0.07547282,  0.07891038,  0.08234887,
 0.08578833,  0.08922881,  0.09267034,  0.09611298,  0.09955675,
 0.1030017 ,  0.10644788,  0.10989532,  0.11334407,  0.11679416,
 0.12024565,  0.12369857,  0.12715297,  0.13060888,  0.13406635,
 0.13752543,  0.14098615,  0.14444857,  0.14791271,  0.15137863,
 0.15484638,  0.15831598,  0.16178749,  0.16526095,  0.16873641,
```

```
0.17221391,  0.17569349,  0.17917519,  0.18265908,  0.18614518,
0.18963354,  0.19312421,  0.19661724,  0.20011267,  0.20361054,
0.20711091,  0.21061382,  0.21411931,  0.21762744,  0.22113825,
0.22465178,  0.2281681 ,  0.23168723,  0.23520924,  0.23873416,
0.24226206,  0.24579297,  0.24932695,  0.25286405,  0.25640431,
0.25994779,  0.26349453,  0.2670446 ,  0.27059803,  0.27415488,
0.27771521,  0.28127906,  0.28484648,  0.28841753,  0.29199227,
0.29557074,  0.299153  ,  0.3027391 ,  0.3063291 ,  0.30992305,
0.31352101,  0.31712304,  0.32072918,  0.3243395 ,  0.32795405,
0.33157289,  0.33519607,  0.33882367,  0.34245573,  0.34609231,
0.34973347,  0.35337928,  0.35702979,  0.36068506,  0.36434516,
0.36801015,  0.37168008,  0.37535503,  0.37903506,  0.38272022,
0.38641059,  0.39010623,  0.39380721,  0.39751359,  0.40122544,
0.40494282,  0.40866581,  0.41239447,  0.41612887,  0.41986909,
0.42361519,  0.42736724,  0.43112532,  0.43488949,  0.43865984,
0.44243644,  0.44621936,  0.45000867,  0.45380445,  0.45760679,
0.46141575,  0.46523142,  0.46905388,  0.4728832 ,  0.47671946,
0.48056276,  0.48441317,  0.48827077,  0.49213565,  0.4960079 ,
0.4998876 ,  0.50377484,  0.5076697 ,  0.51157228,  0.51548267,
0.51940096,  0.52332724,  0.5272616 ,  0.53120414,  0.53515496,
0.53911414,  0.5430818 ,  0.54705802,  0.55104291,  0.55503657,
0.5590391 ,  0.56305061,  0.5670712 ,  0.57110098,  0.57514005,
0.57918853,  0.58324652,  0.58731414,  0.5913915 ,  0.59547872,
0.59957591,  0.60368318,  0.60780067,  0.61192849,  0.61606676,
0.62021561,  0.62437516,  0.62854555,  0.63272689,  0.63691932,
0.64112299,  0.64533801,  0.64956452,  0.65380267,  0.6580526 ,
0.66231445,  0.66658836,  0.67087448,  0.67517297,  0.67948396,
0.68380762,  0.6881441 ,  0.69249356,  0.69685616,  0.70123206,
0.70562143,  0.71002444,  0.71444126,  0.71887206,  0.72331702,
0.72777631,  0.73225012,  0.73673864,  0.74124205,  0.74576055,
0.75029432,  0.75484356,  0.75940848,  0.76398929,  0.76858618,
0.77319937,  0.77782907,  0.7824755 ,  0.78713889,  0.79181947,
0.79651745,  0.80123308,  0.80596659,  0.81071823,  0.81548825,
0.8202769 ,  0.82508442,  0.8299111 ,  0.83475719,  0.83962296,
0.84450869,  0.84941466,  0.85434116,  0.85928849,  0.86425694,
0.86924681,  0.87425842,  0.87929209,  0.88434814,  0.88942689,
0.89452869,  0.89965388,  0.90480282,  0.90997585,  0.91517335,
0.92039569,  0.92564325,  0.93091643,  0.93621562,  0.94154123,
0.94689368,  0.9522734 ,  0.95768082,  0.96311639,  0.96858056,
0.97407381,  0.9795966 ,  0.98514945,  0.99073283,  0.99634727,
1.0019933 ,  1.00767144,  1.01338227,  1.01912634,  1.02490423,
1.03071654,  1.03656388,  1.04244688,  1.04836618,  1.05432244,
1.06031635,  1.06634859,  1.07241989,  1.07853098,  1.08468261,
1.09087556,  1.09711064,  1.10338867,  1.10971049,  1.11607697,
1.12248901,  1.12894754,  1.13545351,  1.14200789,  1.14861171,
1.155266  ,  1.16197185,  1.16873035,  1.17554267,  1.18240998,
1.18933352,  1.19631454,  1.20335435,  1.21045431,  1.21761582,
```

```
        1.22484033,  1.23212934,  1.2394844 ,  1.24690714,  1.25439922,
        1.26196238,  1.26959842,  1.27730922,  1.28509673,  1.29296295,
        1.30091001,  1.30894008,  1.31705546,  1.32525852,  1.33355173,
        1.3419377 ,  1.35041911,  1.35899879,  1.36767969,  1.3764649 ,
        1.38535765,  1.39436132,  1.40347947,  1.41271583,  1.4220743 ,
        1.431559  ,  1.44117426,  1.45092464,  1.46081495,  1.47085025,
        1.4810359 ,  1.49137757,  1.50188124,  1.51255328,  1.52340042,
        1.53442983,  1.54564912,  1.55706641,  1.56869036,  1.58053022,
        1.59259587,  1.60489794,  1.6174478 ,  1.63025771,  1.64334086,
        1.6567115 ,  1.67038506,  1.68437825,  1.69870925,  1.71339788,
        1.72846577,  1.74393663,  1.75983653,  1.77619419,  1.79304141,
        1.81041348,  1.82834975,  1.84689427,  1.8660966 ,  1.88601273,
        1.90670633,  1.92825019,  1.95072808,  1.97423711,  1.99889075,
        2.0248228 ,  2.05219258,  2.08119197,  2.11205508,  2.14507173,
        2.18060696,  2.21912992,  2.26125818,  2.30782877,  2.36001798,
        2.41955673,  2.48915191,  2.57340905,  2.68121219,  2.83371839,
        3.10612952]),
array([  22,  431,  441,  506,  605,  623,  627,  683,  705,  754,  795,
        801,  822,  920,  959,  981,  985,  986, 1000, 1005, 1011, 1013,
       1027, 1096, 1096, 1098, 1107, 1115, 1162, 1162, 1167, 1204, 1248,
       1263, 1301, 1317, 1321, 1341, 1349, 1360, 1406, 1416, 1421, 1446,
       1450, 1461, 1471, 1472, 1495, 1501, 1510, 1526, 1529, 1530, 1536,
       1538, 1543, 1550, 1562, 1589, 1600, 1605, 1606, 1607, 1623, 1635,
       1650, 1683, 1685, 1685, 1693, 1708, 1712, 1746, 1749, 1787, 1795,
       1796, 1807, 1812, 1815, 1817, 1834, 1842, 1851, 1865, 1872, 1891,
       1913, 1917, 1927, 1944, 1951, 1969, 1977, 1977, 1985, 1996, 2028,
       2034, 2046, 2056, 2077, 2077, 2114, 2115, 2121, 2132, 2133, 2134,
       2162, 2169, 2177, 2192, 2209, 2210, 2227, 2236, 2252, 2277, 2294,
       2298, 2302, 2311, 2368, 2376, 2395, 2402, 2416, 2417, 2423, 2424,
       2424, 2425, 2425, 2429, 2431, 2432, 2455, 2471, 2475, 2485, 2493,
       2496, 2566, 2594, 2633, 2659, 2660, 2689, 2703, 2710, 2729, 2732,
       2739, 2743, 2744, 2765, 2792, 2802, 2808, 2832, 2843, 2895, 2913,
       2914, 2918, 2927, 2933, 2935, 2947, 2999, 3005, 3053, 3068, 3068,
       3071, 3095, 3115, 3117, 3126, 3129, 3141, 3163, 3190, 3194, 3204,
       3214, 3214, 3228, 3239, 3243, 3249, 3267, 3272, 3285, 3292, 3310,
       3322, 3331, 3333, 3348, 3351, 3351, 3368, 3372, 3376, 3387, 3389,
       3392, 3403, 3409, 3422, 3423, 3425, 3429, 3456, 3485, 3487, 3510,
       3520, 3523, 3542, 3544, 3570, 3574, 3577, 3598, 3606, 3613, 3614,
       3620, 3623, 3624, 3641, 3644, 3649, 3659, 3663, 3669, 3709, 3717,
       3727, 3740, 3744, 3747, 3750, 3761, 3767, 3777, 3784, 3784, 3785,
       3786, 3805, 3811, 3820, 3830, 3831, 3840, 3846, 3855, 3867, 3872,
       3873, 3894, 3907, 3910, 3915, 3922, 3926, 3940, 3944, 3956, 3958,
       3959, 3974, 3974, 3982, 4010, 4023, 4035, 4036, 4040, 4046, 4058,
       4066, 4067, 4068, 4073, 4073, 4075, 4086, 4094, 4097, 4098, 4098,
       4105, 4109, 4118, 4120, 4123, 4127, 4128, 4150, 4151, 4153, 4154,
       4169, 4182, 4186, 4187, 4189, 4191, 4195, 4195, 4205, 4220, 4258,
       4266, 4270, 4274, 4274, 4294, 4302, 4304, 4308, 4318, 4322, 4326,
```

```
       4332, 4333, 4334, 4338, 4339, 4342, 4352, 4359, 4362, 4363, 4367,
       4375, 4378, 4381, 4390, 4400, 4401, 4401, 4433, 4451, 4456, 4458,
       4459, 4459, 4460, 4475, 4484, 4486, 4492, 4507, 4509, 4511, 4521,
       4539, 4541, 4548, 4549, 4553, 4563, 4569, 4570, 4575, 4576, 4579,
       4585, 4586, 4590, 4592, 4595, 4602, 4608, 4629, 4630, 4634, 4639,
       4648, 4649, 4649, 4656, 4660, 4661, 4665, 4669, 4672, 4677, 4679,
       4687, 4694, 4708, 4713, 4714, 4717, 4725, 4727, 4744, 4748, 4758,
       4758, 4760, 4763, 4765, 4773, 4780, 4785, 4788, 4790, 4792, 4795,
       4803, 4826, 4833, 4835, 4839, 4840, 4844, 4845, 4862, 4864, 4866,
       4881, 4891, 4905, 4906, 4911, 4916, 4917, 4940, 4966, 4968, 4972,
       4978, 4985, 4990, 4991, 4996, 5008, 5010, 5020, 5026, 5035, 5041,
       5046, 5047, 5058, 5062, 5084, 5087, 5099, 5102, 5107, 5115, 5115,
       5117, 5119, 5119, 5130, 5138, 5146, 5169, 5170, 5180, 5191, 5191,
       5202, 5202, 5204, 5217, 5225, 5255, 5259, 5260, 5260, 5267, 5298,
       5302, 5305, 5312, 5312, 5315, 5319, 5323, 5336, 5342, 5345, 5362,
       5375, 5382, 5409, 5409, 5423, 5424, 5445, 5459, 5463, 5464, 5478,
       5495, 5499, 5501, 5511, 5515, 5531, 5532, 5538, 5557, 5558, 5566,
       5572, 5582, 5585, 5611, 5629, 5633, 5634, 5668, 5686, 5687, 5698,
       5698, 5713, 5728, 5729, 5740, 5743, 5786, 5805, 5810, 5823, 5847,
       5847, 5870, 5875, 5892, 5895, 5905, 5918, 5923, 5936, 5976, 5986,
       5992, 6031, 6034, 6041, 6043, 6043, 6053, 6073, 6093, 6118, 6133,
       6140, 6153, 6169, 6192, 6196, 6203, 6207, 6211, 6227, 6230, 6233,
       6234, 6235, 6241, 6269, 6273, 6290, 6296, 6299, 6304, 6312, 6359,
       6370, 6392, 6398, 6421, 6436, 6457, 6460, 6530, 6536, 6536, 6544,
       6565, 6569, 6572, 6591, 6591, 6597, 6598, 6606, 6624, 6639, 6660,
       6664, 6685, 6691, 6734, 6770, 6772, 6778, 6779, 6784, 6786, 6824,
       6824, 6825, 6830, 6852, 6855, 6857, 6861, 6864, 6869, 6871, 6879,
       6883, 6883, 6889, 6891, 6904, 6917, 6966, 6969, 6978, 6998, 7001,
       7006, 7013, 7030, 7040, 7055, 7058, 7105, 7109, 7112, 7129, 7132,
       7148, 7175, 7216, 7261, 7264, 7273, 7282, 7286, 7290, 7328, 7333,
       7335, 7338, 7347, 7350, 7359, 7363, 7375, 7384, 7393, 7403, 7410,
       7415, 7421, 7424, 7429, 7436, 7442, 7444, 7446, 7458, 7460, 7461,
       7466, 7494, 7498, 7499, 7504, 7509, 7525, 7534, 7534, 7538, 7570,
       7572, 7580, 7582, 7591, 7592, 7605, 7639, 7641, 7665, 7691, 7693,
       7697, 7702, 7713, 7720, 7733, 7736, 7765, 7767, 7804, 7836, 7852,
       7865, 7870, 7907, 7965, 8009, 8090, 8120, 8156, 8167, 8173, 8227,
       8294, 8362, 8395, 8555, 8714], dtype=int64)),
(1925.1274361641945, 4504.3488372093025, 0.9908084868276722))
```



As we can see, out cnt variable is very close to normal distribution.

Preprocessing original data and Spliting into train and test data

[17]:
```
# selecting predictors
train_feature_space = raw_data.iloc[:,raw_data.columns != 'cnt']
# selecting target class
target_class = raw_data.iloc[:,raw_data.columns == 'cnt']
```

[18]:
```
#droping atemp due to multicollinearity
#droping casual and registered because there sum is equal to target variable ie.
 ↪ 'cnt'

train_feature_space = train_feature_space.
 ↪drop(["atemp","casual","registered"],axis = 1)
```

[19]:
```
train_feature_space.shape
```

[19]: (731, 10)

[20]:
```
# creating training and test set
training_set, test_set, train_taget, test_target =␣
 ↪train_test_split(train_feature_space,
                                                                    ␣
 ↪target_class,
```

```
                                                                    test_size =␣
↪0.30,
                                                                              ␣
↪random_state = 456)

# Cleaning test sets to avoid future warning messages
train_taget = train_taget.values.ravel()
test_target = test_target.values.ravel()
```

[ ]:

1.5 Model1 Linear Regression Model

[21]:
```
X = training_set
X = sm.add_constant(X)
y= np.log(train_taget)

model = sm.OLS(y, X.astype(float)).fit()
```

[22]:
```
model.summary()
```

[22]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.654
Model:                            OLS   Adj. R-squared:                  0.647
Method:                 Least Squares   F-statistic:                     94.40
Date:                Fri, 03 Aug 2018   Prob (F-statistic):          2.20e-108
Time:                        13:51:38   Log-Likelihood:                -184.70
No. Observations:                 511   AIC:                             391.4
Df Residuals:                     500   BIC:                             438.0
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.6112      0.110     69.224      0.000       7.395       7.827
season         0.1286      0.026      4.865      0.000       0.077       0.180
yr             0.4818      0.031     15.339      0.000       0.420       0.543
mnth          -0.0069      0.008     -0.834      0.405      -0.023       0.009
holiday       -0.1800      0.099     -1.815      0.070      -0.375       0.015
weekday        0.0133      0.008      1.670      0.095      -0.002       0.029
workingday     0.0577      0.035      1.655      0.099      -0.011       0.126
weathersit    -0.2331      0.037     -6.228      0.000      -0.307      -0.160
temp           1.5244      0.094     16.269      0.000       1.340       1.708
```

```
hum           -0.2495      0.149     -1.677      0.094      -0.542       0.043
windspeed     -1.0399      0.218     -4.760      0.000      -1.469      -0.611
==============================================================================
Omnibus:                      654.553   Durbin-Watson:                   2.039
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           128684.713
Skew:                          -6.061   Prob(JB):                         0.00
Kurtosis:                      79.792   Cond. No.                         131.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[23]:
```
# Initialize logistic regression model
lModel = LinearRegression()
lModel.fit(X = training_set,y = np.log(train_taget))
```

[23]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

[24]:
```
#predicting using linear regression
lmPredictions = lModel.predict(X=test_set)
```

[25]:
```
x=pd.DataFrame(np.exp(lmPredictions))
```

[26]:
```
x.describe()
```

[26]:
```
                0
count    220.000000
mean    4438.291596
std     2120.623071
min     1034.553984
25%     2730.336107
50%     4096.304410
75%     5708.079487
max    11000.217123
```

[27]:
```
lm_errors = abs(np.exp(lmPredictions) - test_target)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(lm_errors), 2), 'degrees.')
```

```
Mean Absolute Error: 899.5 degrees.
```

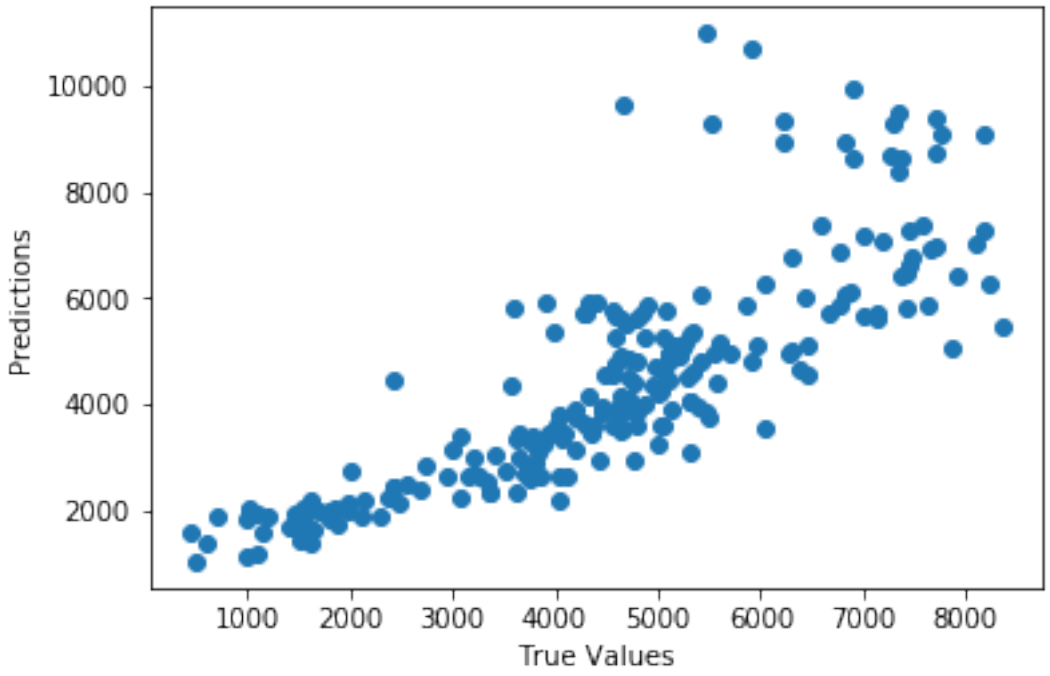
[28]:
```
rmse = sqrt(mean_squared_error(test_target, np.exp(lmPredictions)))
```

[29]:
```
print("RMSE for test set in linear regression is :" , rmse)
```

```
RMSE for test set in linear regression is : 1222.1581373120364
```

[30]:
```
## The line / model
plt.scatter(test_target, np.exp(lmPredictions))
plt.xlabel("True Values")
plt.ylabel("Predictions")
```

[30]:
```
Text(0,0.5,'Predictions')
```



1.6 Model2 Random forest

[31]:
```
rf = RandomForestRegressor(random_state=12345)
```

[32]:
```
rf
```

[32]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=12345, verbose=0,
           warm_start=False)
```

[33]:
```
np.random.seed(12)
start = time.time()
```

```
# selecting best max_depth, maximum features, split criterion and number of␣
 ↪trees
param_dist = {'max_depth': [2,4,6,8,10],
              'bootstrap': [True, False],
              'max_features': ['auto', 'sqrt', 'log2',None],
              "n_estimators" : [100 ,200 ,300 ,400 ,500]
             }
cv_randomForest = RandomizedSearchCV(rf, cv = 10,
                    param_distributions = param_dist,
                    n_iter = 10)

cv_randomForest.fit(training_set, train_taget)
print('Best Parameters using random search: \n',
      cv_randomForest.best_params_)
end = time.time()
print('Time taken in random search: {0: .2f}'.format(end - start))
```

```
Best Parameters using random search:
 {'n_estimators': 300, 'max_features': 'log2', 'max_depth': 8, 'bootstrap':
False}
Time taken in random search:  105.77
```

[34]:
```
# setting parameters

# Set best parameters given by random search # Set be
rf.set_params( max_features = 'log2',
               max_depth =8 ,
               n_estimators = 300
                )
```

[34]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
           max_features='log2', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=12345, verbose=0,
           warm_start=False)
```

[35]:
```
rf.fit(training_set, train_taget)
```

[35]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
           max_features='log2', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=1,
           oob_score=False, random_state=12345, verbose=0,
           warm_start=False)
```

[36]:
```
# Use the forest's predict method on the test data
rfPredictions = rf.predict(test_set)
# Calculate the absolute errors
rf_errors = abs(rfPredictions - test_target)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(rf_errors), 2), 'degrees.')
```

```
Mean Absolute Error: 495.33 degrees.
```

[37]:
```
rmse_rf = sqrt(mean_squared_error(test_target, rfPredictions))
```

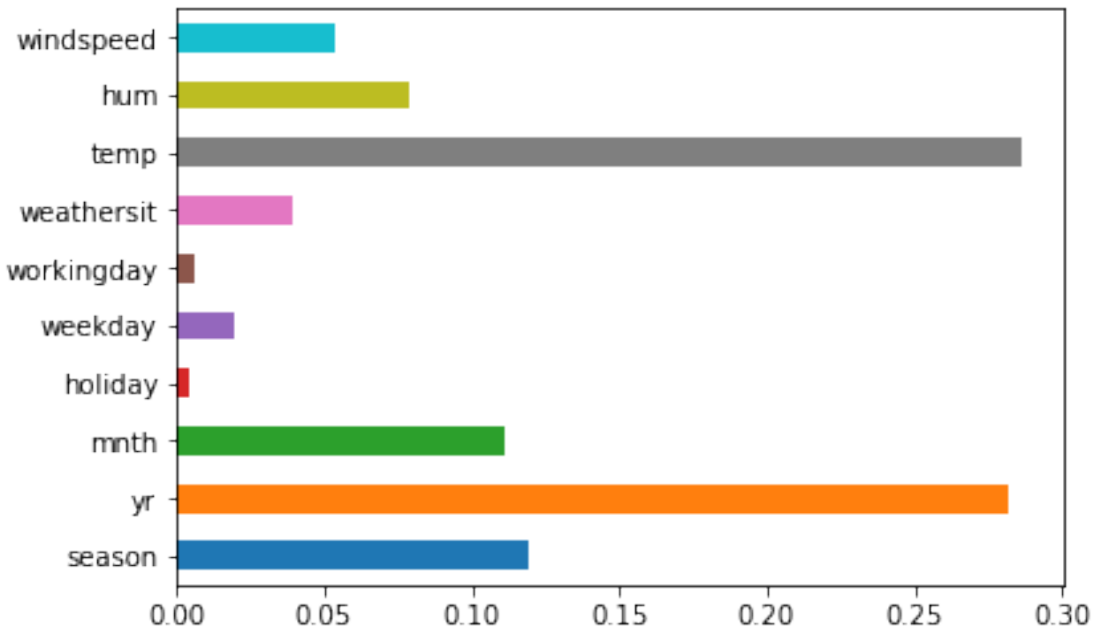
[38]:
```
print("RMSE for test set in random forest regressor  is :" , rmse_rf)
```

```
RMSE for test set in random forest regressor  is : 649.7450571768651
```

1.6.1 Variable importance for random forest

[39]:
```
feature_importance =  pd.Series(rf.feature_importances_, index=training_set.
  ↪columns)
feature_importance.plot(kind='barh')
```

[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6411724e0>
```



[41]:
```
#model input and output
pd.DataFrame(test_set).to_csv('InputLinearRegressionRandomForestPyhon.csv',␣
  ↪index = False)
```

```
pd.DataFrame(np.exp(lmPredictions), columns=['predictions']).
  ↪to_csv('outputLinearRegressionPython.csv')
pd.DataFrame(rfPredictions, columns=['predictions']).
  ↪to_csv('outputRandomForestPython.csv')
```

anime-1

December 11, 2023

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/anime-list/Anime_list.csv
```

[2]:
```
import pandas as pd
import numpy as np
```

[3]:
```
df=pd.read_csv('/kaggle/input/anime-list/Anime_list.csv')
df
```

[3]:

	Anime_name	episode
0	Fullmetal Alchemist: Brotherhood	TV (64 eps)
1	Steins;Gate	TV (24 eps)
2	Gintama°	TV (51 eps)
3	Shingeki no Kyojin Season 3 Part 2	TV (10 eps)
4	Bleach: Sennen Kessen-hen	TV (13 eps)

```
…                                                 …            …
14845  Meitantei Conan Movie 13: Shikkoku no Chaser  Movie (1 eps)
14846                  One Piece Movie 14: Stampede  Movie (1 eps)
14847                         Shoujo Kakumei Utena    TV (39 eps)
14848                      Shoujo Shuumatsu Ryokou    TV (12 eps)
14849                  Bungou Stray Dogs 3rd Season    TV (12 eps)

                  duration            members  Score
0      Apr 2009 - Jul 2010  3,263,142 members   9.09
1      Apr 2011 - Sep 2011  2,505,884 members   9.07
2      Apr 2015 - Mar 2016    614,907 members   9.06
3      Apr 2019 - Jul 2019  2,195,508 members   9.05
4      Oct 2022 - Dec 2022    501,080 members   9.04
…                      …                …    …
14845  Apr 2009 - Apr 2009     58,615 members   8.21
14846  Aug 2019 - Aug 2019    182,431 members   8.21
14847  Apr 1997 - Dec 1997    214,440 members   8.21
14848  Oct 2017 - Dec 2017    333,432 members   8.21
14849  Apr 2019 - Jun 2019    612,395 members   8.21

[14850 rows x 5 columns]
```

[4]:
```
df.head()
```

[4]:
```
                          Anime_name      episode             duration  \
0    Fullmetal Alchemist: Brotherhood  TV (64 eps)  Apr 2009 - Jul 2010
1                        Steins;Gate  TV (24 eps)  Apr 2011 - Sep 2011
2                          Gintama°  TV (51 eps)  Apr 2015 - Mar 2016
3  Shingeki no Kyojin Season 3 Part 2  TV (10 eps)  Apr 2019 - Jul 2019
4           Bleach: Sennen Kessen-hen  TV (13 eps)  Oct 2022 - Dec 2022

             members  Score
0  3,263,142 members   9.09
1  2,505,884 members   9.07
2    614,907 members   9.06
3  2,195,508 members   9.05
4    501,080 members   9.04
```

[5]:
```
df.tail()
```

[5]:
```
                                      Anime_name        episode  \
14845  Meitantei Conan Movie 13: Shikkoku no Chaser  Movie (1 eps)
14846                  One Piece Movie 14: Stampede  Movie (1 eps)
14847                         Shoujo Kakumei Utena    TV (39 eps)
14848                      Shoujo Shuumatsu Ryokou    TV (12 eps)
14849                  Bungou Stray Dogs 3rd Season    TV (12 eps)
```

```
                  duration          members  Score
14845  Apr 2009 - Apr 2009   58,615 members   8.21
14846  Aug 2019 - Aug 2019  182,431 members   8.21
14847  Apr 1997 - Dec 1997  214,440 members   8.21
14848  Oct 2017 - Dec 2017  333,432 members   8.21
14849  Apr 2019 - Jun 2019  612,395 members   8.21
```

[6]:
```
df.shape
```

[6]:
```
(14850, 5)
```

[7]:
```
df.columns
```

[7]:
```
Index(['Anime_name', 'episode', 'duration', 'members', 'Score'], dtype='object')
```

[8]:
```
rows,colums=df.shape
rows
```

[8]:
```
14850
```

[9]:
```
rows,columns=df.shape
columns
```

[9]:
```
5
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14850 entries, 0 to 14849
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_name  14850 non-null  object 
 1   episode     14850 non-null  object 
 2   duration    14850 non-null  object 
 3   members     14850 non-null  object 
 4   Score       14850 non-null  float64
dtypes: float64(1), object(4)
memory usage: 580.2+ KB
```

[11]:
```
df.describe()
```

[11]:
```
              Score
count  14850.000000
mean       8.448015
std        0.173173
min        8.210000
```

```
25%        8.300000
50%        8.410000
75%        8.570000
max        9.090000
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Anime_name    0
episode       0
duration      0
members       0
Score         0
dtype: int64
```

[13]:
```
df.isnull()
```

[13]:
```
       Anime_name  episode  duration  members  Score
0           False    False     False    False  False
1           False    False     False    False  False
2           False    False     False    False  False
3           False    False     False    False  False
4           False    False     False    False  False
…             …        …         …        …      …
14845       False    False     False    False  False
14846       False    False     False    False  False
14847       False    False     False    False  False
14848       False    False     False    False  False
14849       False    False     False    False  False

[14850 rows x 5 columns]
```

[14]:
```
df['Anime_name'].unique()
```

[14]:
```
array(['Fullmetal Alchemist: Brotherhood', 'Steins;Gate', 'Gintama°',
       'Shingeki no Kyojin Season 3 Part 2', 'Bleach: Sennen Kessen-hen',
       'Gintama: The Final', 'Hunter x Hunter (2011)', "Gintama'",
       "Gintama": Enchousen",
       'Kaguya-sama wa Kokurasetai: Ultra Romantic',
       'Ginga Eiyuu Densetsu', 'Fruits Basket: The Final', 'Gintama.',
       'Shingeki no Kyojin: The Final Season - Kanketsu-hen', 'Gintama',
       '3-gatsu no Lion 2nd Season', 'Clannad: After Story',
       'Koe no Katachi', 'Code Geass: Hangyaku no Lelouch R2',
       'Gintama Movie 2: Kanketsu-hen - Yorozuya yo Eien Nare',
       'Jujutsu Kaisen 2nd Season',
       'Gintama.: Shirogane no Tamashii-hen - Kouhan-sen', 'Monster',
       'Owarimonogatari 2nd Season', 'Violet Evergarden Movie',
       'Kimi no Na wa.', 'Kingdom 3rd Season', 'Bocchi the Rock!',
```

'Gintama.: Shirogane no Tamashii-hen',
'Kaguya-sama wa Kokurasetai: First Kiss wa Owaranai',
'The First Slam Dunk', 'Vinland Saga Season 2',
'Mob Psycho 100 II', 'Kizumonogatari III: Reiketsu-hen',
'Shingeki no Kyojin: The Final Season',
'Haikyuu!! Karasuno Koukou vs. Shiratorizawa Gakuen Koukou',
'Hajime no Ippo', 'Kimetsu no Yaiba: Yuukaku-hen',
'Sen to Chihiro no Kamikakushi',
'Shingeki no Kyojin: The Final Season Part 2',
'Monogatari Series: Second Season', 'Vinland Saga', 'Cowboy Bebop',
'Kingdom 4th Season', 'Kusuriya no Hitorigoto',
'Mushishi Zoku Shou 2nd Season', '"Oshi no Ko"',
'Tian Guan Cifu Er',
'Shouwa Genroku Rakugo Shinjuu: Sukeroku Futatabi-hen',
'Ashita no Joe 2', 'Bleach: Sennen Kessen-hen - Ketsubetsu-tan',
'86 Part 2', 'Mob Psycho 100 III', 'One Piece',
'Rurouni Kenshin: Meiji Kenkaku Romantan - Tsuioku-hen',
'Code Geass: Hangyaku no Lelouch', 'Great Teacher Onizuka',
'Mushishi Zoku Shou',
'Mushoku Tensei: Isekai Ittara Honki Dasu Part 2', 'Odd Taxi',
'Shiguang Dailiren', 'Mononoke Hime', 'Violet Evergarden',
'Bungou Stray Dogs 5th Season',
"Fate/stay night Movie: Heaven's Feel - III. Spring Song",
'Hajime no Ippo: New Challenger', 'Howl no Ugoku Shiro',
'Mushishi', 'Made in Abyss',
'Made in Abyss: Retsujitsu no Ougonkyou',
'Shigatsu wa Kimi no Uso', 'Natsume Yuujinchou Shi',
'Kaguya-sama wa Kokurasetai? Tensai-tachi no Renai Zunousen',
'Haikyuu!! Second Season', 'Pluto', 'Tengen Toppa Gurren Lagann',
'Death Note', 'Jujutsu Kaisen',
'Made in Abyss Movie 3: Fukaki Tamashii no Reimei',
'Natsume Yuujinchou Roku', 'Ping Pong the Animation',
'Shingeki no Kyojin Season 3',
'Kimetsu no Yaiba Movie: Mugen Ressha-hen',
'Seishun Buta Yarou wa Yumemiru Shoujo no Yume wo Minai',
'Shin Evangelion Movie: ||', 'Suzumiya Haruhi no Shoushitsu',
'Cyberpunk: Edgerunners', 'Hajime no Ippo: Rising',
'JoJo no Kimyou na Bouken Part 6: Stone Ocean Part 3',
'Mushishi Zoku Shou: Suzu no Shizuku', 'Kenpuu Denki Berserk',
'JoJo no Kimyou na Bouken Part 5: Ougon no Kaze',
'Kizumonogatari II: Nekketsu-hen', 'Natsume Yuujinchou Go',
'Natsume Yuujinchou San', 'Ookami Kodomo no Ame to Yuki',
'Shouwa Genroku Rakugo Shinjuu', 'Spy x Family',
'Tengen Toppa Gurren Lagann Movie 2: Lagann-hen',
'Yojouhan Shinwa Taikei',
'Kage no Jitsuryokusha ni Naritakute! 2nd Season',
'Fate/Zero 2nd Season', 'Fruits Basket 2nd Season',

'Haikyuu!! To the Top Part 2', 'Slam Dunk',
'Kimi no Suizou wo Tabetai',
'Shinseiki Evangelion Movie: Air/Magokoro wo, Kimi ni',
'Shoujo Kageki Revue Starlight Movie', 'Nana', 'Perfect Blue',
'Shingeki no Kyojin', 'Chainsaw Man', 'Zoku Natsume Yuujinchou',
'Steins;Gate 0', 'Yuru Camp Season 2', 'Mushishi: Hihamukage',
'Ousama Ranking', 'Bakuman. 3rd Season',
'Gintama Movie 1: Shinyaku Benizakura-hen', 'Gintama.: Porori-hen',
'Sora yori mo Tooi Basho', 'Kara no Kyoukai Movie 5: Mujun Rasen',
'Koukaku Kidoutai: Stand Alone Complex 2nd GIG',
'Samurai Champloo', 'Shingeki no Kyojin Season 2',
'Hotaru no Haka',
'JoJo no Kimyou na Bouken Part 4: Diamond wa Kudakenai',
'One Punch Man', 'Yakusoku no Neverland', 'Summertime Render',
'Uchuu Kyoudai', 'Ansatsu Kyoushitsu 2nd Season',
"Fate/stay night Movie: Heaven's Feel - II. Lost Butterfly",
'Kimetsu no Yaiba', 'Mob Psycho 100',
'Karakai Jouzu no Takagi-san Movie',
'Mahou Shoujo Madoka Magica Movie 3: Hangyaku no Monogatari',
'Aria the Origination', 'Banana Fish', 'Gintama: The Semi-Final',
'Rainbow: Nisha Rokubou no Shichinin', 'Shiguang Dailiren II',
'Nichijou', 'Yuu Yuu Hakusho', 'Chihayafuru 3',
'Jujutsu Kaisen 0 Movie', 'Bungou Stray Dogs 4th Season',
'Golden Kamuy 3rd Season', 'Mo Dao Zu Shi: Wanjie Pian',
'Owarimonogatari', 'Road of Naruto',
'Steins;Gate Movie: Fuka Ryouiki no Déjà vu', 'Yuru Camp Movie',
'Zoku Owarimonogatari', 'Haikyuu!!',
'JoJo no Kimyou na Bouken Part 3: Stardust Crusaders 2nd Season',
'Kono Subarashii Sekai ni Shukufuku wo! Movie: Kurenai Densetsu',
'Re:Zero kara Hajimeru Isekai Seikatsu 2nd Season Part 2',
'Mushishi Zoku Shou: Odoro no Michi',
'Saenai Heroine no Sodatekata Fine',
'Gintama: Yorinuki Gintama-san on Theater 2D', 'Grand Blue',
'Fruits Basket: Prelude', 'Kono Oto Tomare! Part 2',
'Koukaku Kidoutai: Stand Alone Complex', 'Mononoke',
'Natsume Yuujinchou Movie: Utsusemi ni Musubu',
'Saiki Kusuo no Ψ-nan 2', 'Karakai Jouzu no Takagi-san 3',
'Saiki Kusuo no Ψ-nan', 'Shingeki no Kyojin: Kuinaki Sentaku',
'Violet Evergarden Gaiden: Eien to Jidou Shuki Ningyou',
'Dr. Stone: New World Part 2', 'Hunter x Hunter',
'Kaguya-sama wa Kokurasetai: Tensai-tachi no Renai Zunousen',
'Josee to Tora to Sakana-tachi', 'Major S5', 'Mo Dao Zu Shi',
'Natsume Yuujinchou Roku Specials',
'Sayonara no Asa ni Yakusoku no Hana wo Kazarou',
"Vivy: Fluorite Eye's Song", 'Gintama°: Aizome Kaori-hen',
'Houseki no Kuni', 'Kamisama Hajimemashita: Kako-hen',
'Kara no Kyoukai Movie 7: Satsujin Kousatsu (Go)',

'Kaze ga Tsuyoku Fuiteiru', 'Kimetsu no Yaiba: Mugen Ressha-hen',
'Mushoku Tensei: Isekai Ittara Honki Dasu',
'Tensei shitara Slime Datta Ken 2nd Season', '3-gatsu no Lion',
'Barakamon', 'Chihayafuru 2', 'Cowboy Bebop: Tengoku no Tobira',
'Cross Game', 'Made in Abyss Movie 2: Hourou Suru Tasogare',
'Mahou Shoujo Madoka Magica Movie 2: Eien no Monogatari',
'Mo Dao Zu Shi: Xian Yun Pian', 'Non Non Biyori Nonstop',
'Kaze no Tani no Nausicaä', 'Kizumonogatari I: Tekketsu-hen',
'Mahou Shoujo Madoka Magica', 'Baccano!', 'Haikyuu!! To the Top',
'Yahari Ore no Seishun Love Comedy wa Machigatteiru. Kan',
'Usagi Drop', 'Shinseiki Evangelion', 'Fumetsu no Anata e',
'Gintama: Shiroyasha Koutan', 'Hellsing Ultimate', 'K-On! Movie',
'Bakuman. 2nd Season', 'IDOLiSH7 Third Beat! Part 2',
'Initial D First Stage', 'Suzume no Tojimari', 'Tian Guan Cifu',
'Psycho-Pass', 'Ramayana: The Legend of Prince Rama',
'Re:Zero kara Hajimeru Isekai Seikatsu 2nd Season',
'Kidou Senshi Gundam: The Origin', 'Kiseijuu: Sei no Kakuritsu',
'Natsume Yuujinchou: Itsuka Yuki no Hi ni', 'One Outs',
'Romeo no Aoi Sora', 'Sasaki to Miyano Movie: Sotsugyou-hen',
'Bakemonogatari',
'Tensei shitara Slime Datta Ken 2nd Season Part 2',
'Versailles no Bara',
'Violet Evergarden: Kitto "Ai" wo Shiru Hi ga Kuru no Darou',
'Tian Guan Cifu Special', 'Uchuu Senkan Yamato 2199',
'Kingdom 2nd Season', 'Major S6', 'Natsume Yuujinchou Go Specials',
'Boku no Hero Academia 6th Season',
'Fate/stay night: Unlimited Blade Works 2nd Season', 'Given',
'Hibike! Euphonium 2', 'Hunter x Hunter: Original Video Animation',
'Katanagatari', 'Kemono no Souja Erin',
'Mushoku Tensei II: Isekai Ittara Honki Dasu',
'Natsume Yuujinchou', 'NHK ni Youkoso!',
'Ookami to Koushinryou II', 'Kuroko no Basket 3rd Season',
'Meitantei Conan Movie 06: Baker Street no Bourei',
'Meitantei Conan Movie 26: Kurogane no Submarine',
'Sakamichi no Apollon', 'Wu Liuqi: Xuanwu Guo Pian',
'Ano Hi Mita Hana no Namae wo Bokutachi wa Mada Shiranai.',
'Ashita no Joe', 'Blue Lock', 'Boku dake ga Inai Machi',
'Diamond no Ace: Second Season', 'Evangelion Movie 2: Ha',
'Ginga Eiyuu Densetsu: Die Neue These - Gekitotsu',
'Kage no Jitsuryokusha ni Naritakute!', '86', 'Beck',
'Doukyuusei (Movie)', 'Initial D Final Stage',
'Kimetsu no Yaiba: Katanakaji no Sato-hen',
'Kino no Tabi: The Beautiful World', 'Major: World Series',
'Rurouni Kenshin: Meiji Kenkaku Romantan',
'Steins;Gate: Oukoubakko no Poriomania', 'Spy x Family Part 2',
'Tenki no Ko', 'Yuukoku no Moriarty Part 2', 'Blue Giant',
'Dr. Stone', 'Fate/Zero',

```
       'Ginga Eiyuu Densetsu: Die Neue These - Sakubou',
       'Gintama: Shinyaku Benizakura-hen', 'Hotarubi no Mori e',
       'Kobayashi-san Chi no Maid Dragon S', 'Redline', 'Shinsekai yori',
       'Shirobako', 'Koukaku Kidoutai', 'Nodame Cantabile',
       'Tokyo Godfathers', 'Tsubasa: Tokyo Revelations',
       'World Trigger 3rd Season', 'Yuru Camp ', 'Gin no Saji 2nd Season',
       'Gyakkyou Burai Kaiji: Ultimate Survivor',
       'Diamond no Ace: Act II',
       'Ginga Eiyuu Densetsu: Die Neue These - Seiran 3',
       'Hajime no Ippo: Champion Road',
       'Kono Subarashii Sekai ni Shukufuku wo! 2', 'Naruto: Shippuuden',
       'Planetes', 'Space Dandy 2nd Season', 'Stranger: Mukou Hadan',
       'Tenkuu no Shiro Laputa',
       'Steins;Gate: Kyoukaimenjou no Missing Link - Divide By Zero',
       'Tonari no Totoro', 'Dororo',
       'Gyakkyou Burai Kaiji: Hakairoku-hen',
       'Hunter x Hunter: Greed Island Final',
       'Kuroshitsuji Movie: Book of the Atlantic', 'Sennen Joyuu',
       'Meitantei Conan: Episode One - Chiisaku Natta Meitantei',
       'Non Non Biyori Movie: Vacation',
       'Seishun Buta Yarou wa Bunny Girl Senpai no Yume wo Minai',
       'Bakemono no Ko', "BanG Dream! It's MyGO!!!!!",
       'Boku no Kokoro no Yabai Yatsu', 'Golden Kamuy 2nd Season',
       'Hajime no Ippo: Mashiba vs. Kimura',
       'Danshi Koukousei no Nichijou',
       'Dungeon ni Deai wo Motomeru no wa Machigatteiru Darou ka IV: Fuka Shou -
Yakusai-hen',
       'Fate/strange Fake: Whispers of Dawn', 'Horimiya: Piece',
       'Tanoshii Muumin Ikka', 'Youjo Senki Movie',
       'Kidou Senshi Gundam: Tekketsu no Orphans 2nd Season',
       'Nodame Cantabile Finale', 'Re:Zero kara Hajimeru Isekai Seikatsu',
       'Sasaki to Miyano', 'Kamisama Hajimemashita ',
       'Kono Sekai no Katasumi ni', 'Majo no Takkyuubin', 'Major S3',
       'Mimi wo Sumaseba', 'Ookami to Koushinryou',
       'Fruits Basket 1st Season', 'Great Pretender',
       'IDOLiSH7 Third Beat!', 'Tengoku Daimakyou', 'Trigun',
       'SKET Dance', 'Sono Bisque Doll wa Koi wo Suru',
       'Violet Evergarden: Recollections', 'xxxHOLiC Kei',
       'Yahari Ore no Seishun Love Comedy wa Machigatteiru. Zoku',
       'Kuroko no Basket 2nd Season', 'Magi: The Kingdom of Magic',
       'Mahou Shoujo Madoka Magica Movie 1: Hajimari no Monogatari',
       'Major S1', 'Meitantei Conan Movie 13: Shikkoku no Chaser',
       'One Piece Movie 14: Stampede', 'Shoujo Kakumei Utena',
       'Shoujo Shuumatsu Ryokou', 'Bungou Stray Dogs 3rd Season'],
      dtype=object)
```

[15]:
```
df['members'].unique
```

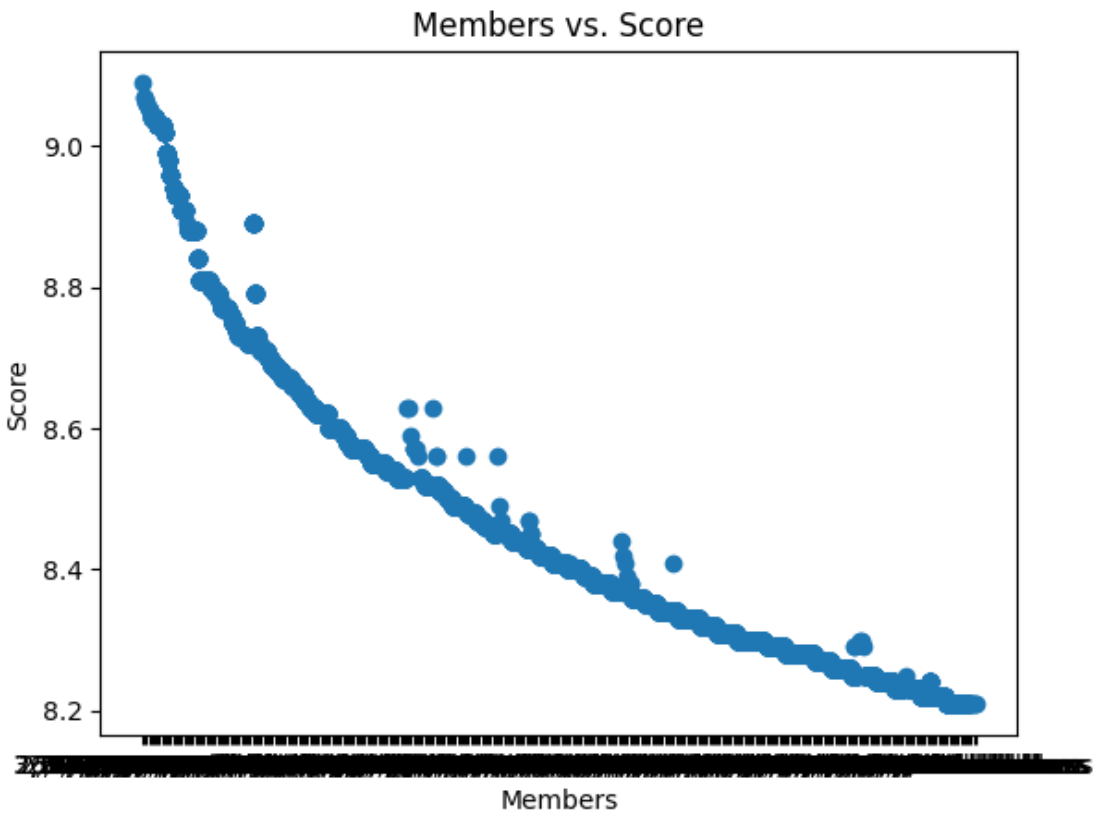
[15]:
```
<bound method Series.unique of 0        3,263,142 members
1        2,505,884 members
2          614,907 members
3        2,195,508 members
4          501,080 members
              …
14845       58,615 members
14846      182,431 members
14847      214,440 members
14848      333,432 members
14849      612,395 members
Name: members, Length: 14850, dtype: object>
```

[16]:
```
df['members'].isnull().sum()
```

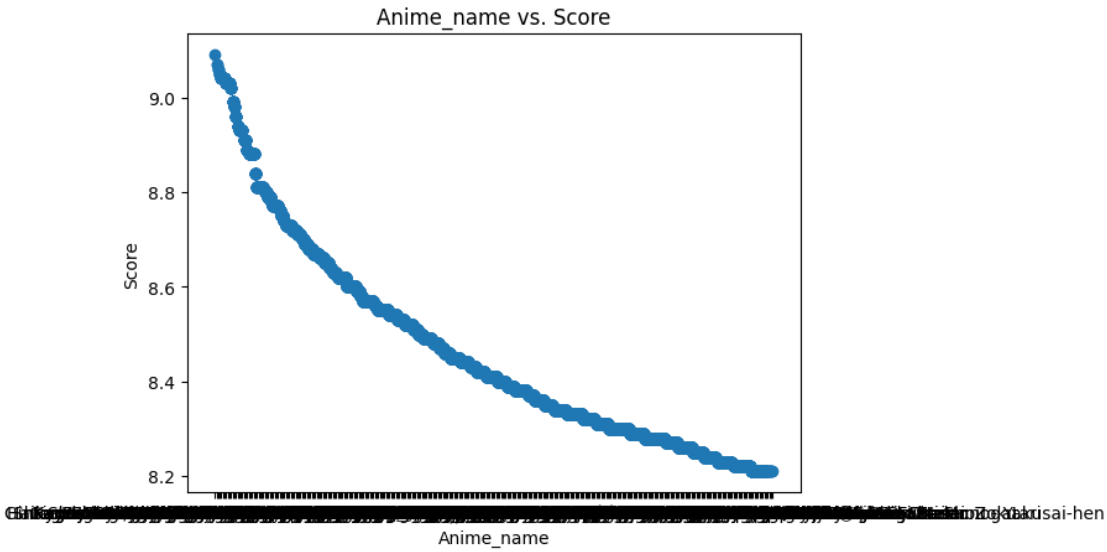
[16]:
```
0
```

[17]:
```
import matplotlib.pyplot as plt
```

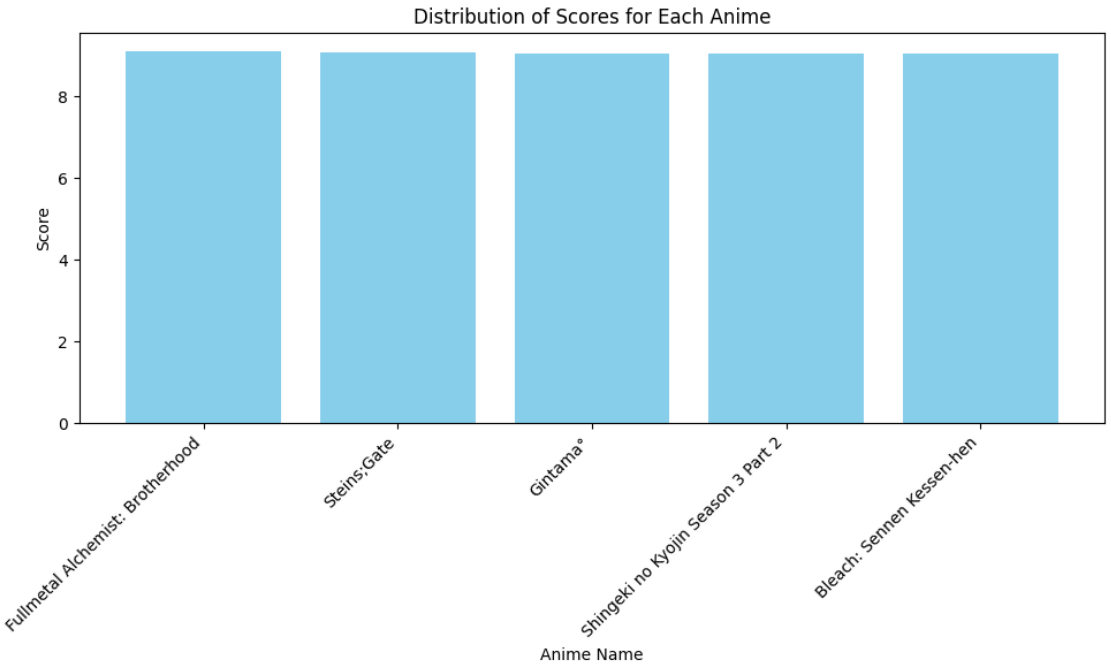
[18]:
```
plt.scatter(df['members'], df['Score'])
plt.title('Members vs. Score')
plt.xlabel('Members')
plt.ylabel('Score')
plt.show()
```



[19]:
```
plt.scatter(df['Anime_name'], df['Score'])
plt.title('Anime_name vs. Score')
plt.xlabel('Anime_name')
plt.ylabel('Score')
plt.show()
```



[20]:
```
import matplotlib.pyplot as plt

# Assuming your dataset is stored in a DataFrame called 'anime_data'
anime_names = df['Anime_name'][:5]  # Selecting the first five anime names
scores = df['Score'][:5]  # Corresponding scores

# Plotting the histogram
plt.figure(figsize=(10, 6))
plt.bar(anime_names, scores, color='skyblue')
plt.xlabel('Anime Name')
plt.ylabel('Score')
plt.title('Distribution of Scores for Each Anime')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better␣
 ↪visibility
plt.tight_layout()

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



How many rows and columns are there in the dataset?**

[49]:
```
row,columns=df.shape
rows
```

[49]: 14850

[50]:
```
row,columns=df.shape
columns
```

[50]: 5

Question 2: What is the average score of the anime in the dataset?

[22]:
```
average_score = df['Score'].mean()
print(f"The average score of the anime is: {average_score:.2f}")
```

```
The average score of the anime is: 8.45
```

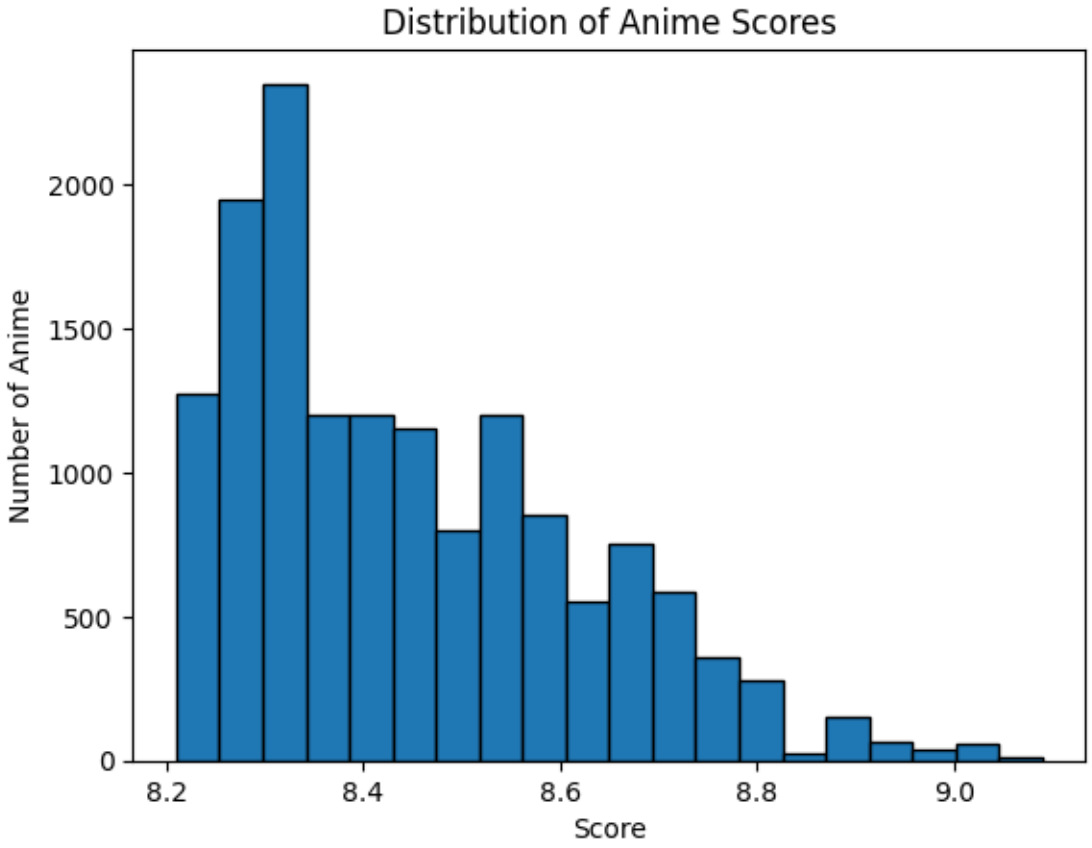
Question 3: What is the maximum and minimum score among the anime in the dataset?

[26]:
```
max_score = df['Score'].max()
min_score = df['Score'].min()
print(f"The maximum score is: {max_score}\nThe minimum score is: {min_score}")
```

```
The maximum score is: 9.09
The minimum score is: 8.21
```

Question 4: What is the distribution of anime scores?

[31]:
```
import matplotlib.pyplot as plt

# Plot a histogram of anime scores
plt.hist(df['Score'], bins=20, edgecolor='black')
plt.xlabel('Score')
plt.ylabel('Number of Anime')
plt.title('Distribution of Anime Scores')
plt.show()
```



Question 5: How many anime have a score higher than 9?

[45]:
```
# Count anime with a score higher than 9
num_high_score_anime = len(df[df['Score'] > 9])
print(f"There are {num_high_score_anime} anime with a score higher than 9.")
```

```
There are 66 anime with a score higher than 9.
```

[ ]:

rain-prediction

December 11, 2023

[271]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv
```

[272]:
```
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('dark_background')
```

[273]:
```
df = pd.read_csv('/kaggle/input/weather-dataset-rattle-package/weatherAUS.csv')
rain
```

[273]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	\
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	

```
1       2008-12-02   Albury      7.4     25.1       0.0          NaN
2       2008-12-03   Albury     12.9     25.7       0.0          NaN
3       2008-12-04   Albury      9.2     28.0       0.0          NaN
4       2008-12-05   Albury     17.5     32.3       1.0          NaN
…            …       …       …       …        …          …
145455  2017-06-21    Uluru      2.8     23.4       0.0          NaN
145456  2017-06-22    Uluru      3.6     25.3       0.0          NaN
145457  2017-06-23    Uluru      5.4     26.9       0.0          NaN
145458  2017-06-24    Uluru      7.8     27.0       0.0          NaN
145459  2017-06-25    Uluru     14.9      NaN       0.0          NaN

        Sunshine WindGustDir  WindGustSpeed WindDir9am  … Humidity9am  \
0            NaN           W           44.0          W  …        71.0
1            NaN         WNW           44.0        NNW  …        44.0
2            NaN         WSW           46.0          W  …        38.0
3            NaN          NE           24.0         SE  …        45.0
4            NaN           W           41.0        ENE  …        82.0
…            …          …             …         …  …          …
145455       NaN           E           31.0         SE  …        51.0
145456       NaN         NNW           22.0         SE  …        56.0
145457       NaN           N           37.0         SE  …        53.0
145458       NaN          SE           28.0        SSE  …        51.0
145459       NaN         NaN            NaN        ESE  …        62.0

        Humidity3pm  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  \
0              22.0       1007.7       1007.1       8.0       NaN     16.9
1              25.0       1010.6       1007.8       NaN       NaN     17.2
2              30.0       1007.6       1008.7       NaN       2.0     21.0
3              16.0       1017.6       1012.8       NaN       NaN     18.1
4              33.0       1010.8       1006.0       7.0       8.0     17.8
…             …          …          …       …       …      …
145455         24.0       1024.6       1020.3       NaN       NaN     10.1
145456         21.0       1023.5       1019.1       NaN       NaN     10.9
145457         24.0       1021.0       1016.8       NaN       NaN     12.5
145458         24.0       1019.4       1016.5       3.0       2.0     15.1
145459         36.0       1020.2       1017.9       8.0       8.0     15.0

        Temp3pm  RainToday  RainTomorrow
0          21.8         No            No
1          24.3         No            No
2          23.2         No            No
3          26.5         No            No
4          29.7         No            No
…         …        …           …
145455     22.4         No            No
145456     24.5         No            No
145457     26.1         No            No
```

```
145458     26.0         No            No
145459     20.9         No           NaN

[145460 rows x 23 columns]
```

[274]:
```
df.head()
```

[274]:
```
         Date Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  \
0  2008-12-01   Albury     13.4     22.9       0.6          NaN       NaN
1  2008-12-02   Albury      7.4     25.1       0.0          NaN       NaN
2  2008-12-03   Albury     12.9     25.7       0.0          NaN       NaN
3  2008-12-04   Albury      9.2     28.0       0.0          NaN       NaN
4  2008-12-05   Albury     17.5     32.3       1.0          NaN       NaN

  WindGustDir  WindGustSpeed WindDir9am  … Humidity9am  Humidity3pm  \
0           W           44.0          W  …        71.0         22.0
1         WNW           44.0        NNW  …        44.0         25.0
2         WSW           46.0          W  …        38.0         30.0
3          NE           24.0         SE  …        45.0         16.0
4           W           41.0        ENE  …        82.0         33.0

   Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  \
0       1007.7       1007.1       8.0       NaN     16.9     21.8         No
1       1010.6       1007.8       NaN       NaN     17.2     24.3         No
2       1007.6       1008.7       NaN       2.0     21.0     23.2         No
3       1017.6       1012.8       NaN       NaN     18.1     26.5         No
4       1010.8       1006.0       7.0       8.0     17.8     29.7         No

   RainTomorrow
0            No
1            No
2            No
3            No
4            No

[5 rows x 23 columns]
```

[275]:
```
df.describe()
```

[275]:
```
             MinTemp        MaxTemp       Rainfall   Evaporation  \
count  143975.000000  144199.000000  142199.000000  82670.000000
mean       12.194034      23.221348       2.360918      5.468232
std         6.398495       7.119049       8.478060      4.193704
min        -8.500000      -4.800000       0.000000      0.000000
25%         7.600000      17.900000       0.000000      2.600000
50%        12.000000      22.600000       0.000000      4.800000
75%        16.900000      28.200000       0.800000      7.400000
```

```
max        33.900000      48.100000     371.000000    145.000000

           Sunshine  WindGustSpeed   WindSpeed9am   WindSpeed3pm  \
count  75625.000000  135197.000000  143693.000000  142398.000000
mean       7.611178      40.035230      14.043426      18.662657
std        3.785483      13.607062       8.915375       8.809800
min        0.000000       6.000000       0.000000       0.000000
25%        4.800000      31.000000       7.000000      13.000000
50%        8.400000      39.000000      13.000000      19.000000
75%       10.600000      48.000000      19.000000      24.000000
max       14.500000     135.000000     130.000000      87.000000

         Humidity9am    Humidity3pm   Pressure9am    Pressure3pm  \
count  142806.000000  140953.000000  130395.00000  130432.000000
mean       68.880831      51.539116    1017.64994    1015.255889
std        19.029164      20.795902       7.10653       7.037414
min         0.000000       0.000000     980.50000     977.100000
25%        57.000000      37.000000    1012.90000    1010.400000
50%        70.000000      52.000000    1017.60000    1015.200000
75%        83.000000      66.000000    1022.40000    1020.000000
max       100.000000     100.000000    1041.00000    1039.600000

           Cloud9am      Cloud3pm        Temp9am       Temp3pm
count  89572.000000  86102.000000  143693.000000  141851.00000
mean       4.447461      4.509930      16.990631      21.68339
std        2.887159      2.720357       6.488753       6.93665
min        0.000000      0.000000      -7.200000      -5.40000
25%        1.000000      2.000000      12.300000      16.60000
50%        5.000000      5.000000      16.700000      21.10000
75%        7.000000      7.000000      21.600000      26.40000
max        9.000000      9.000000      40.200000      46.70000
```

[276]:
```
df.shape
```

[276]:
```
(145460, 23)
```

[277]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
```

```
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

[278]:
```
df.nunique()
```

[278]:
```
Date             3436
Location           49
MinTemp           389
MaxTemp           505
Rainfall          681
Evaporation       358
Sunshine          145
WindGustDir        16
WindGustSpeed      67
WindDir9am         16
WindDir3pm         16
WindSpeed9am       43
WindSpeed3pm       44
Humidity9am       101
Humidity3pm       101
Pressure9am       546
Pressure3pm       549
Cloud9am           10
Cloud3pm           10
Temp9am           441
Temp3pm           502
RainToday           2
RainTomorrow        2
```

```
dtype: int64
```

[279]:
```
df.isnull().sum()
```

[279]:
```
Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
```

[280]:
```
df = df.drop(["Evaporation","Sunshine","Cloud9am","Temp3pm","Location","Date"],␣
↪axis=1)
df.head()
```

[280]:

	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am
0	13.4	22.9	0.6	W	44.0	W
1	7.4	25.1	0.0	WNW	44.0	NNW
2	12.9	25.7	0.0	WSW	46.0	W
3	9.2	28.0	0.0	NE	24.0	SE
4	17.5	32.3	1.0	W	41.0	ENE

	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
0	WNW	20.0	24.0	71.0	22.0
1	WSW	4.0	22.0	44.0	25.0
2	WSW	19.0	26.0	38.0	30.0
3	E	11.0	9.0	45.0	16.0
4	NW	7.0	20.0	82.0	33.0

```
   Pressure9am  Pressure3pm  Cloud3pm  Temp9am RainToday RainTomorrow
0       1007.7       1007.1       NaN     16.9        No           No
1       1010.6       1007.8       NaN     17.2        No           No
2       1007.6       1008.7       2.0     21.0        No           No
3       1017.6       1012.8       NaN     18.1        No           No
4       1010.8       1006.0       8.0     17.8        No           No
```

[281]:
```
df = df.dropna(axis = 0)
df.shape
```

[281]:
```
(74279, 17)
```

[282]:
```
df.isnull().sum()
```

[282]:
```
MinTemp          0
MaxTemp          0
Rainfall         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud3pm         0
Temp9am          0
RainToday        0
RainTomorrow     0
dtype: int64
```

[283]:
```
df.columns
```

[283]:
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustDir', 'WindGustSpeed',
       'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm',
       'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud3pm',
       'Temp9am', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

[284]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['WindGustDir'] = le.fit_transform(df['WindGustDir'])
df['WindDir9am'] = le.fit_transform(df['WindDir9am'])
df['WindDir3pm'] = le.fit_transform(df['WindDir3pm'])
df['RainToday'] = le.fit_transform(df['RainToday'])
```

```
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])
```

[285]:
```
df.head()
```

[285]:
```
     MinTemp  MaxTemp  Rainfall  WindGustDir  WindGustSpeed  WindDir9am  \
2       12.9     25.7       0.0           15           46.0          13
4       17.5     32.3       1.0           13           41.0           1
11      15.9     21.7       2.2            5           31.0           4
12      15.9     18.6      15.6           13           61.0           6
13      12.6     21.0       3.6           12           44.0          13

     WindDir3pm  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
2            15          19.0          26.0         38.0         30.0
4             7           7.0          20.0         82.0         33.0
11            1          15.0          13.0         89.0         91.0
12            6          28.0          28.0         76.0         93.0
13           11          24.0          20.0         65.0         43.0

     Pressure9am  Pressure3pm  Cloud3pm  Temp9am  RainToday  RainTomorrow
2         1007.6       1008.7       2.0     21.0          0             0
4         1010.8       1006.0       8.0     17.8          0             0
11        1010.5       1004.2       8.0     15.9          1             1
12         994.3        993.0       8.0     17.4          1             1
13        1001.2       1001.8       7.0     15.8          1             0
```

[286]:
```
x = df.drop(['RainTomorrow'], axis = 1)
y = df['RainTomorrow']
```

[287]:
```
x.head()
```

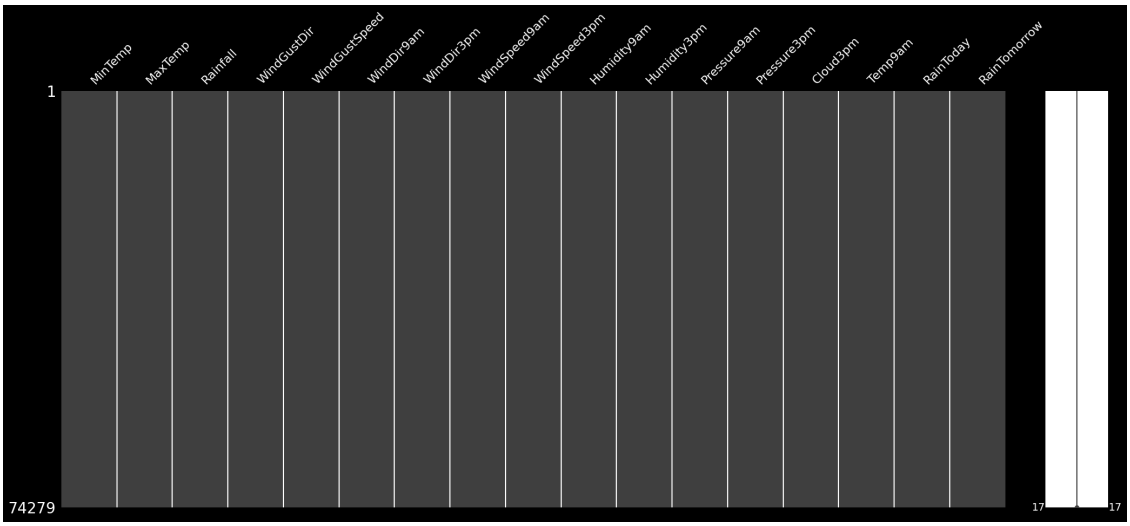
[287]:
```
     MinTemp  MaxTemp  Rainfall  WindGustDir  WindGustSpeed  WindDir9am  \
2       12.9     25.7       0.0           15           46.0          13
4       17.5     32.3       1.0           13           41.0           1
11      15.9     21.7       2.2            5           31.0           4
12      15.9     18.6      15.6           13           61.0           6
13      12.6     21.0       3.6           12           44.0          13

     WindDir3pm  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
2            15          19.0          26.0         38.0         30.0
4             7           7.0          20.0         82.0         33.0
11            1          15.0          13.0         89.0         91.0
12            6          28.0          28.0         76.0         93.0
13           11          24.0          20.0         65.0         43.0

     Pressure9am  Pressure3pm  Cloud3pm  Temp9am  RainToday
2         1007.6       1008.7       2.0     21.0          0
4         1010.8       1006.0       8.0     17.8          0
```

11	1010.5	1004.2	8.0	15.9	1
12	994.3	993.0	8.0	17.4	1
13	1001.2	1001.8	7.0	15.8	1

visualizing the missing values

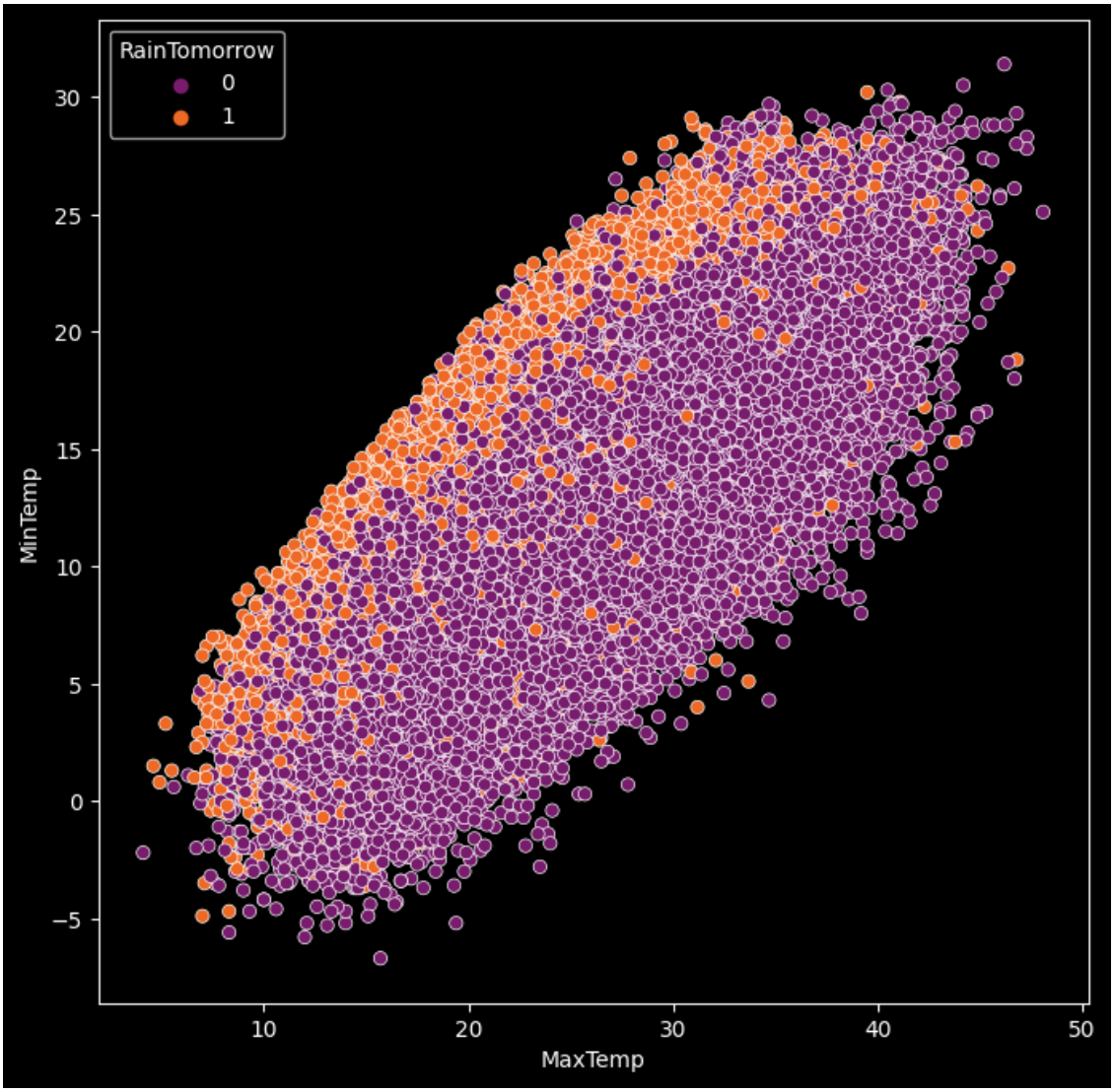
[288]:
```
msno.matrix(df)
```

[288]: <Axes: >



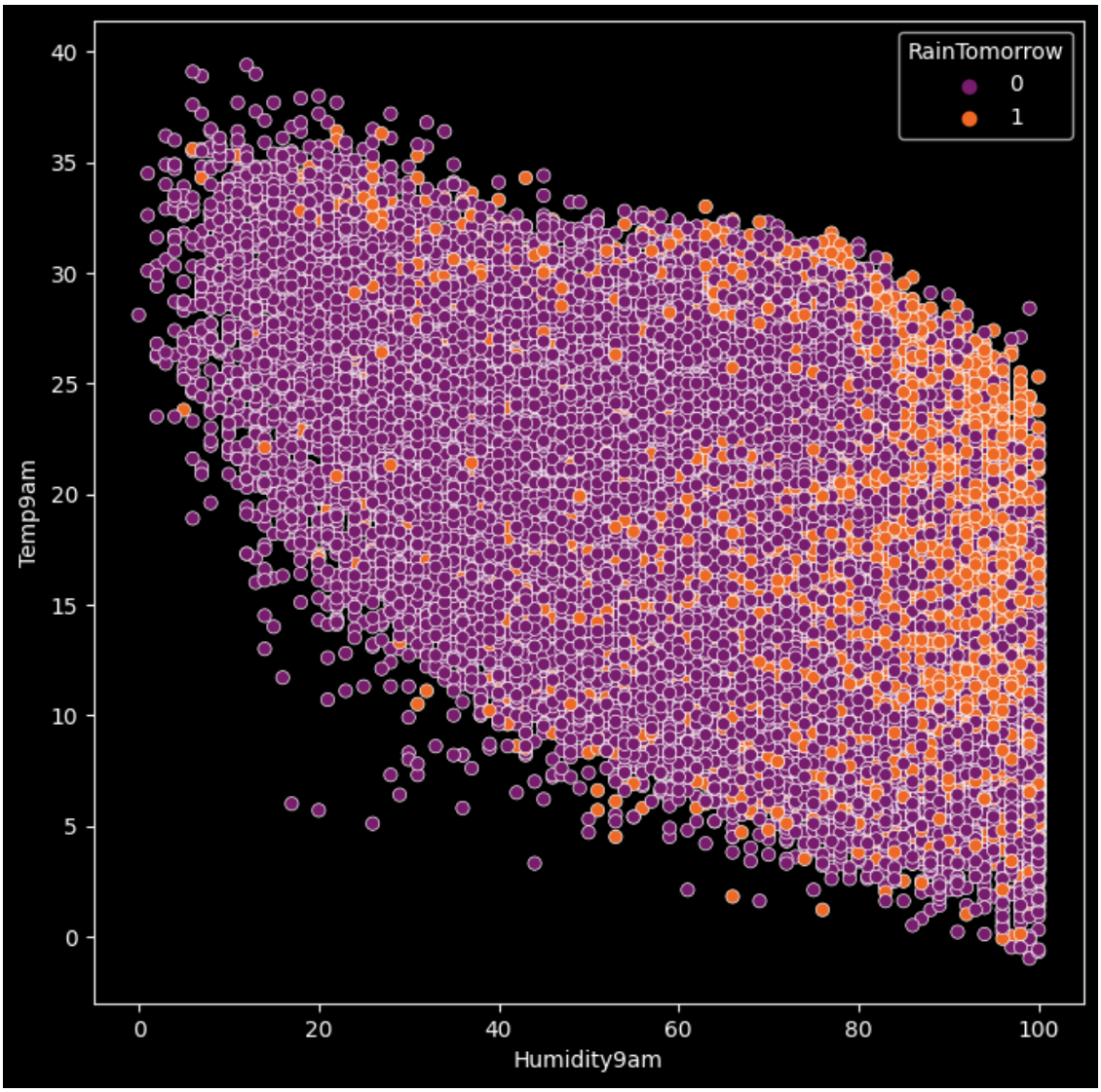
[289]:
```
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'MaxTemp', y = 'MinTemp', hue = 'RainTomorrow', palette =␣
↪'inferno',data = df)
```

[289]: <Axes: xlabel='MaxTemp', ylabel='MinTemp'>



[290]:
```
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'Humidity9am', y = 'Temp9am', hue = 'RainTomorrow', palette␣
↪= 'inferno',data = df)
```

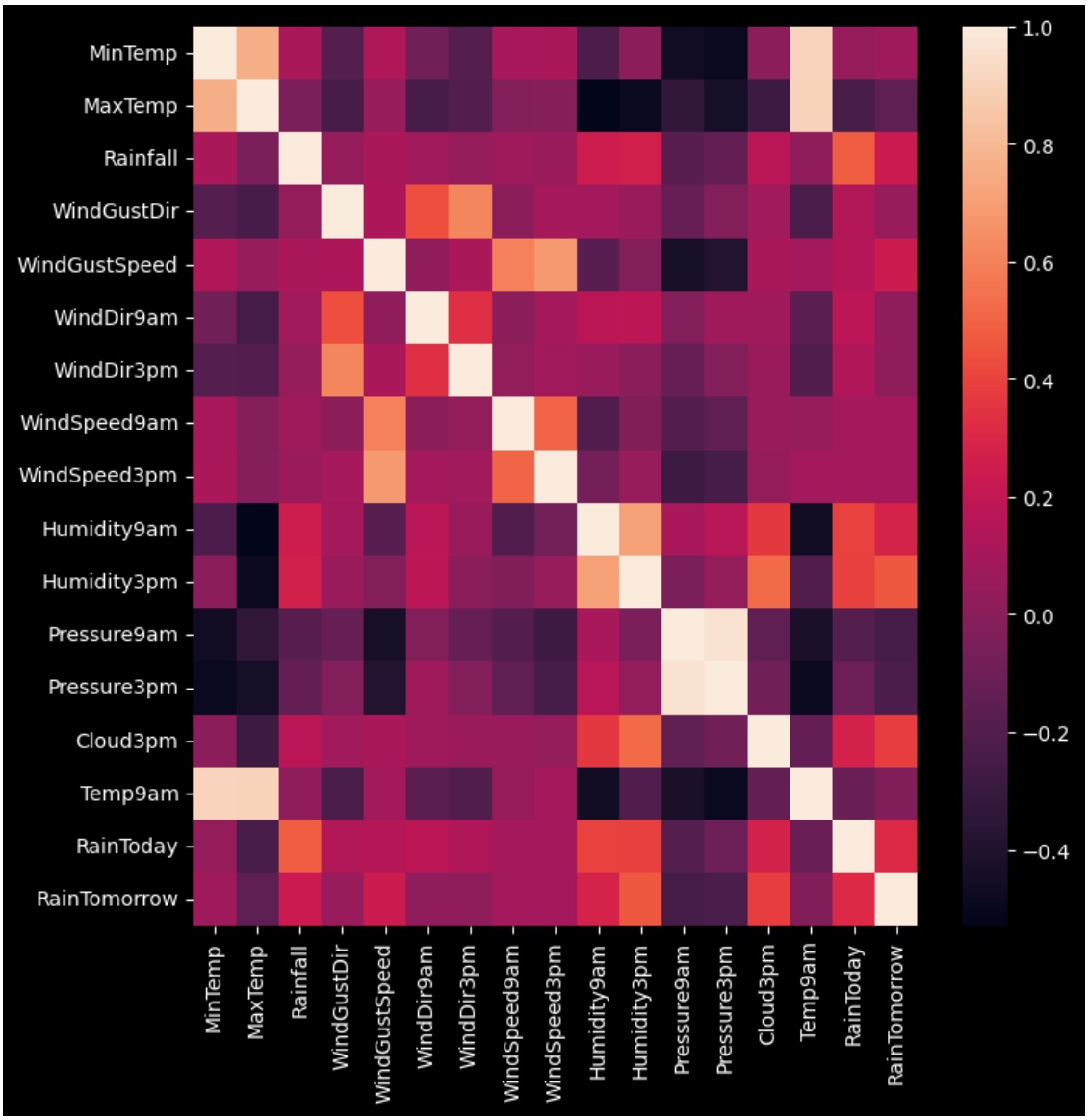
[290]: <Axes: xlabel='Humidity9am', ylabel='Temp9am'>



Heatmap

[291]:
```
plt.figure(figsize = (8,8))
sns.heatmap(df.corr())
```

[291]: <Axes: >



changing yes and no to 1 and 0 in some columns

[292]:
```
df['RainTomorrow'] = df['RainTomorrow'].map({'Yes':1, 'No':0})
df['RainToday'] = df['RainToday'].map({'Yes':1, 'No':0})
print(df.RainToday)
print(df.RainTomorrow)
```

```
2         NaN
4         NaN
11        NaN
12        NaN
13        NaN
           ..
```

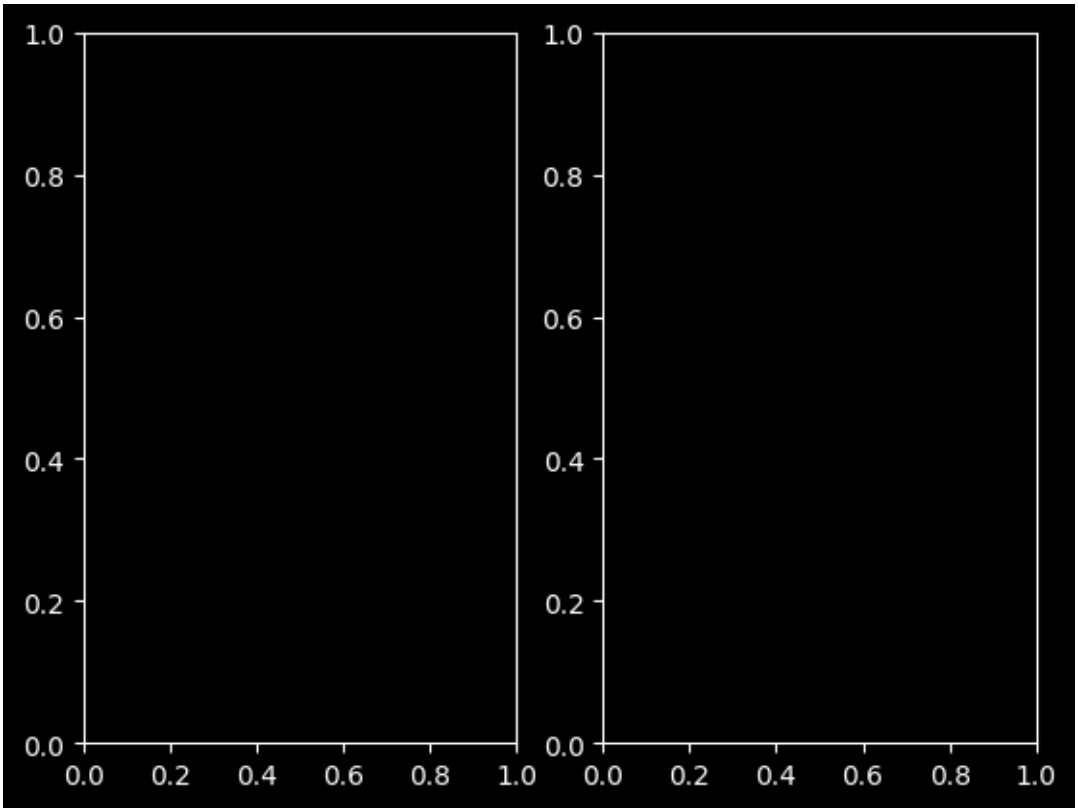
```
145428   NaN
145432   NaN
145433   NaN
145452   NaN
145458   NaN
Name: RainToday, Length: 74279, dtype: float64
2        NaN
4        NaN
11       NaN
12       NaN
13       NaN
          ..
145428   NaN
145432   NaN
145433   NaN
145452   NaN
145458   NaN
Name: RainTomorrow, Length: 74279, dtype: float64
```

count of rain today and tomorrow

[293]:
```
fig, ax =plt.subplots(1,2)
print(df.RainToday.value_counts())
print(df.RainTomorrow.value_counts())

plt.figure(figsize=(20,20))
sns.scatterplot(data=df,x='RainToday', ax=ax[0])
sns.scatterplot(data=df,x='RainTomorrow', ax=ax[1])
```

```
Series([], Name: count, dtype: int64)
Series([], Name: count, dtype: int64)
```

[293]:
```
<Axes: >
```




```
<Figure size 2000x2000 with 0 Axes>
```

[ ]:

[294]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
```

[295]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score
```

Logistic regression

[296]:
```
from sklearn.linear_model import LogisticRegression
lr =  LogisticRegression()
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[10814   630]
 [ 1630  1782]]
              precision    recall  f1-score   support
```

```
              0       0.87      0.94      0.91     11444
              1       0.74      0.52      0.61      3412

       accuracy                           0.85     14856
      macro avg       0.80      0.73      0.76     14856
   weighted avg       0.84      0.85      0.84     14856

0.8478729133010231

/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Decision tree

[297]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
predictions = dt.predict(x_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[9808 1636]
 [1505 1907]]
              precision    recall  f1-score   support

           0       0.87      0.86      0.86     11444
           1       0.54      0.56      0.55      3412

    accuracy                           0.79     14856
   macro avg       0.70      0.71      0.71     14856
weighted avg       0.79      0.79      0.79     14856

0.7885702746365105
```

Random Forest Classifier

[298]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
predictions = rf.predict(x_test)
```

```
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

```
[[10861   583]
 [ 1558  1854]]
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     11444
           1       0.76      0.54      0.63      3412

    accuracy                           0.86     14856
   macro avg       0.82      0.75      0.77     14856
weighted avg       0.85      0.86      0.85     14856

0.8558831448572967
```

XGBoost Classifier

[299]:
```
import xgboost as xgb
xgb = xgb.XGBClassifier()
xgb.fit(x_train,y_train)
pred = xgb.predict(x_test)
print('acc',accuracy_score(y_test, pred))
print('f1',classification_report(y_test, pred))
print('matrix',confusion_matrix(y_test,pred))
```

```
acc 0.8544022617124394
f1               precision    recall  f1-score   support

           0       0.88      0.94      0.91     11444
           1       0.74      0.57      0.64      3412

    accuracy                           0.85     14856
   macro avg       0.81      0.75      0.78     14856
weighted avg       0.85      0.85      0.85     14856

matrix [[10750   694]
 [ 1469  1943]]
```

car-prediction

December 11, 2023

[ ]:
```
Topic.  Car prediction
```

[2]:
```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('"C:/Users/SHADAB/OneDrive/Documents/car␣
 ↪data (1).csv"'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
        import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics
car_dataset = pd.read_csv('C:/Users/SHADAB/OneDrive/Documents/car data (1).csv')
car_dataset.head()
```

[2]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	\
0	ritz	2014	3.35	5.59	27000	Petrol	
1	sx4	2013	4.75	9.54	43000	Diesel	
2	ciaz	2017	7.25	9.85	6900	Petrol	
3	wagon r	2011	2.85	4.15	5200	Petrol	
4	swift	2014	4.60	6.87	42450	Diesel	

	Selling_type	Transmission	Owner
0	Dealer	Manual	0
1	Dealer	Manual	0
2	Dealer	Manual	0
3	Dealer	Manual	0
4	Dealer	Manual	0

question 1–> “How many types of fuel are there for cars—petrol, diesel, and CNG?”

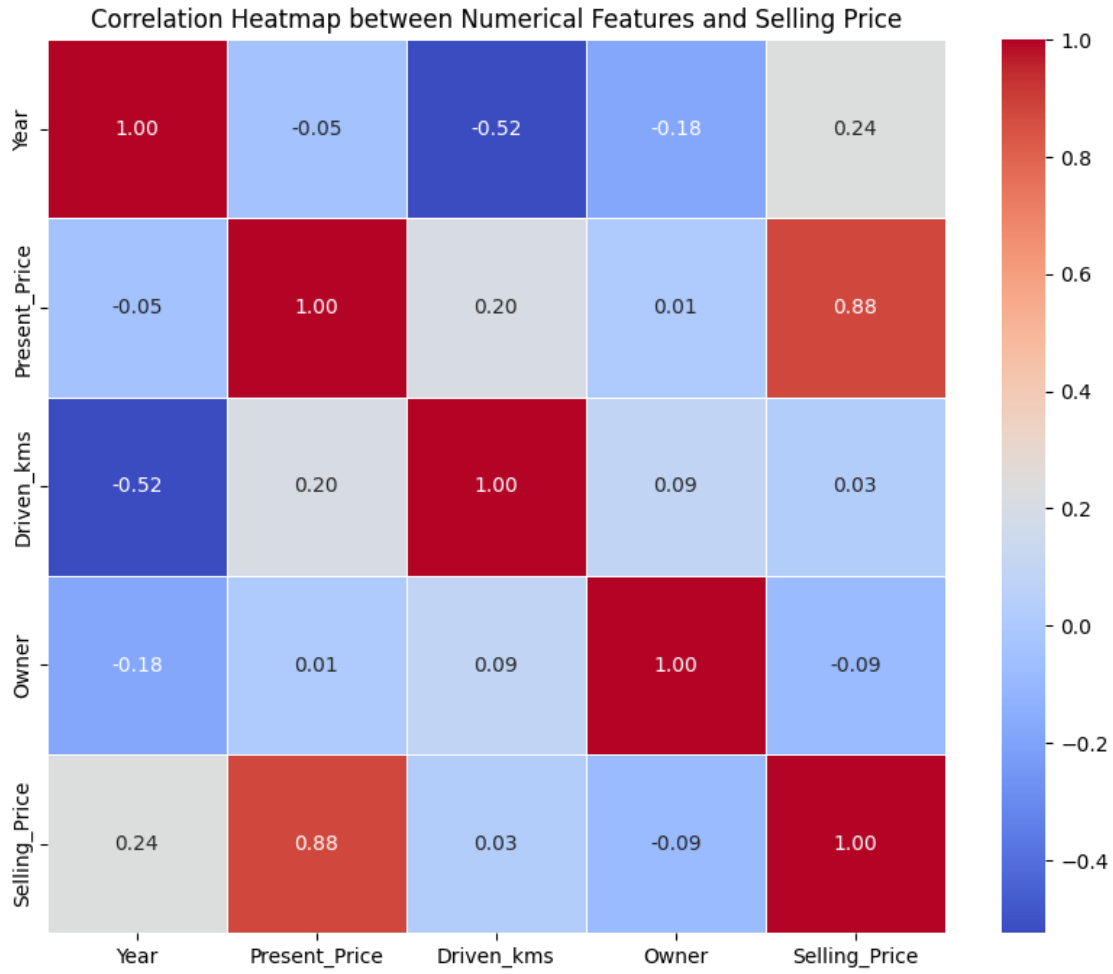
[15]:
```
print(car_dataset.Fuel_Type.value_counts())
```

```
Fuel_Type
```

```
Petrol    239
Diesel     60
CNG         2
Name: count, dtype: int64
```

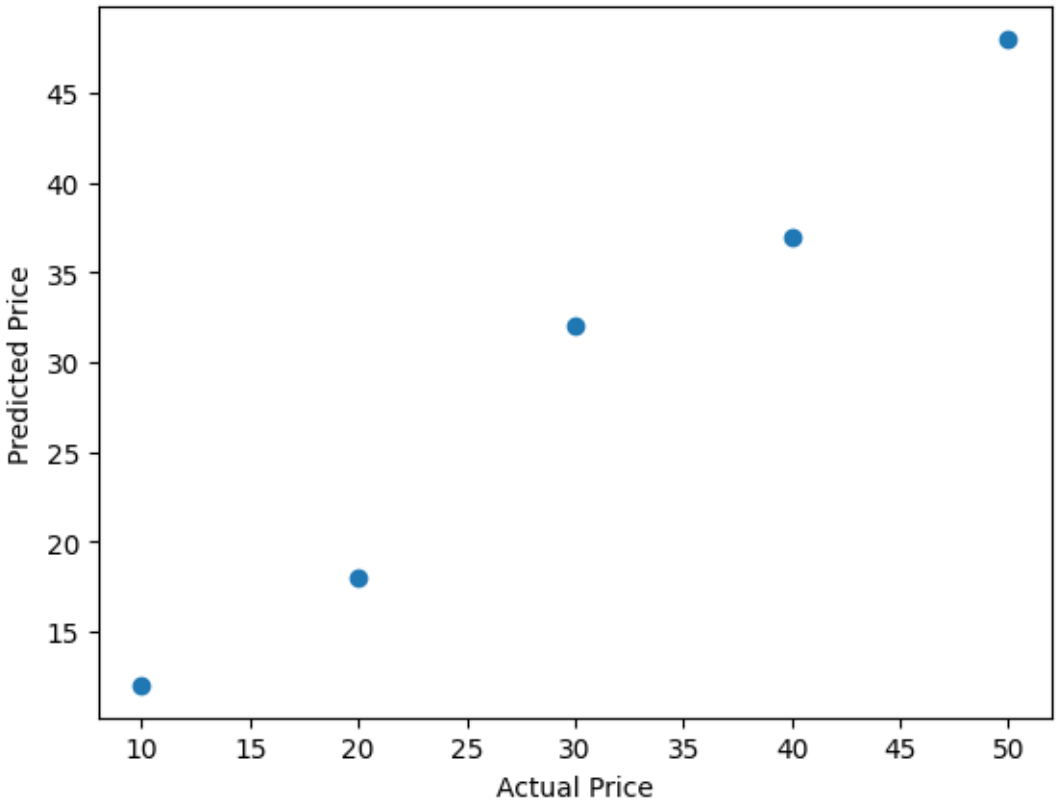
[ ]:
```
Question--> 2 show there matrix any correlation between the numerical features␣
 ↪and the selling price?
```

[21]:
```
def numerical_features_correlation(dataset):
    numerical_features = dataset.select_dtypes(include=['int64', 'float64']).
 ↪columns.tolist()
    numerical_features.remove('Selling_Price')  # Remove the target variable

    # Correlation matrix
    corr_matrix = dataset[numerical_features + ['Selling_Price']].corr()

    # Plotting the heatmap
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
    plt.title("Correlation Heatmap between Numerical Features and Selling␣
 ↪Price")
    plt.show()

# Assuming 'car_dataset' is the variable containing your dataset
numerical_features_correlation(car_dataset)
```



[ ]:
```
Question 3--> How closely do the predicted prices align with the actual prices?
```

[4]:
```
Y_train = [10, 20, 30, 40, 50]
training_data_prediction = [12, 18, 32, 37, 48]

plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.show()
```



[16]:
```
X = car_dataset.drop(['Car_Name','Selling_Price'],axis=1)
Y = car_dataset['Selling_Price']
```

[18]:
```
print(X)
```

```
     Year  Present_Price  Driven_kms Fuel_Type Selling_type Transmission  \
0    2014           5.59       27000    Petrol       Dealer       Manual
1    2013           9.54       43000    Diesel       Dealer       Manual
2    2017           9.85        6900    Petrol       Dealer       Manual
3    2011           4.15        5200    Petrol       Dealer       Manual
4    2014           6.87       42450    Diesel       Dealer       Manual
..    …              …          …        …           …            …
296  2016          11.60       33988    Diesel       Dealer       Manual
297  2015           5.90       60000    Petrol       Dealer       Manual
298  2009          11.00       87934    Petrol       Dealer       Manual
299  2017          12.50        9000    Diesel       Dealer       Manual
300  2016           5.90        5464    Petrol       Dealer       Manual

     Owner
0        0
1        0
```

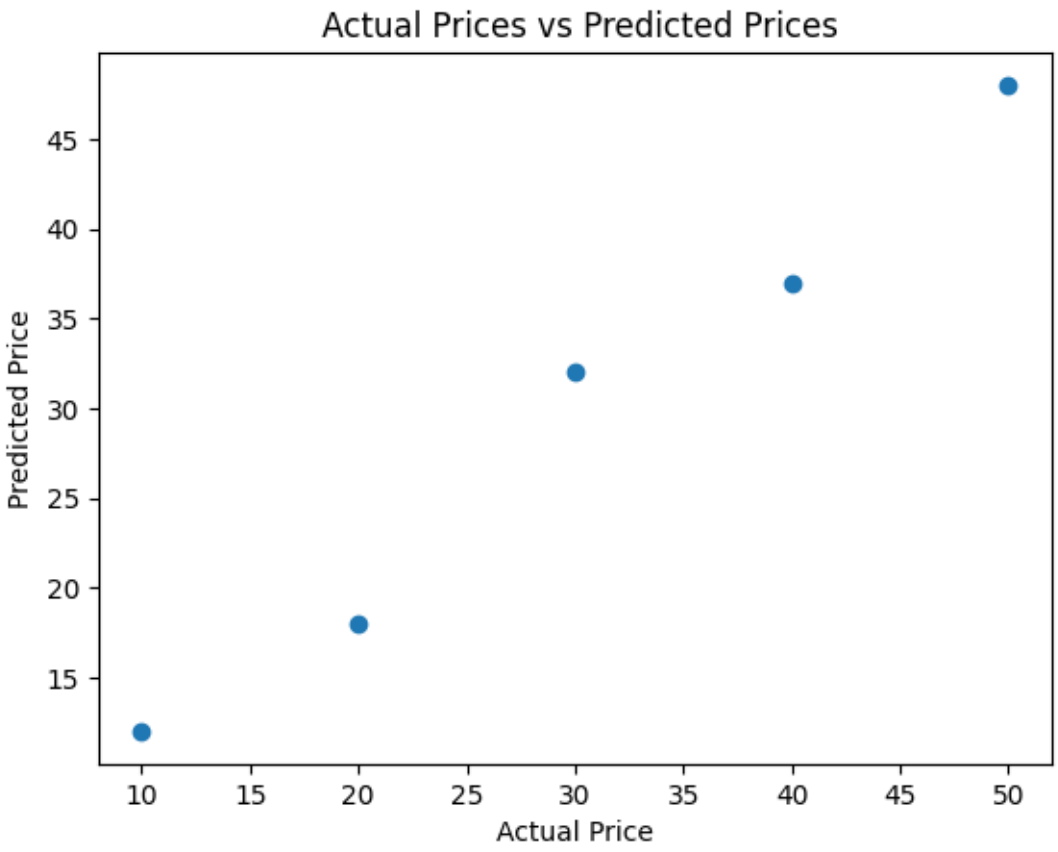
```
2        0
3        0
4        0
..     …
296      0
297      0
298      0
299      0
300      0

[301 rows x 7 columns]
```

[19]:
```
print(Y)
```

```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       …
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

[ ]:
```
Question 5-->   5- How well do the predicated prices align with actual prices␣
 ↪in the scatter prices
```

[20]:
```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```


Actual Prices vs Predicted Prices
Predicted Price
Actual Price
10
15
20
25
30
35
40
45
50
15
20
25
30
35
40
45

crime-against-women

December 11, 2023

1 Crimes against women in India

1.0.1 This project contains the data record of all types of crime against women in india in the span of year 2001-2014. Main aim of working on this project is to reflect the situation of women in our society and raise concern about this matter.

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1.1 Importing Data

Reading the csv file.

[4]:
```
crimes_df = pd.read_csv('/content/crimes_against_women_2001-2014.csv')
```

[5]:
```
crimes_df
```

[5]:
```
       Unnamed: 0        STATE/UT           DISTRICT  Year  Rape  \
0               0  ANDHRA PRADESH           ADILABAD  2001    50
1               1  ANDHRA PRADESH          ANANTAPUR  2001    23
2               2  ANDHRA PRADESH           CHITTOOR  2001    27
3               3  ANDHRA PRADESH           CUDDAPAH  2001    20
4               4  ANDHRA PRADESH      EAST GODAVARI  2001    23
…             …               …                  …   …   …
10672         832     Lakshadweep        Lakshadweep  2014     1
10673         833     Lakshadweep  Total District(s)  2014     1
10674         834      Puducherry           Karaikal  2014     3
10675         835      Puducherry         Puducherry  2014     7
10676         836      Puducherry  Total District(s)  2014    10

       Kidnapping and Abduction  Dowry Deaths  \
0                            30            16
1                            30             7
2                            34            14
3                            20            17
```

```
4                              26             12
…                             …              …
10672                           0              0
10673                           0              0
10674                           1              0
10675                           6              1
10676                           7              1

       Assault on women with intent to outrage her modesty  \
0                                                      149
1                                                      118
2                                                      112
3                                                      126
4                                                      109
…                                                     …
10672                                                    1
10673                                                    1
10674                                                   12
10675                                                   20
10676                                                   32

       Insult to modesty of Women  Cruelty by Husband or his Relatives  \
0                              34                                   175
1                              24                                   154
2                              83                                   186
3                              38                                    57
4                              58                                   247
…                             …                                    …
10672                           2                                     0
10673                           2                                     0
10674                           1                                     1
10675                           7                                     3
10676                           8                                     4

       Importation of Girls
0                         0
1                         0
2                         0
3                         0
4                         0
…                        …
10672                     0
10673                     0
10674                     0
10675                     0
10676                     0
```

```
[10677 rows x 11 columns]
```

Let us find out the number of rows and column of the particular dataset

[6]:
```
crimes_df.shape
```

[6]:
```
(10677, 11)
```

2 Data Preparation and Cleaning

To start with the very basic of data cleaning, let's find out if any of the columns have any Null or missing values

[7]:
```
crimes_df.isna().sum()
```

[7]:
```
Unnamed: 0                                          0
STATE/UT                                            0
DISTRICT                                            0
Year                                                0
Rape                                                0
Kidnapping and Abduction                            0
Dowry Deaths                                        0
Assault on women with intent to outrage her modesty 0
Insult to modesty of Women                          0
Cruelty by Husband or his Relatives                 0
Importation of Girls                                0
dtype: int64
```

None of the column has any Null values.

Now let's find the total number of 'Unique' districts, where the crimes have been committed

[8]:
```
districts = len(crimes_df.DISTRICT.unique())
```

[9]:
```
districts
```

[9]:
```
1605
```

But there are 718 districts in India, in total, which means there is messy or false datas in a huge amount, in this case, we better drop the column "District" and also "Unnamed: 0", as it is of no use, in our data analysis process.

[10]:
```
crimes_df.drop(['DISTRICT', 'Unnamed: 0'], axis = 1, inplace=True)
```

[11]:
```
crimes_df.rename( columns = {'Kidnapping and Abduction':
 ↪'Kidnapping_Abduction','Dowry Deaths':'Dowry_Deaths',
```

```
                              'Assault on women with intent to outrage her␣
↪modesty':'Hurting_of_womens_modesty',
                              'Insult to modesty of Women':
↪'Insult_to_womens_modesty',
                              'Cruelty by Husband or his Relatives':
↪'Domestic_Cruelty',
                              'Importation of Girls':'Importation_of_Girls'},␣
↪inplace = True)
```

[12]:
```
crimes_df
```

[12]:

	STATE/UT	Year	Rape	Kidnapping_Abduction	Dowry_Deaths	\
0	ANDHRA PRADESH	2001	50	30	16	
1	ANDHRA PRADESH	2001	23	30	7	
2	ANDHRA PRADESH	2001	27	34	14	
3	ANDHRA PRADESH	2001	20	20	17	
4	ANDHRA PRADESH	2001	23	26	12	
…	…	…	…	…	…	
10672	Lakshadweep	2014	1	0	0	
10673	Lakshadweep	2014	1	0	0	
10674	Puducherry	2014	3	1	0	
10675	Puducherry	2014	7	6	1	
10676	Puducherry	2014	10	7	1	

	Hurting_of_womens_modesty	Insult_to_womens_modesty	Domestic_Cruelty	\
0	149	34	175	
1	118	24	154	
2	112	83	186	
3	126	38	57	
4	109	58	247	
…	…	…	…	
10672	1	2	0	
10673	1	2	0	
10674	12	1	1	
10675	20	7	3	
10676	32	8	4	

	Importation_of_Girls
0	0
1	0
2	0
3	0
4	0
…	…
10672	0
10673	0
10674	0

```
10675                    0
10676                    0

[10677 rows x 9 columns]
```

Now, let's start with analysing the datas of the column "STATE/UT", for that let's find out the names of all the states/UT through .unique()

[13]:
```
print(crimes_df['STATE/UT'].unique())
```

```
['ANDHRA PRADESH' 'ARUNACHAL PRADESH' 'ASSAM' 'BIHAR' 'CHHATTISGARH' 'GOA'
 'GUJARAT' 'HARYANA' 'HIMACHAL PRADESH' 'JAMMU & KASHMIR' 'JHARKHAND'
 'KARNATAKA' 'KERALA' 'MADHYA PRADESH' 'MAHARASHTRA' 'MANIPUR' 'MEGHALAYA'
 'MIZORAM' 'NAGALAND' 'ODISHA' 'PUNJAB' 'RAJASTHAN' 'SIKKIM' 'TAMIL NADU'
 'TRIPURA' 'UTTAR PRADESH' 'UTTARAKHAND' 'WEST BENGAL' 'A & N ISLANDS'
 'CHANDIGARH' 'D & N HAVELI' 'DAMAN & DIU' 'DELHI' 'LAKSHADWEEP'
 'PUDUCHERRY' 'Andhra Pradesh' 'Arunachal Pradesh' 'Assam' 'Bihar'
 'Chhattisgarh' 'Goa' 'Gujarat' 'Haryana' 'Himachal Pradesh'
 'Jammu & Kashmir' 'Jharkhand' 'Karnataka' 'Kerala' 'Madhya Pradesh'
 'Maharashtra' 'Manipur' 'Meghalaya' 'Mizoram' 'Nagaland' 'Odisha'
 'Punjab' 'Rajasthan' 'Sikkim' 'Tamil Nadu' 'Tripura' 'Uttar Pradesh'
 'Uttarakhand' 'West Bengal' 'A&N Islands' 'Chandigarh' 'D&N Haveli'
 'Daman & Diu' 'Delhi UT' 'Lakshadweep' 'Puducherry' 'Telangana'
 'A & N Islands']
```

We can see from above that there are lot many repeated datas, like some of them are repeated again by using capital letters and some of them have issues with space too, like A&N Islands and also Delhi has been repeated again by mentioning it as Delhi UT

[14]:
```
# Fist we will remove all the repeated uppercase values
def remove_uppercase(r):
    r = r['STATE/UT'].strip()
    r = r.upper()
    return r
crimes_df['STATE/UT'] = crimes_df.apply(remove_uppercase, axis=1)

#Now use replace function to replace the other type of repeated datas as␣
 ↪dicussed above
crimes_df['STATE/UT'].replace("A&N ISLANDS", "A & N ISLANDS", inplace = True)
crimes_df['STATE/UT'].replace("D&N HAVELI", "D & N HAVELI", inplace = True)
crimes_df['STATE/UT'].replace("DELHI UT", "DELHI", inplace = True)
```

Let's go through the datas now!

[15]:
```
crimes_df['STATE/UT'].unique()
```

[15]:
```
array(['ANDHRA PRADESH', 'ARUNACHAL PRADESH', 'ASSAM', 'BIHAR',
       'CHHATTISGARH', 'GOA', 'GUJARAT', 'HARYANA', 'HIMACHAL PRADESH',
       'JAMMU & KASHMIR', 'JHARKHAND', 'KARNATAKA', 'KERALA',
       'MADHYA PRADESH', 'MAHARASHTRA', 'MANIPUR', 'MEGHALAYA', 'MIZORAM',
       'NAGALAND', 'ODISHA', 'PUNJAB', 'RAJASTHAN', 'SIKKIM',
       'TAMIL NADU', 'TRIPURA', 'UTTAR PRADESH', 'UTTARAKHAND',
       'WEST BENGAL', 'A & N ISLANDS', 'CHANDIGARH', 'D & N HAVELI',
       'DAMAN & DIU', 'DELHI', 'LAKSHADWEEP', 'PUDUCHERRY', 'TELANGANA'],
      dtype=object)
```

Let's check the total number of States+UT

[16]:
```
len(crimes_df['STATE/UT'].unique())
```

[16]:
```
36
```

Which is coming out perfect, hence we are done with our data cleaning process of our dataset

3 Exploratory Analysis and Visualization

Let us find out the total population of women over the years, 2001-2014, who has been a victim of the crime based on their gender.

[17]:
```
victims_raped = crimes_df.Rape.sum()
victims_kidnapped_abducted = crimes_df.Kidnapping_Abduction.sum()
dowery_death = crimes_df.Dowry_Deaths.sum()
modesty_assault = crimes_df.Hurting_of_womens_modesty.sum()
insult_to_modesty = crimes_df.Insult_to_womens_modesty.sum()
domestic_violence = crimes_df.Domestic_Cruelty.sum()
girls_imported = crimes_df.Importation_of_Girls.sum()
```

[18]:
```
total_population_of_victim_overall = victims_raped + victims_raped +␣
↪dowery_death +modesty_assault+ insult_to_modesty + domestic_violence+␣
↪girls_imported
total_population_of_victim_overall
```

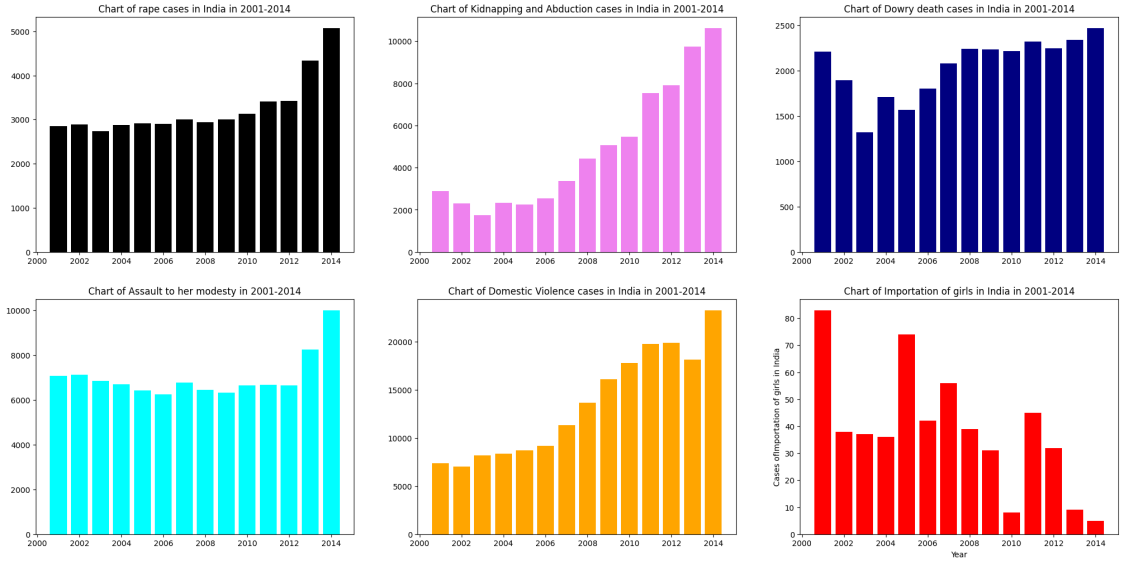
[18]:
```
5194570
```

This above analysis potrays a heartbreaking situation of women in our society, as more than 5 million number of females, over the years 2001-2014, have been a victim of assault, violance, rape or even death, in India alone.

Now let us analyse the all the cases sepately by using bar graph. Note - For our ease, we are doing the analysis for six caterories, excluding the "Insult_to_modesty_of_Women" column.

[19]:
```
fig, axes = plt.subplots(2, 3, figsize=(25, 12))

axes[0,0].set_title("Chart of rape cases in India in 2001-2014")
axes[0,0].bar(crimes_df.Year, crimes_df.Rape, color = 'black');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases of Rape in India') #Y-axis

axes[0,1].set_title("Chart of Kidnapping and Abduction cases in India in␣
 ↪2001-2014")
axes[0,1].bar(crimes_df.Year, crimes_df.Kidnapping_Abduction, color = 'violet');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases of Kidnapping and Abduction in India') #Y-axis

axes[0,2].set_title("Chart of Dowry death cases in India in 2001-2014")
axes[0,2].bar(crimes_df.Year, crimes_df.Dowry_Deaths, color = 'navy');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases of Dowry deaths in India') #Y-axis

axes[1,0].set_title("Chart of Assault to her modesty in 2001-2014")
axes[1,0].bar(crimes_df.Year, crimes_df.Hurting_of_womens_modesty, color =␣
 ↪'cyan');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases of Assaulting a women for her modesty in India') #Y-axis

axes[1,1].set_title("Chart of Domestic Violence cases in India in 2001-2014")
axes[1,1].bar(crimes_df.Year, crimes_df.Domestic_Cruelty, color = 'orange');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases of Domestic Violence in India') #Y-axis

axes[1,2].set_title("Chart of Importation of girls in India in 2001-2014")
axes[1,2].bar(crimes_df.Year, crimes_df.Importation_of_Girls, color = 'red');
plt.xlabel('Year') #X-axis
plt.ylabel('Cases ofImportation of girls in India') #Y-axis
```

[19]:
```
Text(0, 0.5, 'Cases ofImportation of girls in India')
```



3.0.1 There are two things to be concluded from the above bar chart -

1) The cases have incraesed over the years.

2) 2014 has been the year, where violance against women was reported the maximum, under each of the cases like, rape, domestic violence etc, which can also be proved below, which can also be cross examined with the code below.

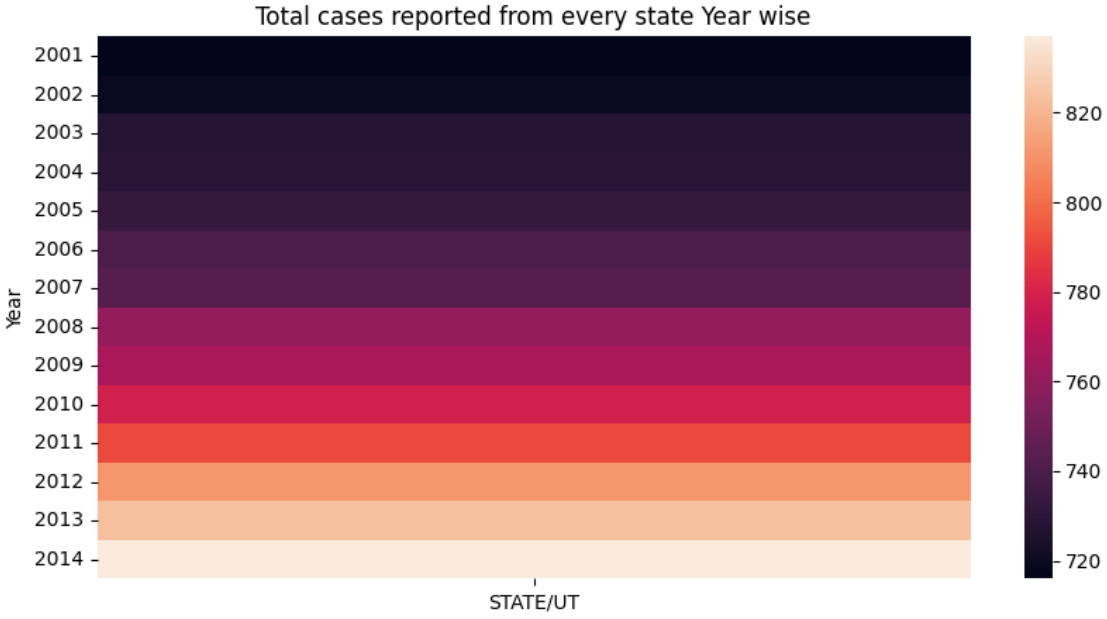
[20]:
```
count_df = crimes_df.groupby('Year')[['STATE/UT']].count()
count_df
```

[20]:

Year	STATE/UT
2001	716
2002	719
2003	728
2004	729
2005	733
2006	740
2007	743
2008	761
2009	767
2010	779
2011	791
2012	811
2013	823
2014	837

Let us use seaborn to take help of heatmap to represent how more cases from each state started coming up more frequently with each passing year.

[21]:
```
plt.figure(figsize=(10,5))
plt.title("Total cases reported from every state Year wise")
sns.heatmap(count_df);
```



3.0.2 This heatmap shows how more number of state's name started appearing on the dataset, for reporting crimes and sadly the crime only increased with the passing year.

It gives us the conclusion that overall(cases under each category), from every state, has increased with time, not decreased.

4 Asking and Answering Questions

4.0.1 As part of this data analysis, it is very crucial to raise question and find answer to them. Here we will try to find out some of the most essential questions, that will help us in drawing a major conclusion from our dataset.

Note - We will be focusing on four major categories for the rest of the analysis, excluding "Assault_for_her_modesty" and "Insult_to_modesty_of_Women".

[22]:
```
crimes_df = crimes_df.drop(['Hurting_of_womens_modesty',␣
 ↪'Insult_to_womens_modesty'], axis=1)
```

Q . Create a dataframe containing 10 highest reported rape cases in India, in the span of year 2001-2014.

[23]:
```
max_rape_cases = crimes_df.sort_values('Rape', ascending = False).head(10)
max_rape_cases
```

[23]:

	STATE/UT	Year	Rape	Kidnapping_Abduction	Dowry_Deaths	Domestic_Cruelty	Importation_of_Girls
10244	MADHYA PRADESH	2014	5076	5688	733	6451	0
9426	MADHYA PRADESH	2013	4335	2873	776	4988	7
10445	RAJASTHAN	2014	3759	4421	408	15905	0
10595	UTTAR PRADESH	2014	3467	10626	2469	10471	0
10291	MAHARASHTRA	2014	3438	2457	279	7696	1
8611	MADHYA PRADESH	2012	3425	1127	743	3988	6
7810	MADHYA PRADESH	2011	3406	1088	811	3732	45
9628	RAJASTHAN	2013	3285	4047	453	15094	1
7025	MADHYA PRADESH	2010	3135	1030	892	3756	5
9472	MAHARASHTRA	2013	3063	1874	320	8542	0

We see from the analysis, the top states that reported the maximum number of rape cases, along with the year, in which they occured.

Where, Madhya Pradesh reported the maximum rape cases in the year 2014.

4.0.2 Q. Create a dataframe containing 10 highest reported deaths caused by Dowry cases in India, in the span of year 2001-2014.

[24]:
```
max_dowry_death_cases = crimes_df.sort_values('Dowry_Deaths', ascending =␣
↪False).head(10)
max_dowry_death_cases
```

[24]:

	STATE/UT	Year	Rape	Kidnapping_Abduction	Dowry_Deaths
10595	UTTAR PRADESH	2014	3467	10626	2469
9760	UTTAR PRADESH	2013	3050	9737	2335
8132	UTTAR PRADESH	2011	2042	7525	2322
8938	UTTAR PRADESH	2012	1963	7910	2244
5796	UTTAR PRADESH	2008	1871	4439	2237
6563	UTTAR PRADESH	2009	1759	5078	2232
7342	UTTAR PRADESH	2010	1563	5468	2217

```
650    UTTAR PRADESH  2001  1958                  2879          2211
5040   UTTAR PRADESH  2007  1648                  3363          2076
1366   UTTAR PRADESH  2002  1415                  2298          1893

       Domestic_Cruelty  Importation_of_Girls
10595             10471                     0
9760               8781                     0
8132               7121                     0
8938               7661                     0
5796               8312                     0
6563               8566                     0
7342               7978                     0
650                7365                     0
5040               7650                     0
1366               5679                     0
```

From our analysis, we observe that the highest reported dowry death was in Uttar Pradesh in 2014, with number of reports being 2469.

One more here to be observed as well is that, Uttar Pradesh is the ONLY state that appears in this list.

4.0.3 Q. Create a dataframe containing 10 highest reported Domestic Violance cases in India, in the span of year 2001-2014.

[25]:
```
max_domestic_violance_cases = crimes_df.sort_values('Domestic_Cruelty',␣
 ↪ascending = False).head(10)
max_domestic_violance_cases
```

[25]:
```
             STATE/UT  Year  Rape  Kidnapping_Abduction  Dowry_Deaths  \
10640     WEST BENGAL  2014  1466                  4976           501
8982      WEST BENGAL  2012  2046                  4168           593
8172      WEST BENGAL  2011  2363                  3711           510
9804      WEST BENGAL  2013  1685                  3830           481
7381      WEST BENGAL  2010  2311                  2764           507
6602      WEST BENGAL  2009  2336                  2187           506
10445       RAJASTHAN  2014  3759                  4421           408
9628        RAJASTHAN  2013  3285                  4047           453
9050   ANDHRA PRADESH  2013  1635                  1595           492
5835      WEST BENGAL  2008  2263                  1907           451

       Domestic_Cruelty  Importation_of_Girls
10640             23278                     4
8982              19865                    12
8172              19772                     0
9804              18116                     9
```

7381	17796	8
6602	16112	5
10445	15905	0
9628	15094	1
9050	15084	0
5835	13663	5

According to our analysis, we see the maximum cases of Domestic Violance cases came from West Bengal in the year 2014, with number of cases reported being 23278.

4.0.4 Q. Create a dataframe containing 10 highest reported Importation cases in India, in the span of year 2001-2014.

[26]:
```
max_importation_case = crimes_df.sort_values('Importation_of_Girls', ascending
 ↪= False).head(10)
max_importation_case
```

[26]:

	STATE/UT	Year	Rape	Kidnapping_Abduction	Dowry_Deaths	\
115	BIHAR	2001	888	518	859	
3013	BIHAR	2005	1147	929	1014	
3597	WEST BENGAL	2005	1686	1039	446	
3590	WEST BENGAL	2005	148	97	48	
4486	BIHAR	2007	1555	1260	1172	
3005	BIHAR	2005	28	4	40	
7810	MADHYA PRADESH	2011	3406	1088	811	
3746	BIHAR	2006	1232	1084	1188	
102	BIHAR	2001	27	11	90	
5378	JHARKHAND	2008	791	499	266	

	Domestic_Cruelty	Importation_of_Girls
115	1558	83
3013	1574	74
3597	6936	61
3590	545	60
4486	1635	56
3005	73	48
7810	3732	45
3746	1689	42
102	152	39
5378	851	39

According to our analysis, maximum of Importation of girls has been reported in Bihar in the year 2011

4.0.5 Q. Find out the total number of cases, in span of 2001-2014 under each category, state wise.

[27]:
```
counts_df = crimes_df.groupby('STATE/UT')[['Rape', 'Kidnapping_Abduction',
 ↪'Dowry_Deaths','Domestic_Cruelty', 'Importation_of_Girls']].sum()
counts_df
```

[27]:

STATE/UT	Rape	Kidnapping_Abduction	Dowry_Deaths	Domestic_Cruelty	Importation_of_Girls
A & N ISLANDS	336	212	20		
ANDHRA PRADESH	32150	34504	13844		
ARUNACHAL PRADESH	1316	1470	6		
ASSAM	40190	62074	3268		
BIHAR	30758	57086	32206		
CHANDIGARH	770	1682	90		
CHHATTISGARH	29308	11808	2758		
D & N HAVELI	132	224	2		
DAMAN & DIU	60	44	6		
DELHI	20312	46586	3758		
GOA	1062	640	38		
GUJARAT	11644	34670	1108		
HARYANA	17110	20016	7372		
HIMACHAL PRADESH	4674	4116	112		
JAMMU & KASHMIR	7038	21164	294		
JHARKHAND	22826	14186	7896		
KARNATAKA	15056	16262	7016		
KERALA	20030	4452	700		
LAKSHADWEEP	20	2	0		
MADHYA PRADESH	90996	35608	21090		
MAHARASHTRA	48974	30368	9696		
MANIPUR	1068	2606	6		
MEGHALAYA	2642	670	36		
MIZORAM	2070	30	8		
NAGALAND	562	190	2		
ODISHA	30480	25588	10782		
PUDUCHERRY	208	306	56		
PUNJAB	14656	15096	3524		
RAJASTHAN	45684	66278	11854		
SIKKIM	570	180	4		
TAMIL NADU	16660	30908	5060		
TELANGANA	1958	1422	578		
TRIPURA	5060	2202	752		
UTTAR PRADESH	51150	135906	57256		
UTTARAKHAND	3752	6484	1974		
WEST BENGAL	47876	61158	12308		

STATE/UT		
A & N ISLANDS	288	0
ANDHRA PRADESH	280906	34
ARUNACHAL PRADESH	476	0
ASSAM	115300	22
BIHAR	69770	904
CHANDIGARH	2080	0
CHHATTISGARH	23436	12
D & N HAVELI	90	0
DAMAN & DIU	76	0
DELHI	42834	2
GOA	532	0
GUJARAT	146468	0
HARYANA	68414	4
HIMACHAL PRADESH	7796	0
JAMMU & KASHMIR	5390	0
JHARKHAND	23910	298
KARNATAKA	72706	94
KERALA	111626	0
LAKSHADWEEP	14	0
MADHYA PRADESH	102816	134
MAHARASHTRA	193202	6
MANIPUR	578	0
MEGHALAYA	460	8
MIZORAM	134	6
NAGALAND	32	2
ODISHA	49206	36
PUDUCHERRY	234	0
PUNJAB	30840	4
RAJASTHAN	262200	14
SIKKIM	108	0
TAMIL NADU	45524	30
TELANGANA	12738	0
TRIPURA	16086	0
UTTAR PRADESH	193738	6
UTTARAKHAND	9756	2
WEST BENGAL	344124	254

4.0.6 Q. Find out the top 5 states, where maximum numbers of cases has been reported in TOTAL in span of 2001-2014, each category wise.

For "Rape" case -

[28]:
```
counts_df.sort_values(by = 'Rape', ascending = False).head(5)
```

[28]:

STATE/UT	Rape	Kidnapping_Abduction	Dowry_Deaths	Domestic_Cruelty	\
MADHYA PRADESH	90996	35608	21090	102816	

```
UTTAR PRADESH   51150                135906         57256            193738
MAHARASHTRA     48974                 30368          9696            193202
WEST BENGAL     47876                 61158         12308            344124
RAJASTHAN       45684                 66278         11854            262200

                Importation_of_Girls
STATE/UT
MADHYA PRADESH                   134
UTTAR PRADESH                      6
MAHARASHTRA                        6
WEST BENGAL                      254
RAJASTHAN                         14
```

Madhya Pradesh has reported the highest number of rape cases in TOTAL in span of 2001-2014, where UP, Maharastra, West Bengal and Rajasthan follows the list.

For Kidnapping and abduction case -

[29]:
```
counts_df.sort_values(by = 'Kidnapping_Abduction', ascending = False).head(5)
```

[29]:
```
                 Rape  Kidnapping_Abduction  Dowry_Deaths  Domestic_Cruelty  \
STATE/UT
UTTAR PRADESH   51150                135906         57256            193738
RAJASTHAN       45684                 66278         11854            262200
ASSAM           40190                 62074          3268            115300
WEST BENGAL     47876                 61158         12308            344124
BIHAR           30758                 57086         32206             69770

                Importation_of_Girls
STATE/UT
UTTAR PRADESH                      6
RAJASTHAN                         14
ASSAM                             22
WEST BENGAL                      254
BIHAR                            904
```

Uttar Pradesh has reported the highest number of cases under "Kidnapping and Abduction" in TOTAL in span of 2001-2014, where Rajasthan, Assam, West Bengal and Bihar follows the list.

For cases of deaths due to dowry -

[30]:
```
counts_df.sort_values(by = 'Dowry_Deaths', ascending = False).head(5)
```

[30]:
```
                 Rape  Kidnapping_Abduction  Dowry_Deaths  Domestic_Cruelty  \
STATE/UT
UTTAR PRADESH   51150                135906         57256            193738
```

BIHAR	30758	57086	32206	69770
MADHYA PRADESH	90996	35608	21090	102816
ANDHRA PRADESH	32150	34504	13844	280906
WEST BENGAL	47876	61158	12308	344124

STATE/UT	Importation_of_Girls
UTTAR PRADESH	6
BIHAR	904
MADHYA PRADESH	134
ANDHRA PRADESH	34
WEST BENGAL	254

Uttar Pradesh has reported the highest number of Deaths caused by Dowry cases in TOTAL in span of 2001-2014, where Bihar, Madhya Pradesh, Andhra Pradesh and West Bengal follows the list.

For Domestic Violance case -

[31]:
```
counts_df.sort_values(by = 'Domestic_Cruelty', ascending = False).head(5)
```

[31]:

STATE/UT	Rape	Kidnapping_Abduction	Dowry_Deaths	Domestic_Cruelty	\
WEST BENGAL	47876	61158	12308	344124	
ANDHRA PRADESH	32150	34504	13844	280906	
RAJASTHAN	45684	66278	11854	262200	
UTTAR PRADESH	51150	135906	57256	193738	
MAHARASHTRA	48974	30368	9696	193202	

STATE/UT	Importation_of_Girls
WEST BENGAL	254
ANDHRA PRADESH	34
RAJASTHAN	14
UTTAR PRADESH	6
MAHARASHTRA	6

West Bengal has reported the highest number of cases of Domestic Violence in TOTAL in span of 2001-2014, where Andhra Pradesh, Rajsthan, Uttar Pradesh and Maharashtra follows the list.

For Importation of Girls case -

[32]:
```
counts_df.sort_values(by = 'Importation_of_Girls', ascending = False).head(5)
```

[32]:

STATE/UT	Rape	Kidnapping_Abduction	Dowry_Deaths	Domestic_Cruelty	\
BIHAR	30758	57086	32206	69770	

```
JHARKHAND       22826                14186          7896            23910
WEST BENGAL     47876                61158         12308           344124
MADHYA PRADESH  90996                35608         21090           102816
KARNATAKA       15056                16262          7016            72706

                Importation_of_Girls
STATE/UT
BIHAR                            904
JHARKHAND                        298
WEST BENGAL                      254
MADHYA PRADESH                   134
KARNATAKA                         94
```

Bihar has reported the highest number of Importation of girl's cases in TOTAL in span of 2001-2014, where Jharkhand, West Bengal, Madhya Pradesh and Karnataka follows the list.

Q: Which state has featured in both the lists of "Maximum number of rape cases" and "Maximum number of Importation cases".

[33]:
```
max_importation_case = max_importation_case.merge(max_rape_cases)
max_importation_case
```

[33]:
```
          STATE/UT  Year  Rape  Kidnapping_Abduction  Dowry_Deaths  \
0  MADHYA PRADESH  2011  3406                  1088           811

   Domestic_Cruelty  Importation_of_Girls
0              3732                    45
```

It is Madhya Pradesh, that has maximum cases reported in both the categories.

Q: Which state has featured in both the lists of "Maximum number of rape cases" and "Maximum number of Deaths due to Dowry cases".

[34]:
```
max_dowry_death_cases = max_dowry_death_cases.merge(max_rape_cases)
max_dowry_death_cases
```

[34]:
```
         STATE/UT  Year  Rape  Kidnapping_Abduction  Dowry_Deaths  \
0  UTTAR PRADESH  2014  3467                 10626          2469

   Domestic_Cruelty  Importation_of_Girls
0             10471                     0
```

We conclude, it is Uttar Pradesh, that has reported maximum cases in both the given categories.

5 Inferences and Conclusion

The main aim of the project was to analyse the situation of women in the year 2001-2014.

We also did a deep analysis through charts and by raising important questions. Let us go through some of the important analysis, we have done through thid project -

1)More than 5 million females has been a victim of some or other type of Violance, based
starting from rape to importing them for buisness.
2) We concluded from the series of bar graphs that 2014 was the year, when crimes were re
under each category.
3) We tried finding out the top 10 highest cases reported ever, along with year in which
and in which state.Where, Madhya Pradesh having highest number of cases of rape in 2014,
highest cases in Dowry death in 2014, West Bengal having highest cases in Domestic Violar
having the highest cases in importaion of girls in 2011.
4)We summarised the TOTAL number of cases happening, in 2001-2014, by each state.
5)We also found out the top 5 states where maximum number of TOTAL cases has been reporte
state wise.
6)We also merged the data in two different cases, first one being "Maximum number of rape
"Maximum number of Importation cases", where we found out it is Madhya Pradesh and in sec
"Maximum number of rape cases" and "Maximum number of Deaths due to Dowry cases", which w
be Uttar Pradesh.

home-loan-approval

December 11, 2023

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

[2]:
```
import os
for dirname, _, filenames in os.walk("C:/Users/ahap0/Downloads/
 ↪loan_sanction_train.csv"):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[3]:
```
df=pd.read_csv("C:/Users/ahap0/Downloads/loan_sanction_train.csv")
```

[4]:
```
df.head()
```

[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

[5]:
```
df.shape
```

[5]:
```
(614, 13)
```

ques 1:calculate the mean median max value, min value, standard deviation &quartile?

[6]:
```
df.describe()
```

[6]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

[7]:
```
df.describe(include='O')
```

[7]:
```
         Loan_ID Gender Married Dependents Education Self_Employed  \
count        614    601     611        599       614           582
unique       614      2       2          4         2             2
top     LP001002   Male     Yes          0  Graduate            No
freq           1    489     398        345       480           500

       Property_Area Loan_Status
count            614         614
unique             3           2
top        Semiurban           Y
freq             233         422
```

[8]:
```
df.columns
```

[8]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
```

```
      dtype='object')
```

ques 2: How to drop the loan I'd column?

[9]:
```
df=df.drop(columns=['Loan_ID'],axis=1)
```

[10]:
```
from sklearn.model_selection import train_test_split
train,test=train_test_split(df,test_size=0.2,random_state=5)
```

[11]:
```
df.isna().sum()/(len(df))
```

[11]:
```
Gender               0.021173
Married              0.004886
Dependents           0.024430
Education            0.000000
Self_Employed        0.052117
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.035831
Loan_Amount_Term     0.022801
Credit_History       0.081433
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

[12]:
```
num_attributes= df.select_dtypes(include=['int64', 'float64']).columns;

cat_attributes=df.select_dtypes(include=['object', 'category']).columns;

train_numerical=train[num_attributes]
train_categorical=train[cat_attributes]

test_numerical=test[num_attributes]
test_categorical=test[cat_attributes]
```

[13]:
```
from sklearn.impute import SimpleImputer
cat_imputer=SimpleImputer(strategy='most_frequent')
train_categorical=pd.DataFrame(cat_imputer.
 ↪fit_transform(train_categorical),columns=train_categorical.columns)
test_categorical=pd.DataFrame(cat_imputer.
 ↪transform(test_categorical),columns=test_categorical.columns)
```

[14]:
```
num_imputer=SimpleImputer(strategy='median')
train_numerical=pd.DataFrame(num_imputer.
 ↪fit_transform(train_numerical),columns=train_numerical.columns)
test_numerical=pd.DataFrame(num_imputer.
 ↪transform(test_numerical),columns=test_numerical.columns)
```

[15]:
```
train.shape,test.shape
```

[15]:
```
((491, 12), (123, 12))
```

[17]:
```
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder(drop='first',sparse=False)
```

[22]:
```
train_categorical=pd.DataFrame(ohe.fit_transform(train_categorical),columns=ohe.
↪get_feature_names_out())
```

```
C:\Users\ahap0\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

ques 3: Define a data in categorical form?

[23]:
```
train_categorical
```

[23]:

	Gender_Male_1.0_1.0_1.0_1.0_1.0	Married_Yes_1.0_1.0_1.0_1.0_1.0	\
0	1.0	1.0	
1	1.0	0.0	
2	1.0	1.0	
3	0.0	0.0	
4	1.0	1.0	
..	…	…	
486	1.0	1.0	
487	1.0	1.0	
488	1.0	1.0	
489	1.0	1.0	
490	0.0	0.0	

	Dependents_1_1.0_1.0_1.0_1.0_1.0	Dependents_2_1.0_1.0_1.0_1.0_1.0	\
0	0.0	1.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	1.0	
..	…	…	
486	0.0	1.0	
487	0.0	0.0	
488	0.0	1.0	
489	0.0	0.0	
490	0.0	0.0	

	Dependents_3+_1.0_1.0_1.0_1.0_1.0	\
0	0.0	

```
1                                    0.0
2                                    0.0
3                                    0.0
4                                    0.0
..                                   …
486                                  0.0
487                                  1.0
488                                  0.0
489                                  0.0
490                                  0.0

     Education_Not Graduate_1.0_1.0_1.0_1.0_1.0  \
0                                            0.0
1                                            1.0
2                                            0.0
3                                            0.0
4                                            0.0
..                                           …
486                                          0.0
487                                          1.0
488                                          1.0
489                                          0.0
490                                          0.0

     Self_Employed_Yes_1.0_1.0_1.0_1.0_1.0  \
0                                      1.0
1                                      0.0
2                                      0.0
3                                      0.0
4                                      1.0
..                                     …
486                                    0.0
487                                    0.0
488                                    0.0
489                                    0.0
490                                    0.0

     Property_Area_Semiurban_1.0_1.0_1.0_1.0_1.0  \
0                                             1.0
1                                             0.0
2                                             1.0
3                                             0.0
4                                             0.0
..                                            …
486                                           0.0
487                                           1.0
488                                           0.0
```

```
489                                          0.0
490                                          0.0

     Property_Area_Urban_1.0_1.0_1.0_1.0_1.0  \
0                                        0.0
1                                        0.0
2                                        0.0
3                                        0.0
4                                        0.0
..                                       …
486                                      1.0
487                                      0.0
488                                      1.0
489                                      0.0
490                                      1.0

     Loan_Status_Y_1.0_1.0_1.0_1.0_1.0
0                                  1.0
1                                  0.0
2                                  1.0
3                                  1.0
4                                  1.0
..                                 …
486                                1.0
487                                0.0
488                                0.0
489                                0.0
490                                1.0

[491 rows x 10 columns]
```

[23]:
```
train_categorical
```

[23]:
```
     Gender_Male_1.0_1.0_1.0_1.0_1.0  Married_Yes_1.0_1.0_1.0_1.0_1.0  \
0                                1.0                              1.0
1                                1.0                              0.0
2                                1.0                              1.0
3                                0.0                              0.0
4                                1.0                              1.0
..                               …                                …
486                              1.0                              1.0
487                              1.0                              1.0
488                              1.0                              1.0
489                              1.0                              1.0
490                              0.0                              0.0

     Dependents_1_1.0_1.0_1.0_1.0_1.0  Dependents_2_1.0_1.0_1.0_1.0_1.0  \
```

```
0                                    0.0                                    1.0
1                                    0.0                                    0.0
2                                    0.0                                    0.0
3                                    0.0                                    0.0
4                                    0.0                                    1.0
..                                   …                                    …
486                                  0.0                                    1.0
487                                  0.0                                    0.0
488                                  0.0                                    1.0
489                                  0.0                                    0.0
490                                  0.0                                    0.0

     Dependents_3+_1.0_1.0_1.0_1.0_1.0  \
0                                     0.0
1                                     0.0
2                                     0.0
3                                     0.0
4                                     0.0
..                                    …
486                                   0.0
487                                   1.0
488                                   0.0
489                                   0.0
490                                   0.0

     Education_Not Graduate_1.0_1.0_1.0_1.0_1.0  \
0                                            0.0
1                                            1.0
2                                            0.0
3                                            0.0
4                                            0.0
..                                           …
486                                          0.0
487                                          1.0
488                                          1.0
489                                          0.0
490                                          0.0

     Self_Employed_Yes_1.0_1.0_1.0_1.0_1.0  \
0                                       1.0
1                                       0.0
2                                       0.0
3                                       0.0
4                                       1.0
..                                      …
486                                     0.0
487                                     0.0
```

```
488                                    0.0
489                                    0.0
490                                    0.0

     Property_Area_Semiurban_1.0_1.0_1.0_1.0_1.0  \
0                                             1.0
1                                             0.0
2                                             1.0
3                                             0.0
4                                             0.0
..                                            …
486                                           0.0
487                                           1.0
488                                           0.0
489                                           0.0
490                                           0.0

     Property_Area_Urban_1.0_1.0_1.0_1.0_1.0  \
0                                         0.0
1                                         0.0
2                                         0.0
3                                         0.0
4                                         0.0
..                                        …
486                                       1.0
487                                       0.0
488                                       1.0
489                                       0.0
490                                       1.0

     Loan_Status_Y_1.0_1.0_1.0_1.0_1.0
0                                  1.0
1                                  0.0
2                                  1.0
3                                  1.0
4                                  1.0
..                                 …
486                                1.0
487                                0.0
488                                0.0
489                                0.0
490                                1.0

[491 rows x 10 columns]
```

ques4:Add the numerical and categorical data using concat function ?

[24]:
```
train=pd.concat([train_numerical,train_categorical],axis=1)
```

[25]:
```
test=pd.concat([test_numerical,test_categorical],axis=1)
```

ques5: Define the column names or labels within a Pandas DataFrame?

[26]:
```
train.columns
```

[26]:
```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Gender_Male_1.0_1.0_1.0_1.0_1.0',
       'Married_Yes_1.0_1.0_1.0_1.0_1.0', 'Dependents_1_1.0_1.0_1.0_1.0_1.0',
       'Dependents_2_1.0_1.0_1.0_1.0_1.0', 'Dependents_3+_1.0_1.0_1.0_1.0_1.0',
       'Education_Not Graduate_1.0_1.0_1.0_1.0_1.0',
       'Self_Employed_Yes_1.0_1.0_1.0_1.0_1.0',
       'Property_Area_Semiurban_1.0_1.0_1.0_1.0_1.0',
       'Property_Area_Urban_1.0_1.0_1.0_1.0_1.0',
       'Loan_Status_Y_1.0_1.0_1.0_1.0_1.0'],
      dtype='object')
```

laptop-price-pridiction

December 11, 2023

[59]:
```
print("Bismillah")
```

```
Bismillah
```

[60]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load your laptop price dataset (replace 'your_laptop_dataset.csv' with your␣
 ↪actual dataset file)
df = pd.read_csv('/content/drive/MyDrive/data.csv')
print(df.head(10))
print("Executed...AJ")
```

```
   Unnamed: 0.1  Unnamed: 0    brand                                    name  \
0             0           0       HP      Victus 15-fb0157AX Gaming Laptop
1             1           1       HP                   15s-fq5007TU Laptop
2             2           2     Acer                  One 14 Z8-415 Laptop
3             3           3   Lenovo  Yoga Slim 6 14IAP8 82WU0095IN Laptop
4             4           4    Apple       MacBook Air 2020 MGND3HN Laptop
5             5           5     Acer               Extensa EX214-53 Laptop
6             6           6     Dell     Inspiron 3520 D560896WIN9B Laptop
7             7           7     Acer   Nitro V ANV15-51 2023 Gaming Laptop
8             8           8     Asus    Vivobook 15 X1502ZA-EJ523WS Laptop
9             9           9  Samsung            Galaxy Book2 Pro 13 Laptop

   price  spec_rating                      processor  \
0  49900    73.000000      5th Gen AMD Ryzen 5 5600H
1  39900    60.000000   12th Gen Intel Core i3 1215U
2  26990    69.323529  11th Gen Intel Core i3 1115G4
3  59729    66.000000   12th Gen Intel Core i5 1240P
4  69990    69.323529                       Apple M1
5  39990    62.000000   12th Gen Intel Core i5 1240P
6  36790    60.000000   12th Gen Intel Core i3 1215U
```

```
7  76990    63.000000  13th Gen Intel Core i5 13420H
8  48990    64.000000  12th Gen Intel Core i5 12500H
9  74990    68.000000   12th Gen Intel Core i5 1240P

                              CPU   Ram Ram_type    ROM ROM_type  \
0           Hexa Core, 12 Threads   8GB     DDR4  512GB      SSD
1   Hexa Core (2P + 4E), 8 Threads   8GB     DDR4  512GB      SSD
2            Dual Core, 4 Threads   8GB     DDR4  512GB      SSD
3   12 Cores (4P + 8E), 16 Threads  16GB   LPDDR5  512GB      SSD
4             Octa Core (4P + 4E)   8GB     DDR4  256GB      SSD
5   12 Cores (4P + 8E), 16 Threads   8GB     DDR4  512GB      SSD
6   Hexa Core (2P + 4E), 8 Threads   8GB     DDR4  512GB      SSD
7  Octa Core (4P + 4E), 12 Threads  16GB     DDR5  512GB      SSD
8   12 Cores (4P + 8E), 16 Threads   8GB     DDR4  512GB      SSD
9   12 Cores (4P + 8E), 16 Threads  16GB   LPDDR5  512GB      SSD

                           GPU  display_size  resolution_width  \
0       4GB AMD Radeon RX 6500M          15.6            1920.0
1            Intel UHD Graphics          15.6            1920.0
2         Intel Iris Xe Graphics          14.0            1920.0
3      Intel Integrated Iris Xe          14.0            2240.0
4  Apple M1 Integrated Graphics          13.3            2560.0
5         Intel Iris Xe Graphics          14.0            1920.0
6            Intel UHD Graphics          15.6            1920.0
7   6GB NVIDIA GeForce RTX 4050          15.6            1920.0
8                 Intel Iris Xe          15.6            1920.0
9         Intel Iris Xe Graphics          13.3            1080.0

   resolution_height             OS  warranty
0             1080.0  Windows 11 OS         1
1             1080.0  Windows 11 OS         1
2             1080.0  Windows 11 OS         1
3             1400.0  Windows 11 OS         1
4             1600.0         Mac OS         1
5             1080.0  Windows 11 OS         1
6             1080.0  Windows 11 OS         1
7             1080.0  Windows 11 OS         1
8             1080.0  Windows 11 OS         1
9             1920.0  Windows 11 OS         1
Executed…AJ
```

[61]:
```
# Data Inspection
print(df.info())
print("Executed...AJ")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 893 entries, 0 to 892
Data columns (total 18 columns):
```

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0.1       893 non-null    int64
 1   Unnamed: 0         893 non-null    int64
 2   brand              893 non-null    object
 3   name               893 non-null    object
 4   price              893 non-null    int64
 5   spec_rating        893 non-null    float64
 6   processor          893 non-null    object
 7   CPU                893 non-null    object
 8   Ram                893 non-null    object
 9   Ram_type           893 non-null    object
 10  ROM                893 non-null    object
 11  ROM_type           893 non-null    object
 12  GPU                893 non-null    object
 13  display_size       893 non-null    float64
 14  resolution_width   893 non-null    float64
 15  resolution_height  893 non-null    float64
 16  OS                 893 non-null    object
 17  warranty           893 non-null    int64
dtypes: float64(4), int64(4), object(10)
memory usage: 125.7+ KB
None
Executed…AJ
```

[62]:
```
# Data Cleaning
# Handling missing values
df['Unnamed: 0.1'].fillna(df['Unnamed: 0.1'].mean(), inplace=True)
df['Unnamed: 0'].fillna(df['Unnamed: 0'].median(), inplace=True)
df['brand'].fillna('brand', inplace=True)
df['name'].fillna('name', inplace=True)
df['price'].fillna(df['price'].mean(), inplace=True)
df['spec_rating'].fillna(df['spec_rating'].mean(), inplace=True)
df['processor'].fillna('processor', inplace=True)
df['CPU'].fillna('CPU', inplace=True)
df['Ram'].fillna('Ram', inplace=True)
df['Ram_type'].fillna('Ram_type', inplace=True)
df['ROM'].fillna('ROM', inplace=True)
df['ROM_type'].fillna('ROM_type', inplace=True)
df['GPU'].fillna('GPU', inplace=True)
df['display_size'].fillna(df['display_size'].mean(), inplace=True)
df['resolution_width'].fillna(df['resolution_width'].mean(), inplace=True)
df['resolution_height'].fillna(df['resolution_height'].mean(), inplace=True)
df['OS'].fillna('OS', inplace=True)
df['warranty'].fillna('warranty', inplace=True)
print("Executed...AJ")
print(df.info())
```

```
print("Executed...AJ")
```

```
Executed…AJ
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 893 entries, 0 to 892
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0.1       893 non-null    int64
 1   Unnamed: 0         893 non-null    int64
 2   brand              893 non-null    object
 3   name               893 non-null    object
 4   price              893 non-null    int64
 5   spec_rating        893 non-null    float64
 6   processor          893 non-null    object
 7   CPU                893 non-null    object
 8   Ram                893 non-null    object
 9   Ram_type           893 non-null    object
 10  ROM                893 non-null    object
 11  ROM_type           893 non-null    object
 12  GPU                893 non-null    object
 13  display_size       893 non-null    float64
 14  resolution_width   893 non-null    float64
 15  resolution_height  893 non-null    float64
 16  OS                 893 non-null    object
 17  warranty           893 non-null    int64
dtypes: float64(4), int64(4), object(10)
memory usage: 125.7+ KB
None
Executed…AJ
```

[63]:
```
# Handling outliers (assuming Price_USD is the target variable)
price_upper_limit = df['price'].quantile(0.95)
df = df[df['price'] <= price_upper_limit]
print(price_upper_limit)
```

```
199990.0
```

[64]:
```
# Data Transformation
# Log transformation on 'Price_USD' to handle skewness
df['price'] = np.log1p(df['price'])
df['price']
```

```
<ipython-input-64-a0be49c4ef4f>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['price'] = np.log1p(df['price'])
```

[64]:
```
0      10.817796
1      10.594157
2      10.203259
3      10.997590
4      11.156122
         …
888    10.714218
889    11.608245
890    12.154732
891    11.775220
892    11.790489
Name: price, Length: 849, dtype: float64
```

[65]:
```
# Split the dataset
features = df[['Unnamed: 0.1', 'Unnamed: 0',␣
 ↪'spec_rating','display_size','resolution_width','resolution_height']]
target = df['price']
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2)
X_train, X_test, y_train, y_test
```

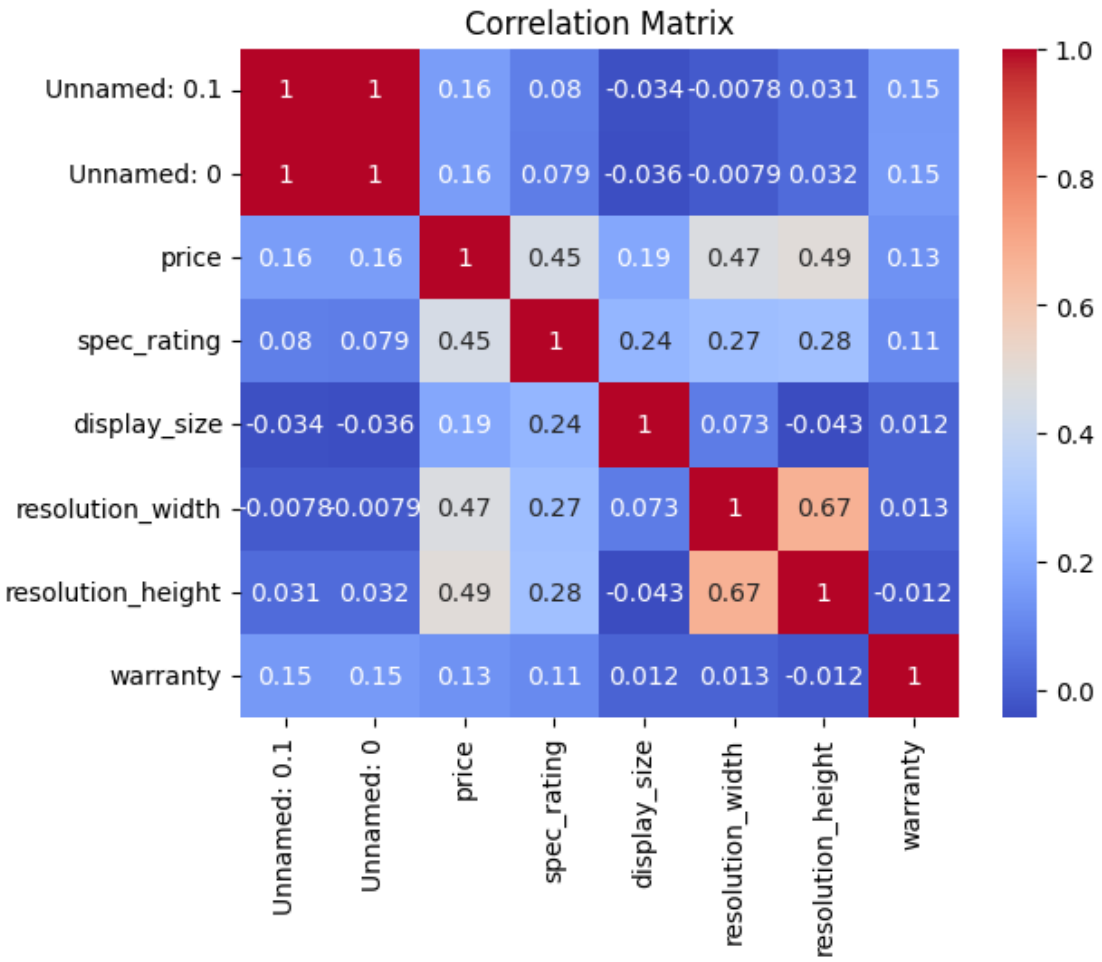
[65]:
```
(     Unnamed: 0.1  Unnamed: 0  spec_rating  display_size  resolution_width  \
 627            659         741    64.000000          15.6            1920.0
 892            930        1019    84.000000          15.6            1920.0
 859            897         985    69.323529          15.6            1920.0
 169            175         197    62.000000          15.6            1920.0
 630            662         744    69.323529          14.0            1920.0
 ..             …           …          …             …                …
 290            309         346    60.000000          15.6            1920.0
 663            695         777    69.323529          14.1            1920.0
 648            680         762    65.000000          15.6            1920.0
 651            683         765    63.000000          14.0            1366.0
 218            230         260    66.000000          15.6            1920.0

      resolution_height
 627             1080.0
 892             1080.0
 859             1080.0
 169             1080.0
 630             1080.0
 ..                 …
 290             1080.0
 663             1080.0
 648             1080.0
```

```
651              768.0
218             1080.0

[679 rows x 6 columns],
     Unnamed: 0.1  Unnamed: 0  spec_rating  display_size  resolution_width  \
126           130         144    66.000000          15.6            1920.0
368           387         428    72.000000          16.0            2880.0
382           402         447    69.323529          14.0            1920.0
363           382         423    73.000000          16.0            2880.0
421           441         505    69.323529          14.0            1920.0
..            …           …          …            …                …
59             60          63    76.000000          15.6            1920.0
374           394         435    60.000000          14.0            1920.0
493           517         585    69.323529          15.6            1920.0
79             81          84    64.000000          14.0            2880.0
465           486         551    69.323529          15.6            1920.0

     resolution_height
126             1080.0
368             1800.0
382             1080.0
363             1800.0
421             1080.0
..                 …
59              1080.0
374             1080.0
493             1080.0
79              1800.0
465             1080.0

[170 rows x 6 columns],
627    11.418516
892    11.790489
859    10.491246
169    10.877877
630    10.239638
         …
290    10.950649
663    10.274741
648    10.760050
651    10.596410
218    11.018482
Name: price, Length: 679, dtype: float64,
126    11.198091
368    11.918331
382    10.571086
363    12.019689
```

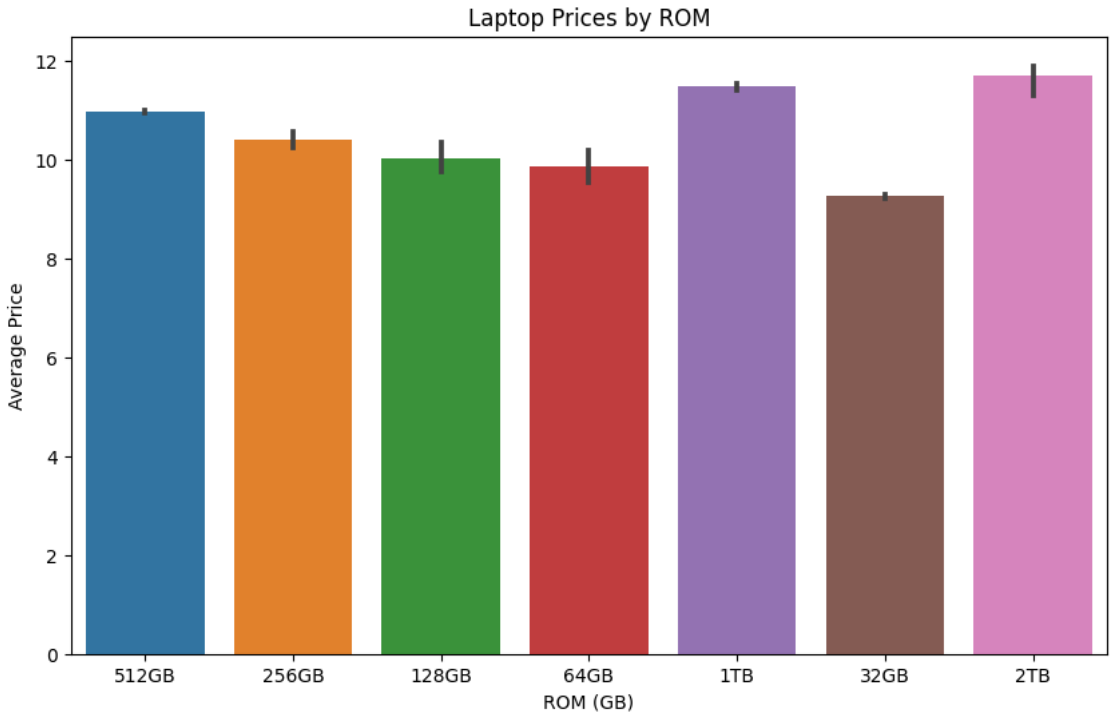
```
421    11.682668
         …
59     11.361998
374    10.463103
493    11.074281
79     11.440355
465    10.275086
Name: price, Length: 170, dtype: float64)
```

[66]:
```
# Exploratory Data Analysis (EDA)
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

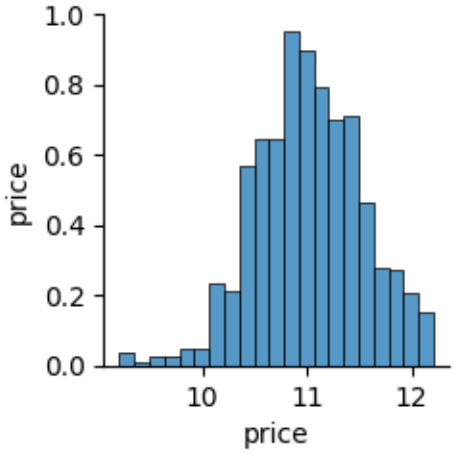
```
<ipython-input-66-07889fbf6bba>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df.corr()
```



[67]:
```
# Exploratory Data Analysis (EDA)
# Visualize the distribution of laptop prices based on storage capacity
plt.figure(figsize=(10, 6))
sns.barplot(x='ROM', y='price', data=df)
plt.title('Laptop Prices by ROM')
plt.xlabel('ROM (GB)')
plt.ylabel('Average Price')
plt.show()
```



[68]:
```
# Visualizations
sns.pairplot(df[['processor', 'Ram', 'ROM', 'price']])
plt.show()
```



[69]:
```
# Statistical Analysis
# T-test example comparing prices of laptops with 8GB RAM and 16GB RAM
from scipy.stats import ttest_ind
```

```
ram_8gb_prices = df[df['display_size'] == 14]['price']
ram_16gb_prices = df[df['spec_rating'] == 60]['price']
t_stat, p_value = ttest_ind(ram_8gb_prices, ram_16gb_prices)
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

```
T-statistic: 2.1960595518281667, P-value: 0.0289814645582138
```

[70]:
```
# Modeling
model = LinearRegression()
model.fit(X_train, y_train)
print("Executed...AJ")
```

```
Executed…AJ
```

[78]:
```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'Prediction is precise tends to the precision of {mse} fraction.')
print("Executed...Amreen has Successfully Completed Analysis...")
```

```
Mean Squared Error: 0.15447466499287116
Prediction is precise tends to the precision of 0.15447466499287116 fraction.
Executed…Amreen has Successfully Completed Analysis…
```

0.1 *QUESTIONS:*

Question1: Retrieve the processor speed and corresponding laptop prices for all entries in the dataset.

[72]:
```
#Fetch Processor Information:
processor_info = df[['processor', 'price']]
processor_info
```

[72]:

	processor	price
0	5th Gen AMD Ryzen 5 5600H	10.817796
1	12th Gen Intel Core i3 1215U	10.594157
2	11th Gen Intel Core i3 1115G4	10.203259
3	12th Gen Intel Core i5 1240P	10.997590
4	Apple M1	11.156122
..	…	…
888	13th Gen Intel Core i3 1315U	10.714218
889	6th Gen AMD Ryzen 7 6800H	11.608245
890	7th Gen AMD Ryzen 9 7940HS	12.154732
891	13th Gen Intel Core i7 13700H	11.775220
892	7th Gen AMD Ryzen 9 7940HS	11.790489

[849 rows x 2 columns]

Question2: Find the average price of laptops with 16 GB of RAM.

[73]:
```
average_price_16gb = df[df['Ram'] == '16GB']['price'].mean()
average_price_16gb
```

[73]:
```
11.298230957218056
```

Question3: Identify laptops with prices above 2000 as potential outliers.

[74]:
```
potential_outliers = df[df['price'] > 2000]
potential_outliers
```

[74]:
```
Empty DataFrame
Columns: [Unnamed: 0.1, Unnamed: 0, brand, name, price, spec_rating, processor,
CPU, Ram, Ram_type, ROM, ROM_type, GPU, display_size, resolution_width,
resolution_height, OS, warranty]
Index: []
```

Question4: Retrieve the actual prices, predicted prices, and the absolute difference for laptops in the test set.

[75]:
```
predictions_df = pd.DataFrame({'Actual_Prices': y_test, 'Predicted_Prices':␣
↪model.predict(X_test)})
predictions_df['Price_Difference'] = abs(predictions_df['Actual_Prices'] -␣
↪predictions_df['Predicted_Prices'])
predictions_df['Price_Difference']
```

[75]:
```
126    0.394831
368    0.220900
382    0.297225
363    0.296415
421    0.816063
         …
59     0.298411
374    0.148410
493    0.087767
79     0.160576
465    0.703025
Name: Price_Difference, Length: 170, dtype: float64
```

Question5: Show the details of laptops with prices below 10, ordered by Display Size in descending order.

[77]:
```
low_cost_laptops = df[df['price'] < 10].sort_values(by='display_size',␣
↪ascending=False)
low_cost_laptops
```

[77]:
```
     Unnamed: 0.1  Unnamed: 0      brand                                  name  \
351           370         411         HP       Chromebook 15a-na0012TU Laptop
497           521         589     Lenovo            V15-IGL 82C3A008IH Laptop
```

```
703           737         821       Asus  Vivobook Go 15 E510MA-EJ001W Laptop
837           875         963        AXL                Vayu Book LAP02 Laptop
26             27          28         HP                 255 G9 840T7PA Laptop
291           310         347     Walker                       NU14A1 Laptop
540           564         644    Ultimus         Lite NU14U4INC44BN-CS Laptop
596           623         704    Ultimus             Neo NU14U3INT54BN Laptop
836           874         962        AXL                Vayu Book LAP01 Laptop
11             11          11    Ultimus          Pro NU14U3INC43BN-CS Laptop
214           223         252      Avita                   Pura NS14A6 Laptop
101           103         106     Lenovo                        E41-55 Laptop
631           663         745       Asus   Chromebook CX1400CKA-EK0335 Laptop
105           107         110      iBall            Excelance CompBook Laptop
110           113         117       Asus          CX1101CMA-GJ0004 Chromebook
239           252         284       Acer    One 11 Z8-284 UN.013SI.033 Laptop
13             13          13  Primebook                     4G Android Laptop
546           570         650       Acer    One 11 Z8-284 UN.013SI.032 Laptop
568           593         673       Acer  C734 NX.H8VSI.004 Chromebook Laptop

       price  spec_rating                     processor  \
351  9.679781    69.323529           Intel Celeron  N4500
497  9.903037    69.323529          Intel Celeron  N4020
703  9.998389    69.323529           intel Celeron  N4020
837  9.679781    69.323529           Intel Celeron  N4020
26   9.896010    69.323529       3rd Gen AMD Athlon 3050U
291  9.740439    69.323529           Intel Celeron  N4020
540  9.581283    69.323529           Intel Celeron  N4020
596  9.998389    69.323529  10th Gen Intel Core i3 1005G1
836  9.472012    69.323529           Intel Celeron  N4020
11   9.304832    69.323529           Intel Celeron  N4020
214  9.998389    69.323529     3rd Gen AMD Ryzen 5  3500U
101  9.851720    69.323529           AMD Athlon Pro 3045B
631  9.615205    69.323529            Intel Celeron N4500
105  9.210340    69.323529    Intel Atom Quad Core Z3735F
110  9.304832    69.323529           Intel Celeron  N4020
239  9.797627    69.323529            Intel Celeron N4500
13   9.304832    69.323529               MediaTek MTK8788
546  9.851720    69.323529            Intel Celeron N4500
568  9.546170    69.323529  Intel Celeron Dual Core N4500

                     CPU  Ram Ram_type    ROM   ROM_type  \
351  Dual Core, 2 Threads  4GB  LPDDR4x  128GB  Hard-Disk
497  Dual Core, 2 Threads  4GB     DDR4  256GB        SSD
703  Dual Core, 2 Threads  4GB     DDR4  256GB        SSD
837  Dual Core, 2 Threads  4GB     DDR4  256GB        SSD
26   Dual Core, 2 Threads  4GB     DDR4  256GB        SSD
291  Dual Core, 2 Threads  4GB     DDR4  128GB        SSD
540  Dual Core, 2 Threads  4GB     DDR4  256GB        SSD
```

```
596  Dual Core, 4 Threads  8GB     DDR4  256GB        SSD
836  Dual Core, 2 Threads  4GB     DDR4  128GB        SSD
11   Dual Core, 2 Threads  4GB     DDR4  128GB        SSD
214  Quad Core, 8 Threads  8GB     DDR4  512GB        SSD
101  Dual Core, 2 Threads  8GB     DDR4  256GB        SSD
631  Dual Core, 2 Threads  4GB  LPDDR4X  128GB        SSD
105             Quad Core  2GB     DDR3   32GB  Hard-Disk
110  Dual Core, 2 Threads  4GB   LPDDR4   32GB  Hard-Disk
239  Dual Core, 2 Threads  8GB     DDR4  128GB        SSD
13              Octa Core  4GB   LPDDR4   64GB  Hard-Disk
546  Dual Core, 2 Threads  8GB     DDR4  256GB        SSD
568  Dual Core, 2 Threads  4GB  LPDDR4X   64GB  Hard-Disk

                             GPU  display_size  resolution_width  \
351  Intel Integrated UHD Graphics          15.6            1366.0
497           Intel Integrated UHD          15.6            1920.0
703  Integrated Intel UHD Graphics          15.6            1920.0
837  Intel Integrated UHD Graphics          15.6            1920.0
26                  AMD Integrated          15.6            1366.0
291  Intel Integrated UHD Graphics          14.1            1920.0
540  Intel Integrated UHD Graphics          14.1            1366.0
596          Intel Integrated Iris          14.1            1920.0
836  Intel Integrated UHD Graphics          14.1            1920.0
11   Intel Integrated UHD Graphics          14.1            1366.0
214              AMD Radeon Vega 8          14.0            1920.0
101            AMD Radeon Graphics          14.0            1920.0
631           Intel Integrated UHD          14.0            1920.0
105              Intel HD Graphics          11.6            1366.0
110         Intel UHD Graphics 600          11.6            1366.0
239           Intel Integrated UHD          11.6            1366.0
13                    ARM Mali G72          11.6            1366.0
546           Intel Integrated UHD          11.6            1366.0
568           Intel Integrated UHD          11.6            1280.0

     resolution_height             OS  warranty
351              768.0      Chrome OS         1
497             1080.0         DOS OS         1
703             1080.0  Windows 11 OS         1
837             1080.0  Windows 11 OS         1
26               768.0         DOS OS         1
291             1080.0  Windows 11 OS         1
540              768.0  Windows 11 OS         1
596             1080.0  Windows 11 OS         1
836             1080.0  Windows 11 OS         1
11               768.0  Windows 11 OS         1
214             1080.0  Windows 10 OS         1
101             1080.0  Windows 11 OS         1
```

631	1080.0	Chrome OS	1
105	768.0	Windows 10 OS	1
110	768.0	Chrome OS	1
239	768.0	Windows 11 OS	1
13	768.0	Android 11 OS	1
546	768.0	Windows 11 OS	1
568	1024.0	Chrome OS	1

winedata-analysis

December 11, 2023

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
```

1 Data Pre-Processing

[2]:
```
df = pd.read_csv('WineQT.csv')
```

[3]:
```
df
```

[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
…	…	…	…	…	…	…	…	…	…	…
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71

```
      alcohol  quality    Id
0         9.4        5     0
1         9.8        5     1
2         9.8        5     2
3         9.8        6     3
4         9.4        5     4
…         …        …    …
1138     11.0        6  1592
1139      9.5        6  1593
1140     10.5        5  1594
1141     11.2        6  1595
1142     10.2        5  1597

[1143 rows x 13 columns]
```

[4]:
```
df.head()
```

[4]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality  Id
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
```

[5]:
```
df.tail()
```

[5]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
1138            6.3             0.510         0.13             2.3      0.076
1139            6.8             0.620         0.08             1.9      0.068
1140            6.2             0.600         0.08             2.0      0.090
1141            5.9             0.550         0.10             2.2      0.062
1142            5.9             0.645         0.12             2.0      0.075
```

```
      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
1138                 29.0                  40.0  0.99574  3.42       0.75
1139                 28.0                  38.0  0.99651  3.42       0.82
1140                 32.0                  44.0  0.99490  3.45       0.58
1141                 39.0                  51.0  0.99512  3.52       0.76
1142                 32.0                  44.0  0.99547  3.57       0.71

      alcohol  quality    Id
1138     11.0        6  1592
1139      9.5        6  1593
1140     10.5        5  1594
1141     11.2        6  1595
1142     10.2        5  1597
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64
 12  Id                    1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

[7]:
```
df.describe()
```

[7]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1143.000000       1143.000000  1143.000000     1143.000000
mean        8.311111          0.531339     0.268364        2.532152
std         1.747595          0.179633     0.196686        1.355917
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.392500     0.090000        1.900000
50%         7.900000          0.520000     0.250000        2.200000
```

```
75%         9.100000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1143.000000          1143.000000           1143.000000  1143.000000
mean      0.086933            15.615486             45.914698     0.996730
std       0.047267            10.250486             32.782130     0.001925
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             21.000000     0.995570
50%       0.079000            13.000000             37.000000     0.996680
75%       0.090000            21.000000             61.000000     0.997845
max       0.611000            68.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality           Id
count  1143.000000  1143.000000  1143.000000  1143.000000  1143.000000
mean      3.311015     0.657708    10.442111     5.657043   804.969379
std       0.156664     0.170399     1.082196     0.805824   463.997116
min       2.740000     0.330000     8.400000     3.000000     0.000000
25%       3.205000     0.550000     9.500000     5.000000   411.000000
50%       3.310000     0.620000    10.200000     6.000000   794.000000
75%       3.400000     0.730000    11.100000     6.000000  1209.500000
max       4.010000     2.000000    14.900000     8.000000  1597.000000
```

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

[8]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
Id                        int64
dtype: object
```

[9]:
```
df.size
```

[9]:
```
14859
```

[10]:
```
df.shape
```

[10]:
```
(1143, 13)
```

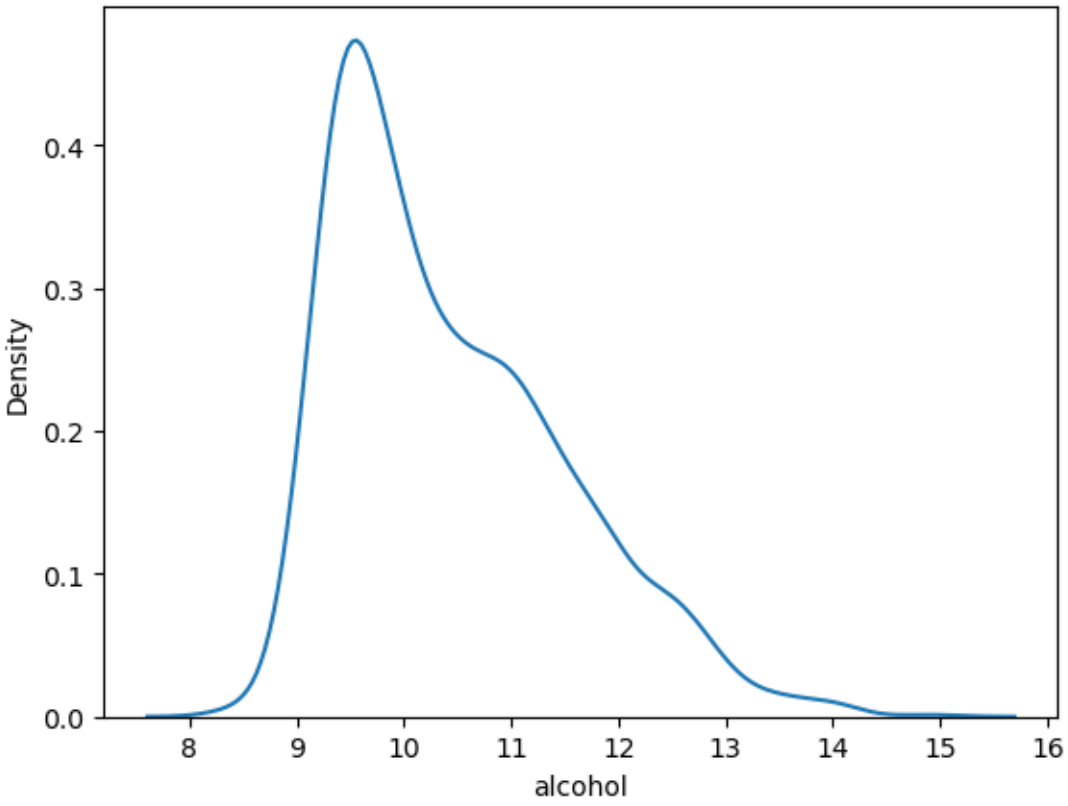
[11]:
```
df.columns
```

[11]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

2 How to visualize the single variable 'alcohol' to visualize its distribution?

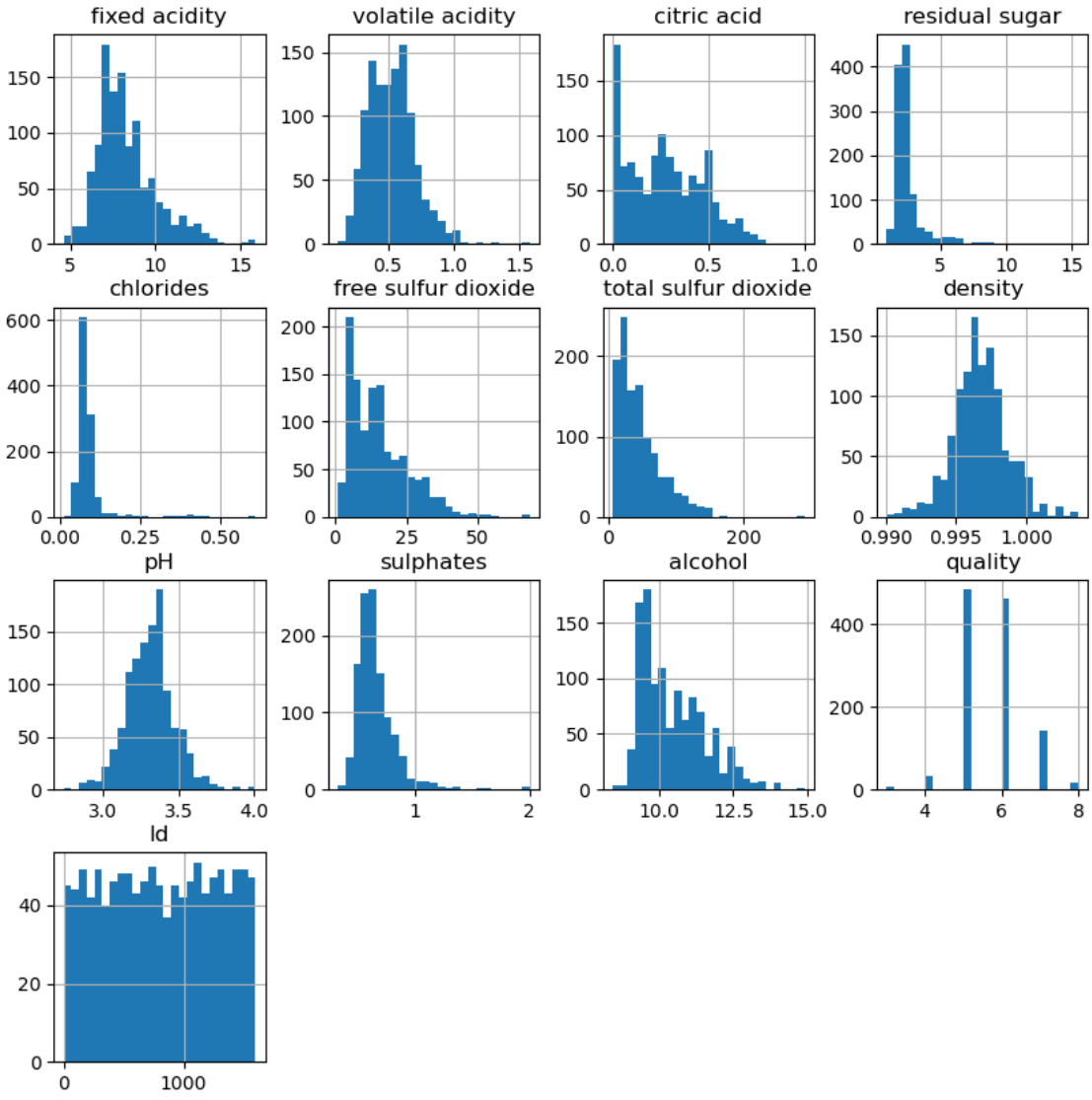
[12]:
```
sns.kdeplot(df['alcohol'])
```

[12]:
```
<Axes: xlabel='alcohol', ylabel='Density'>
```



3 Display histogram for all feature variables.

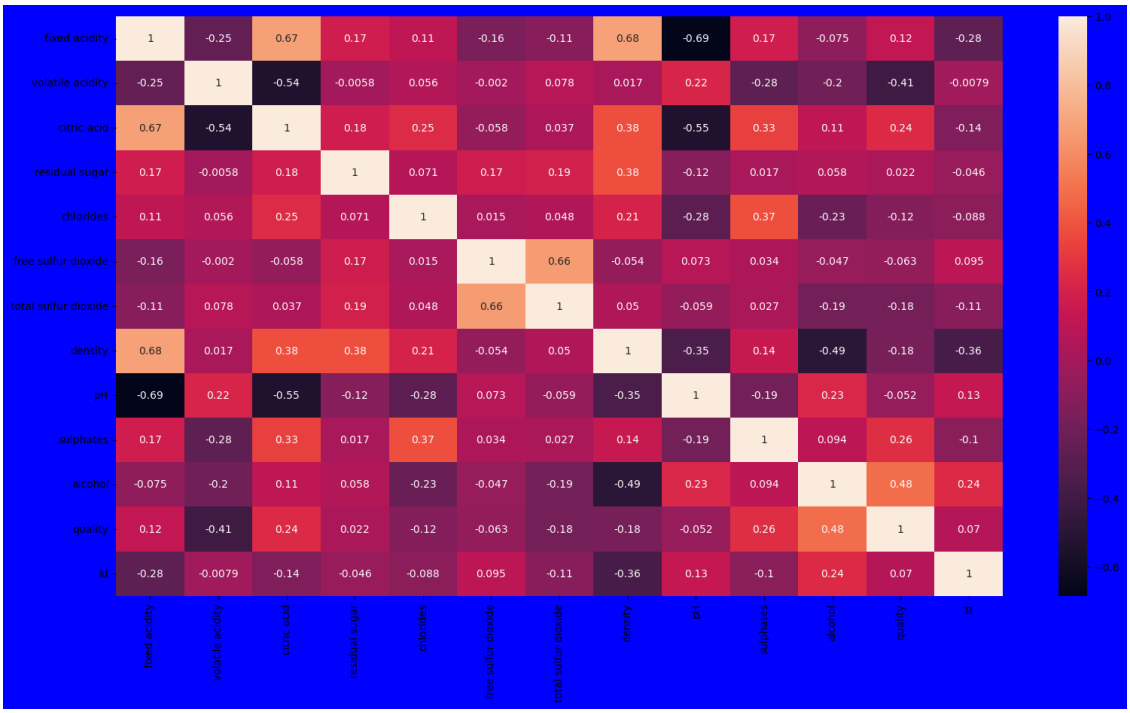
[14]:
```
df.hist(bins=25,figsize=(10,10))
plt.show()
```



4 Use a statistical method that finds the bonding and relationship between two features.

[15]:
```
# ploting heatmap
plt.figure(figsize=[19,10],facecolor='blue')
sns.heatmap(df.corr(),annot=True)
```

[15]: <Axes: >



5 What is the average value of the 'alcohol' column?

[16]:
```
average_alcohol = df['alcohol'].mean()
print(f"The average alcohol content is: {average_alcohol}")
```

```
The average alcohol content is: 10.442111402741325
```

6 What is the correlation between 'fixed acidity' and 'pH'?

[17]:
```
correlation_acidity_pH = df['fixed acidity'].corr(df['pH'])
print(f"The correlation between fixed acidity and pH is:␣
  ↪{correlation_acidity_pH}")
```

```
The correlation between fixed acidity and pH is: -0.685162598823547
```

7 How many records have 'density' greater than 0.998?

[18]:
```
records_above_density_threshold = df[df['density'] > 0.998].shape[0]
print(f"There are {records_above_density_threshold} records with density␣
  ↪greater than 0.998.")
```

```
There are 250 records with density greater than 0.998.
```

netflix-tv-shows

December 11, 2023

[1]:
```
import numpy as np
import pandas as pd
data = pd.read_csv("C:/Users/ahap0/Downloads/Netflix TV Shows and Movies.csv")
```

[2]:
```
data.head()
```

[2]:
```
   index        id                              title   type  \
0      0   tm84618                        Taxi Driver  MOVIE
1      1  tm127384  Monty Python and the Holy Grail  MOVIE
2      2   tm70993                      Life of Brian  MOVIE
3      3  tm190788                       The Exorcist  MOVIE
4      4   ts22164     Monty Python's Flying Circus   SHOW

                                         description  release_year  \
0  A mentally unstable Vietnam War veteran works …          1976
1  King Arthur, accompanied by his squire, recrui…          1975
2  Brian Cohen is an average young Jewish man, bu…          1979
3  12-year-old Regan MacNeil begins to adapt an e…          1973
4  A British sketch comedy series with the shows …          1969

  age_certification  runtime    imdb_id  imdb_score  imdb_votes
0                 R      113  tt0075314         8.3    795222.0
1                PG       91  tt0071853         8.2    530877.0
2                 R       94  tt0079470         8.0    392419.0
3                 R      133  tt0070047         8.1    391942.0
4             TV-14       30  tt0063929         8.8     72895.0
```

Q-1. What is the range of release years for the movies and shows, and can you identify any patterns or trends in the distribution of release years?

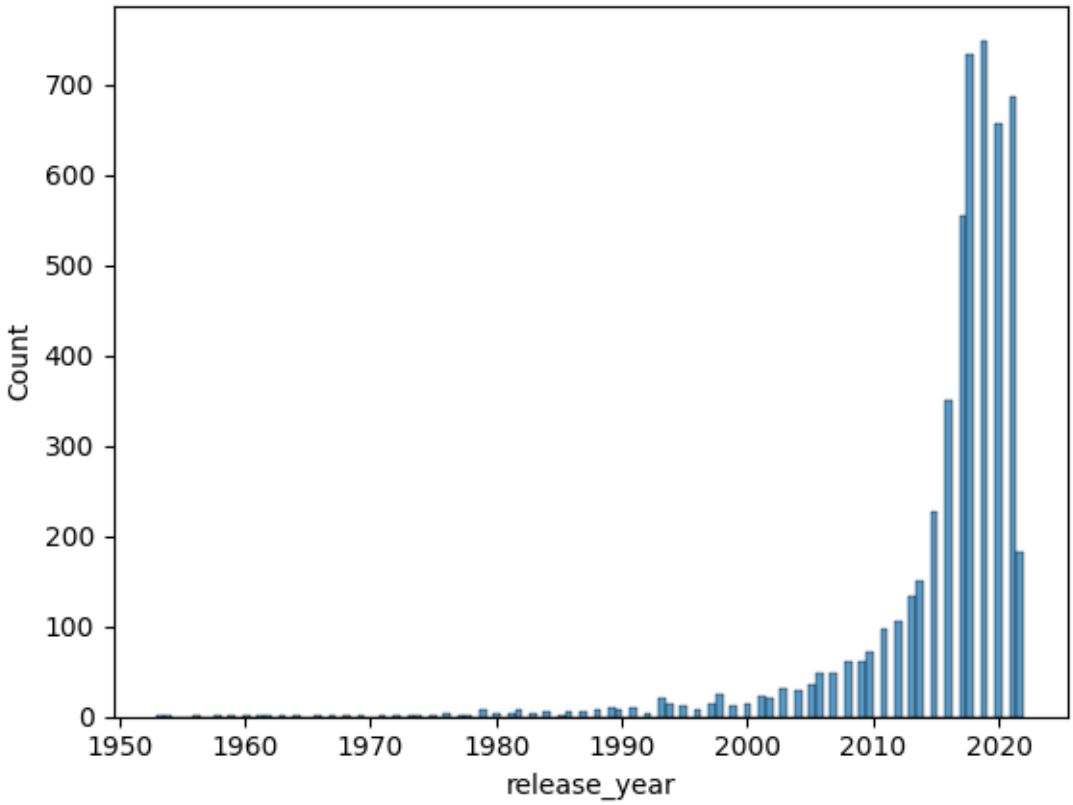
[6]:
```
data['release_year'].describe()
```

[6]:
```
count    5283.000000
mean     2015.879992
std         7.346098
min      1953.000000
25%      2015.000000
50%      2018.000000
```

```
75%      2020.000000
max      2022.000000
Name: release_year, dtype: float64
```

[7]:
```
import seaborn as sns
sns.histplot(data = data, x='release_year')
```

[7]:
```
<Axes: xlabel='release_year', ylabel='Count'>
```



answer from graph: between 2010 to 2020 the titles produced increased drastically

Q-2. What is the average IMDb score for the movies and shows, and can you identify the top 3 movies with the highest IMDb scores

[8]:
```
avg_movie_score = data[data["type"]=="MOVIE"]["imdb_score"].mean()
avg_show_score = data[data["type"]=="SHOW"]["imdb_score"].mean()
print(f"Average imdb score for movies, shows is {avg_movie_score} and␣
 ↪{avg_show_score} respectively.\n")

print("Below are the top 3 movies with highest imdb score:")
data[data["type"]=="MOVIE"].sort_values(by="imdb_score", ascending=False)[:3]
```

Average imdb score for movies, shows is 6.266979747578516 and 7.017377398720683 respectively.

Below are the top 3 movies with highest imdb score:

[8]:
```
                                        title   type  \
3172  David Attenborough: A Life on Our Planet  MOVIE
2685                          C/o Kancharapalem  MOVIE
24                               No Longer Kids  MOVIE

                                            description  release_year  \
3172  The story of life on our planet by the man who…          2020
2685  From a schoolboyâ€ s crush to a middle-aged ba…          2018
24    By coincidence, Ahmad discovers that his fathe…          1979

     age_certification  runtime  imdb_score  imdb_votes
3172                PG       83         9.0     31180.0
2685                PG      152         9.0      6562.0
24                 NaN      235         9.0       943.0
```

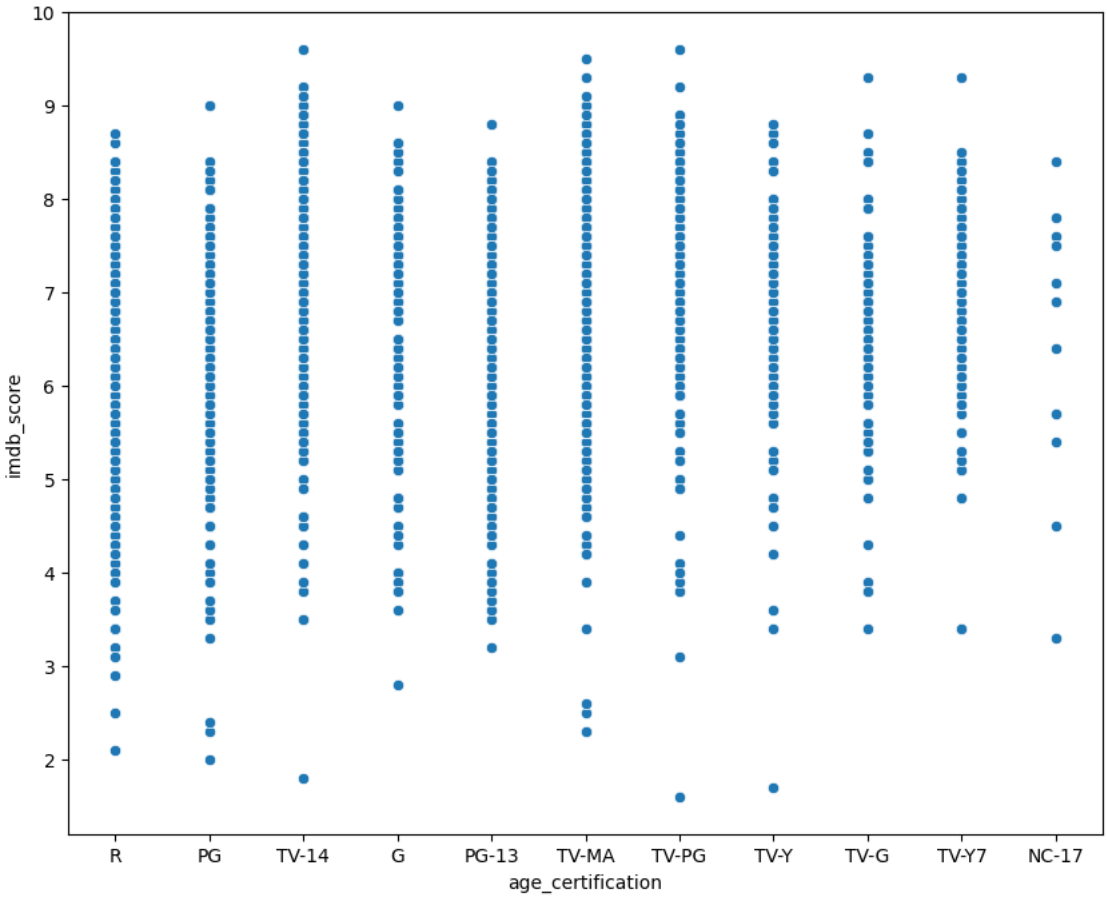
Q-3. What are the unique age certifications present in the dataset, and can you identify any relationship between IMDb score and age certification?

[9]:
```
data["age_certification"].unique()
```

[9]:
```
array(['R', 'PG', 'TV-14', 'G', 'PG-13', nan, 'TV-MA', 'TV-PG', 'TV-Y',
       'TV-G', 'TV-Y7', 'NC-17'], dtype=object)
```

[10]:
```
import matplotlib.pyplot as plt
plt.figure(figsize = (10,8))
sns.scatterplot(data =data, x='age_certification', y='imdb_score')
```

[10]:
```
<Axes: xlabel='age_certification', ylabel='imdb_score'>
```

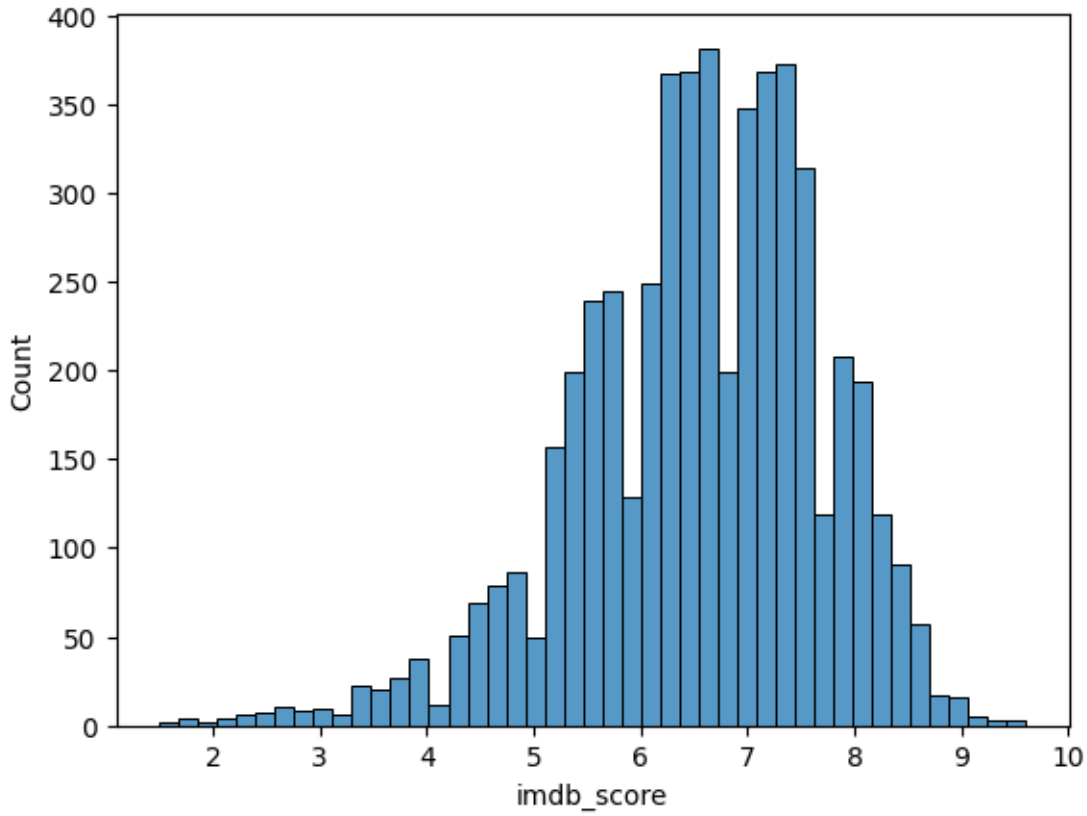


answer: the highest imdb scores went to tv-14 and tv-ma rating. The lowest went to pg rating. The ratings on TV-MA are wisespread

Q-4. Analyze the distribution of IMDb votes, and investigate if there is a correlation between the number of IMDb votes and the IMDb score.

[12]:
```
sns.histplot(data = data, x='imdb_score')
```

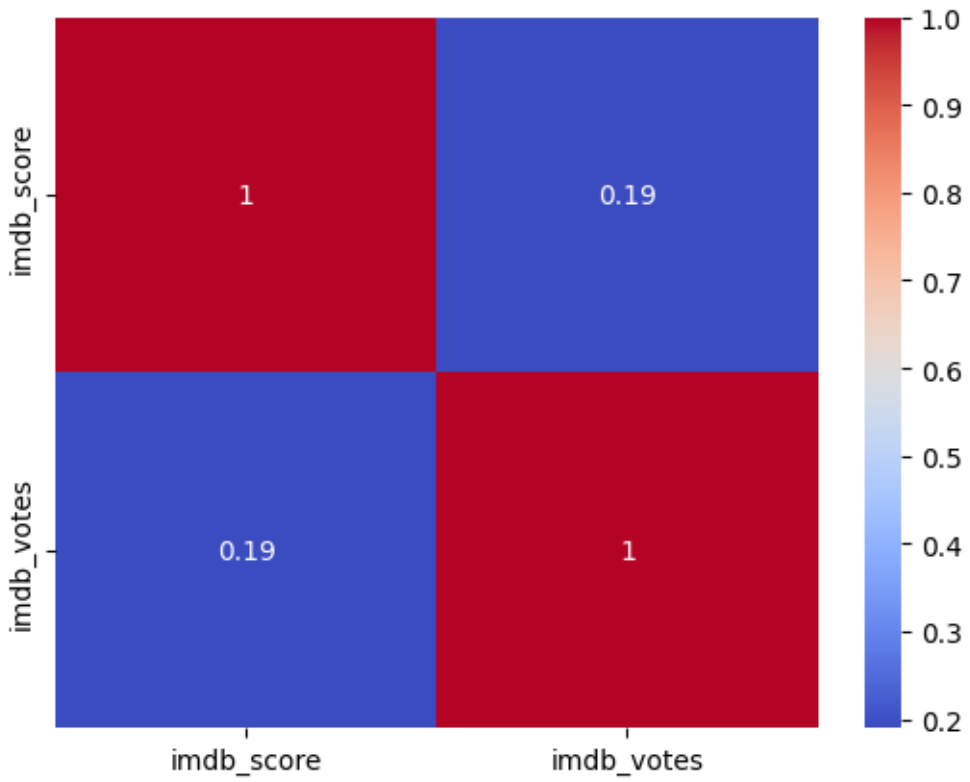
[12]:
```
<Axes: xlabel='imdb_score', ylabel='Count'>
```



answer1: most imdb rating is between 6 to 7 range

[13]:
```
df_temp = data[["imdb_score", "imdb_votes"]]
sns.heatmap( df_temp.corr(), cmap='coolwarm', annot=True)
```

[13]: <Axes: >

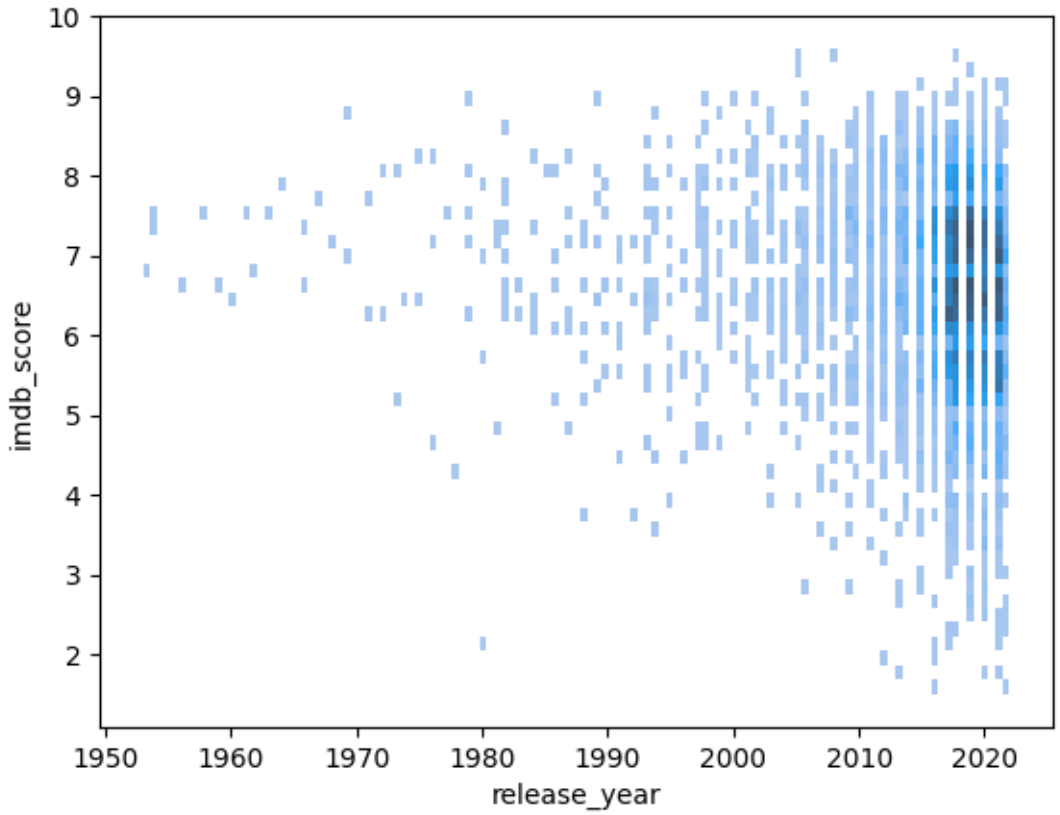


answer2: the correlation between them is 0.19, which is a weak positive corelation

Q-5. Are there any noticeable trends in IMDb scores over the years, and how would you visualize the popularity of movies and shows over time?

[14]:
```
sns.histplot(data=data, x='release_year', y='imdb_score')
```

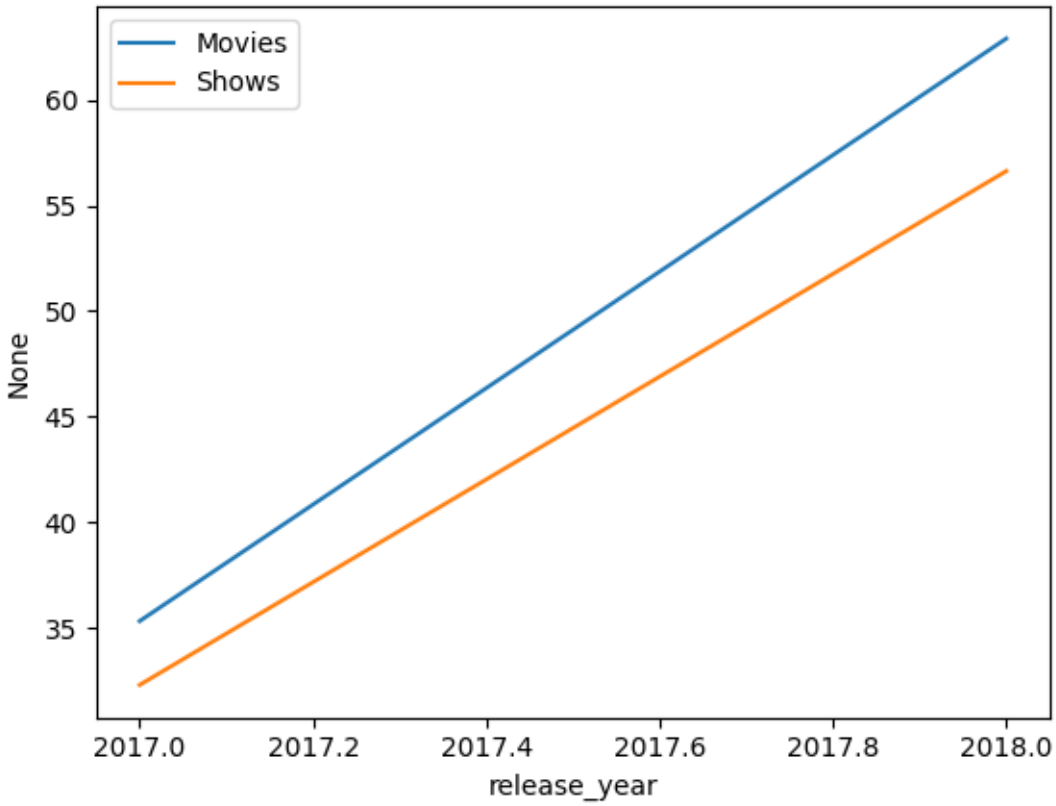
[14]:
```
<Axes: xlabel='release_year', ylabel='imdb_score'>
```



answer: from the graph we can see that there are more imdb ratings recently. The reduction of gadgets cost because of mass -production, the availability of internet explains this higher number of the imdb ratings in the recent years.

[17]:
```
# Visualize the popularity of movies and shows over time
sns.lineplot(data=data, x='release_year', y=data[data['type'] == 'MOVIE'].
 ↪groupby('release_year').size(), label='Movies',errorbar=None)
sns.lineplot(data=data, x='release_year', y=data[data['type'] == 'SHOW'].
 ↪groupby('release_year').size(), label='Shows',errorbar=None)
```

[17]:
```
<Axes: xlabel='release_year', ylabel='None'>
```


Movies
Shows
60
55
50
45
40
35
None
2017.0
2017.2
2017.4
2017.6
2017.8
2018.0
release_year

[ ]:

used-cars

December 11, 2023

1 About the dataset

Used Cars-Data Analysis

Used Car Prices in UK Dataset is a comprehensive collection of automotive information extracted from the popular automotive marketplace website, autotrader.co.uk. This dataset comprises 3,685 data points, each representing a unique vehicle listing, and includes thirteen distinct features providing valuable insights into the world of automobiles. The feature names are: title,Price : price of car in pounds,Mileage(miles),Registration(year),Previous Owners,Fuel Type,Body Type,Engine,Gearbox,Seats,Doors,Emission Class,Service history

#Importing important libraries

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[ ]:
```
df=pd.read_csv('/content/used_cars_UK.csv')
```

[ ]:
```
df.head(10)
```

[ ]:

	Unnamed: 0	title	Price	Mileage(miles)	Registration_Year	\
0	0	SKODA Fabia	6900	70189	2016	
1	1	Vauxhall Corsa	1495	88585	2008	
2	2	Hyundai i30	949	137000	2011	
3	3	MINI Hatch	2395	96731	2010	
4	4	Vauxhall Corsa	1000	85000	2013	
5	5	Hyundai Coupe	800	124196	2007	
6	6	Ford Focus	798	140599	2008	
7	7	Vauxhall Corsa	1995	90000	2009	
8	8	Volvo 740	750	225318	1989	
9	9	Peugeot 207	1299	87000	2008	

	Previous Owners	Fuel type	Body type	Engine	Gearbox	Doors	Seats	\
0	3.0	Diesel	Hatchback	1.4L	Manual	5.0	5.0	

```
1              4.0    Petrol  Hatchback   1.2L     Manual    3.0    5.0
2              NaN    Petrol  Hatchback   1.4L     Manual    5.0    5.0
3              5.0    Petrol  Hatchback   1.4L     Manual    3.0    4.0
4              NaN    Diesel  Hatchback   1.3L     Manual    5.0    5.0
5              3.0    Petrol      Coupe   2.0L     Manual    3.0    4.0
6              NaN    Petrol  Hatchback   1.6L     Manual    5.0    5.0
7              NaN    Petrol  Hatchback   1.2L     Manual    3.0    5.0
8              NaN    Petrol     Estate   2.3L  Automatic    5.0    NaN
9              5.0    Diesel  Hatchback   1.6L     Manual    5.0    5.0

  Emission Class Service history
0         Euro 6             NaN
1         Euro 4            Full
2         Euro 5             NaN
3         Euro 4            Full
4         Euro 5             NaN
5         Euro 4             NaN
6         Euro 4             NaN
7         Euro 4             NaN
8            NaN             NaN
9         Euro 4             NaN
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3685 entries, 0 to 3684
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         3685 non-null   int64
 1   title              3685 non-null   object
 2   Price              3685 non-null   int64
 3   Mileage(miles)     3685 non-null   int64
 4   Registration_Year  3685 non-null   int64
 5   Previous Owners    2276 non-null   float64
 6   Fuel type          3685 non-null   object
 7   Body type          3685 non-null   object
 8   Engine             3640 non-null   object
 9   Gearbox            3685 non-null   object
 10  Doors              3660 non-null   float64
 11  Seats              3650 non-null   float64
 12  Emission Class     3598 non-null   object
 13  Service history    540 non-null    object
dtypes: float64(3), int64(4), object(7)
memory usage: 403.2+ KB
```

[ ]:
```
df.describe(include='all')
```

[ ]:

	Unnamed: 0	title	Price	Mileage(miles)	\
count	3685.000000	3685	3685.000000	3.685000e+03	
unique	NaN	469	NaN	NaN	
top	NaN	Vauxhall Corsa	NaN	NaN	
freq	NaN	223	NaN	NaN	
mean	2314.770963	NaN	5787.145726	8.132816e+04	
std	1415.821308	NaN	4480.810572	3.942083e+04	
min	0.000000	NaN	400.000000	1.000000e+00	
25%	1059.000000	NaN	2490.000000	5.698400e+04	
50%	2279.000000	NaN	4000.000000	8.000000e+04	
75%	3593.000000	NaN	7995.000000	1.030000e+05	
max	4727.000000	NaN	33900.000000	1.110100e+06	

	Registration_Year	Previous Owners	Fuel type	Body type	Engine	\
count	3685.000000	2276.000000	3685	3685	3640	
unique	NaN	NaN	6	10	34	
top	NaN	NaN	Petrol	Hatchback	1.6L	
freq	NaN	NaN	2361	2279	734	
mean	2011.835007	2.807557	NaN	NaN	NaN	
std	5.092566	1.546028	NaN	NaN	NaN	
min	1953.000000	1.000000	NaN	NaN	NaN	
25%	2008.000000	2.000000	NaN	NaN	NaN	
50%	2012.000000	3.000000	NaN	NaN	NaN	
75%	2015.000000	4.000000	NaN	NaN	NaN	
max	2023.000000	9.000000	NaN	NaN	NaN	

	Gearbox	Doors	Seats	Emission Class	Service history
count	3685	3660.000000	3650.000000	3598	540
unique	2	NaN	NaN	6	1
top	Manual	NaN	NaN	Euro 5	Full
freq	2868	NaN	NaN	1256	540
mean	NaN	4.321038	4.900274	NaN	NaN
std	NaN	0.986902	0.577200	NaN	NaN
min	NaN	2.000000	2.000000	NaN	NaN
25%	NaN	3.000000	5.000000	NaN	NaN
50%	NaN	5.000000	5.000000	NaN	NaN
75%	NaN	5.000000	5.000000	NaN	NaN
max	NaN	5.000000	7.000000	NaN	NaN

2 Data preprocessing

Setting index

[ ]:
```
df.set_index(["Unnamed: 0"],inplace=True)
```

[ ]:
```
df.columns
```

```
[ ]: Index(['title', 'Price', 'Mileage(miles)', 'Registration_Year',
             'Previous Owners', 'Fuel type', 'Body type', 'Engine', 'Gearbox',
             'Doors', 'Seats', 'Emission Class', 'Service history'],
            dtype='object')
```

Dropping columns

```
[ ]: df.drop(columns=['Service history'],inplace=True)
```

Filling missing values

```
[ ]: df['Previous Owners'].value_counts()
```

```
[ ]: 2.0    594
     1.0    523
     3.0    475
     4.0    360
     5.0    208
     6.0     60
     7.0     39
     8.0     12
     9.0      5
     Name: Previous Owners, dtype: int64
```

```
[ ]: df['Previous Owners'].fillna(2.0,inplace=True)
```

```
[ ]: df['Previous Owners'].value_counts()
```

```
[ ]: 2.0    2003
     1.0     523
     3.0     475
     4.0     360
     5.0     208
     6.0      60
     7.0      39
     8.0      12
     9.0       5
     Name: Previous Owners, dtype: int64
```

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3685 entries, 0 to 4727
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              3685 non-null   object
 1   Price              3685 non-null   int64
```

```
 2   Mileage(miles)     3685 non-null   int64
 3   Registration_Year  3685 non-null   int64
 4   Previous Owners    3685 non-null   float64
 5   Fuel type          3685 non-null   object
 6   Body type          3685 non-null   object
 7   Engine             3640 non-null   object
 8   Gearbox            3685 non-null   object
 9   Doors              3660 non-null   float64
 10  Seats              3650 non-null   float64
 11  Emission Class     3598 non-null   object
dtypes: float64(3), int64(3), object(6)
memory usage: 374.3+ KB
```

Dropping missing values

[ ]:
```
df=df.dropna()
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3591 entries, 0 to 4727
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              3591 non-null   object
 1   Price              3591 non-null   int64
 2   Mileage(miles)     3591 non-null   int64
 3   Registration_Year  3591 non-null   int64
 4   Previous Owners    3591 non-null   float64
 5   Fuel type          3591 non-null   object
 6   Body type          3591 non-null   object
 7   Engine             3591 non-null   object
 8   Gearbox            3591 non-null   object
 9   Doors              3591 non-null   float64
 10  Seats              3591 non-null   float64
 11  Emission Class     3591 non-null   object
dtypes: float64(3), int64(3), object(6)
memory usage: 364.7+ KB
```

3 Exploratory data analysis and visualisation

list the **Top 15 cars sold** & show the graph .

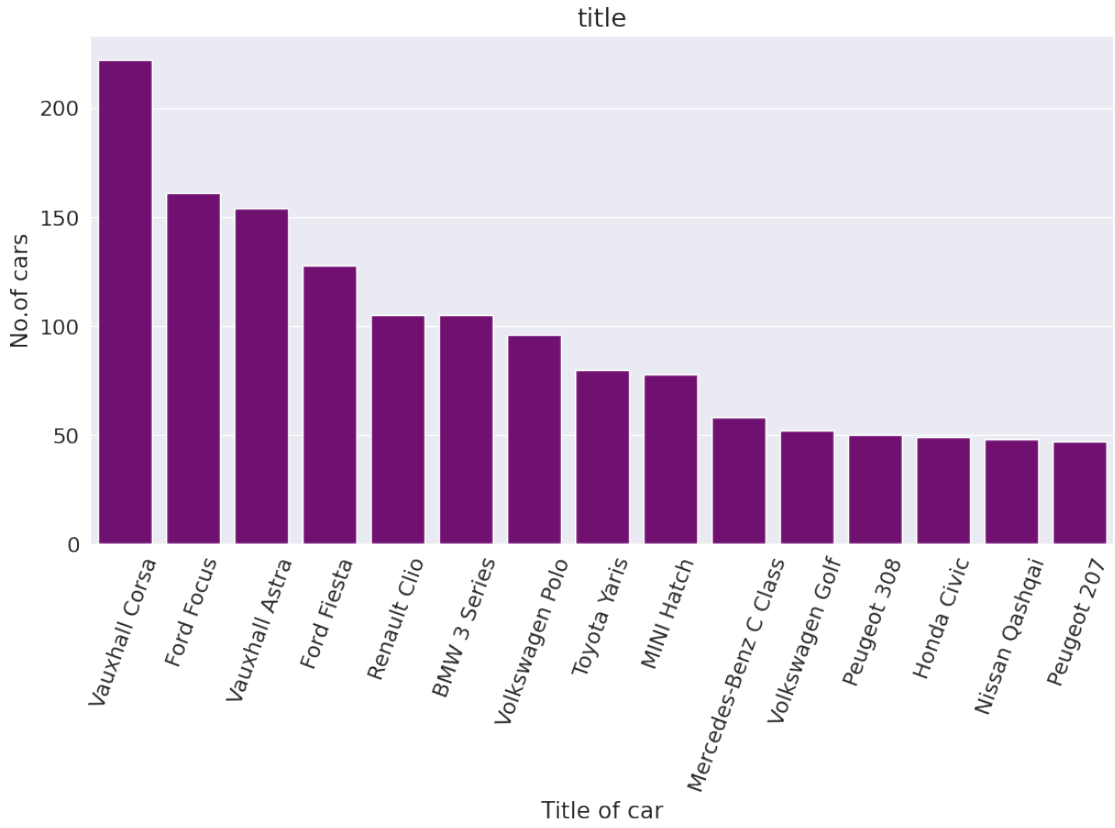
[ ]:
```
top_cars=df['title'].value_counts().head(15)
top_cars
```

[ ]:
```
Vauxhall Corsa          222
Ford Focus              161
```

```
Vauxhall Astra           154
Ford Fiesta              128
Renault Clio             105
BMW 3 Series             105
Volkswagen Polo           96
Toyota Yaris              80
MINI Hatch                78
Mercedes-Benz C Class     58
Volkswagen Golf           52
Peugeot 308               50
Honda Civic               49
Nissan Qashqai            48
Peugeot 207               47
Name: title, dtype: int64
```

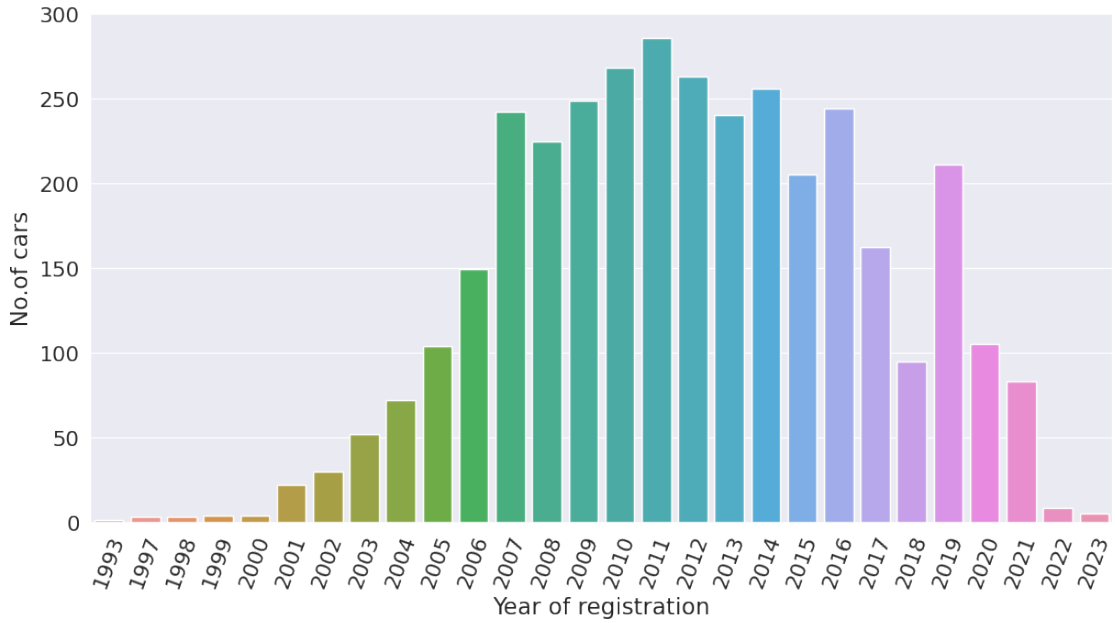
[ ]:
```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

[ ]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=70)
plt.title('title')
chart=sns.barplot(x=top_cars.index, y=top_cars,color='purple')
chart.set_ylabel('No.of cars', fontdict={'size': 15})
chart.set_xlabel('Title of car', fontdict={'size': 15});
```



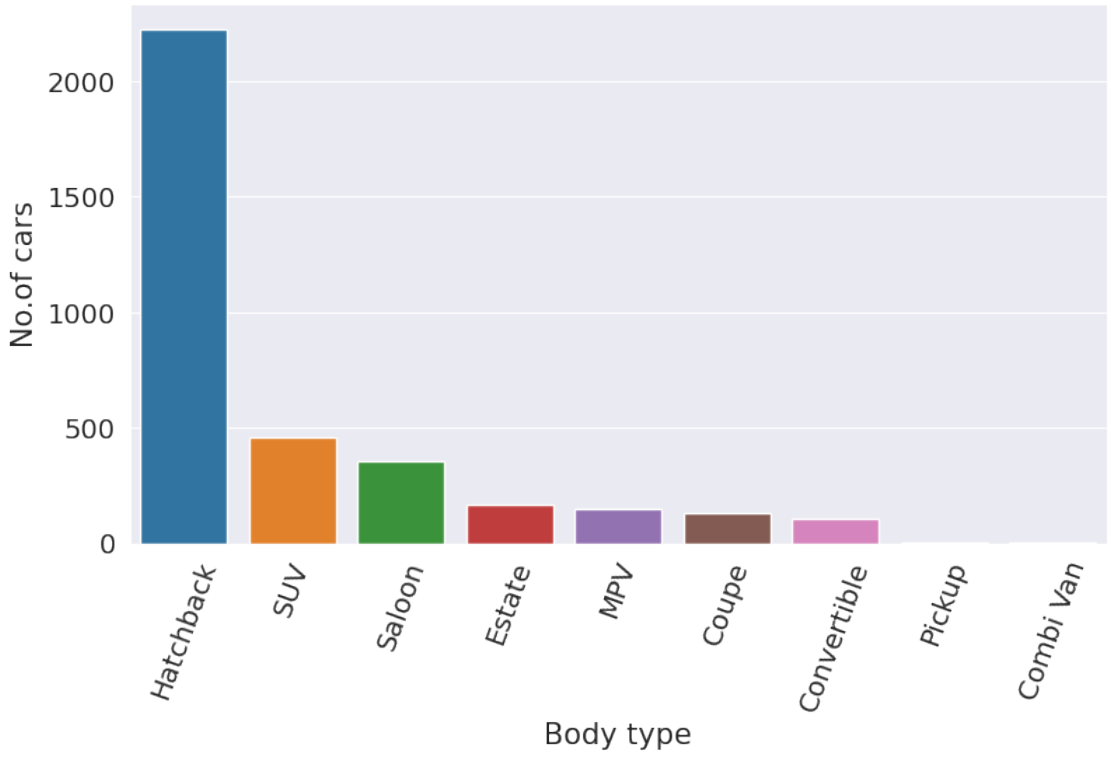
** Show the Year wise cars registered with the help of graph **

[ ]:
```
yearly_data=df['Registration_Year'].value_counts()
yearly_data
plt.figure(figsize=(12,6))
plt.xticks(rotation=70)
chart=sns.barplot(x=yearly_data.index, y=yearly_data)
chart.set_ylabel('No.of cars', fontdict={'size': 15})
chart.set_xlabel('Year of registration', fontdict={'size': 15});
```



Which is the Most popular body type? also represent with the graph.

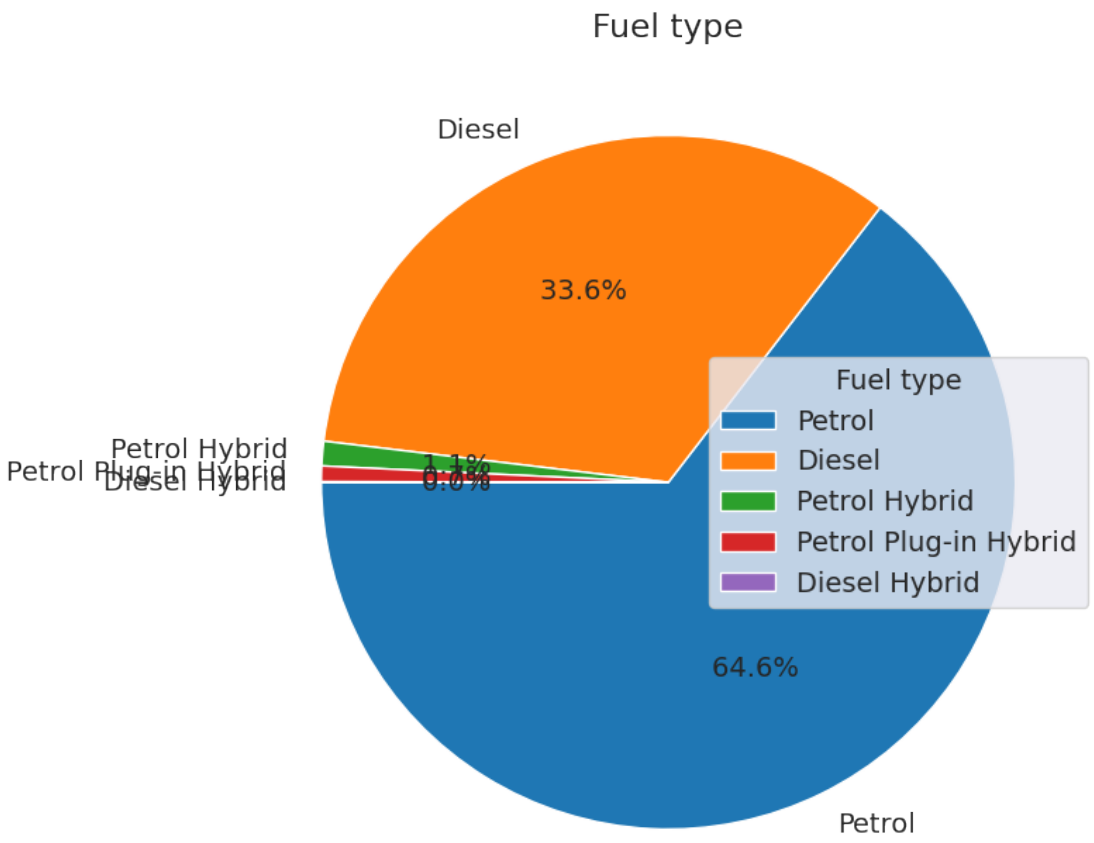
[ ]:
```
plt.xticks(rotation=70)
chart=sns.barplot(x=df['Body type'].value_counts().index, y=df['Body type'].
 ↪value_counts())
chart.set_ylabel('No.of cars', fontdict={'size': 15})
chart.set_xlabel('Body type', fontdict={'size': 15});
```



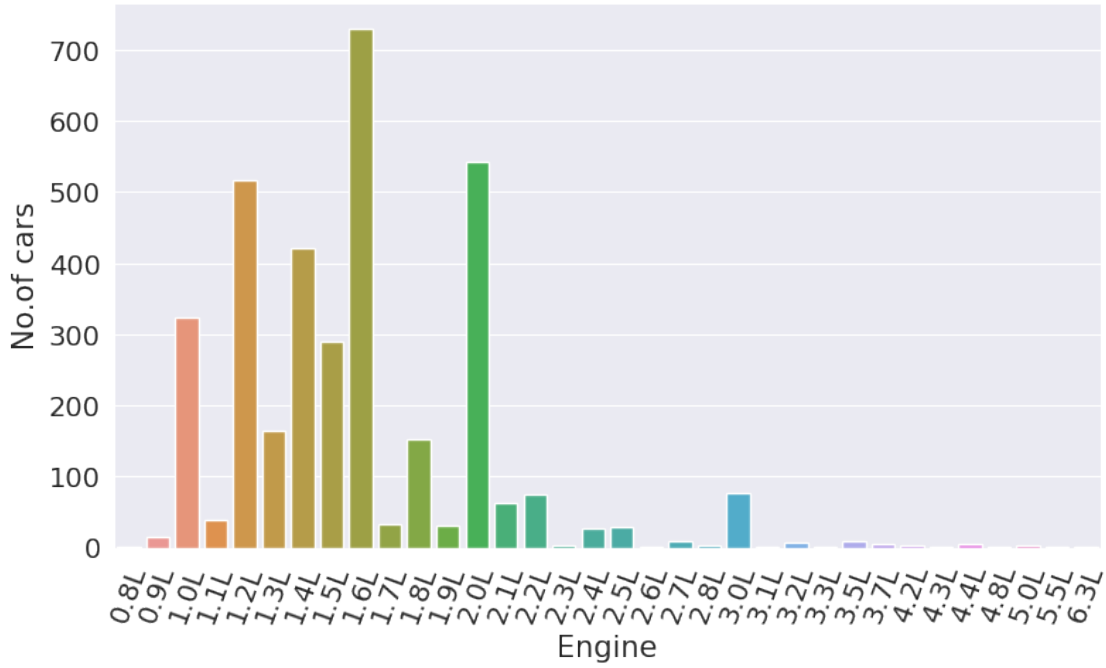
** What is the Most popular fuel type ? Represent with the pie chart & Graph. **

[ ]:
```
plt.figure(figsize=(8, 8))
#values=[value]
labels=['petrol','diesel','petrol hybrid','Petrol Plug-in␣
 ↪Hybrid','dieselhybrid']
plt.pie(df['Fuel type'].value_counts(), labels=df['Fuel type'].value_counts().
 ↪index, autopct='%1.1f%% ',startangle=180)
plt.title('Fuel type')
plt.legend(title='Fuel type')
plt.show
```

[ ]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

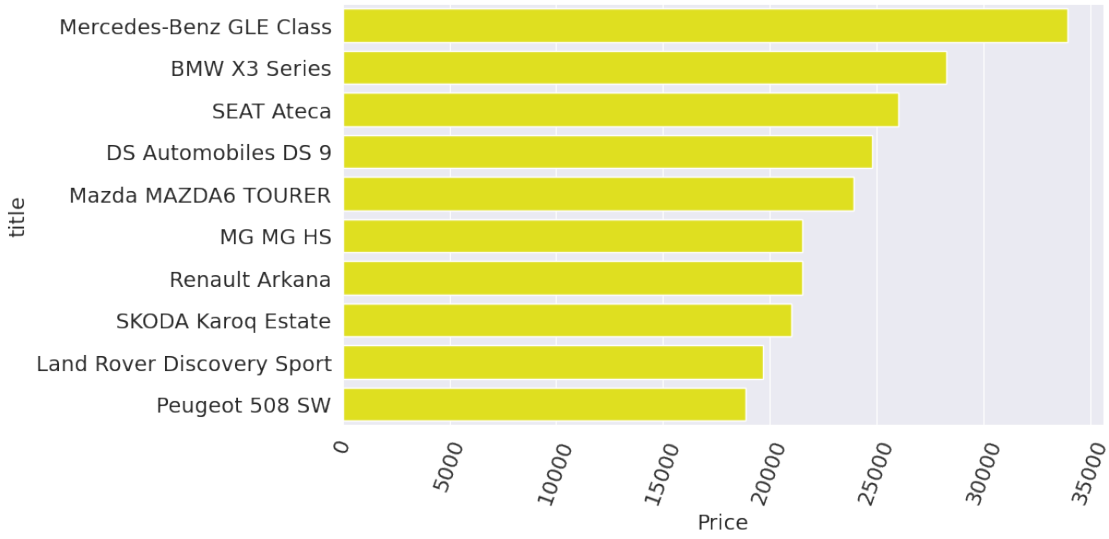


[ ]:
```
plt.xticks(rotation=70)
chart=sns.barplot(x=df['Engine'].value_counts().sort_index().index,␣
 ↪y=df['Engine'].value_counts().sort_index())
chart.set_ylabel('No.of cars', fontdict={'size': 15})
chart.set_xlabel('Engine', fontdict={'size': 15});
```



Which Cars are with highest average price? Represent with graph.

```
[ ]: av_price=df.groupby('title')[['Price']].mean().
       ↪sort_values('Price',ascending=False).head(10)
     plt.xticks(rotation=70)
     sns.barplot(y=av_price.index,x=av_price.Price,color='yellow');
```

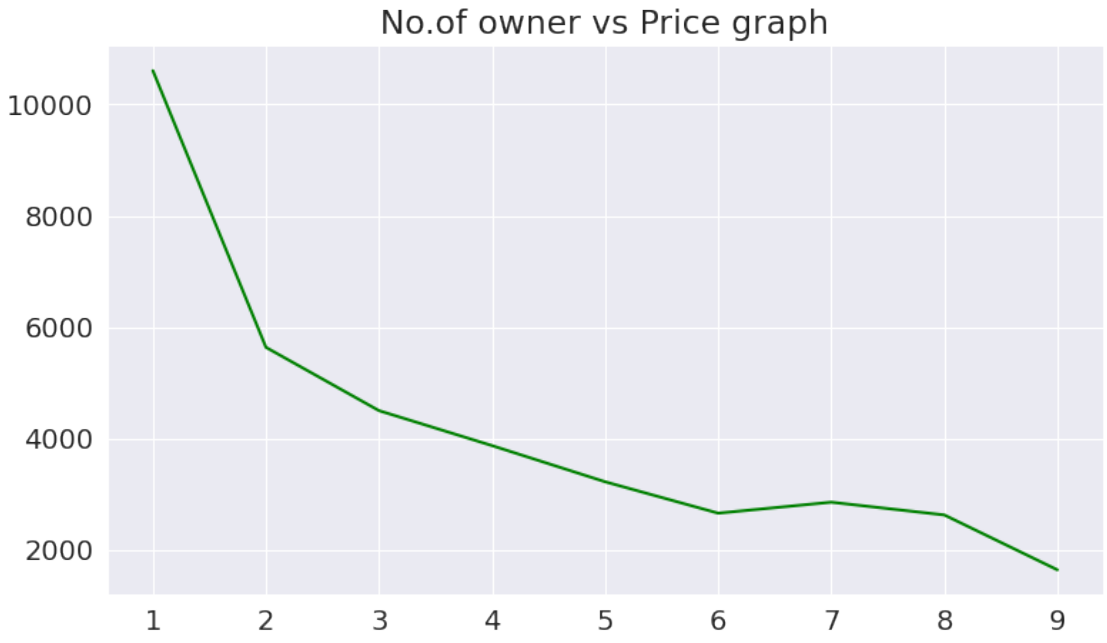


Show How is the Effect on price with the no. of owners?

[ ]:
```
owner_data=df.groupby('Previous Owners')[['Price']].mean()
```

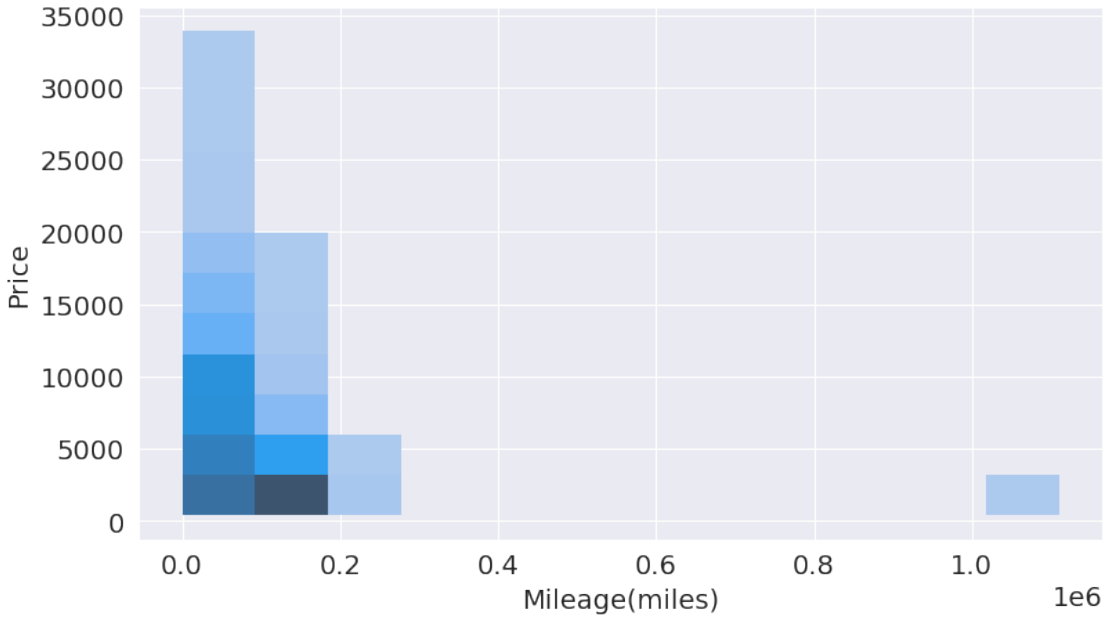
[ ]:
```
plt.title('No.of owner vs Price graph')
plt.plot(owner_data.index,owner_data,color='green')
plt.show
```

[ ]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



What is the Change in price with mileage with the help of graph?

[ ]:
```
sns.histplot(x=df['Mileage(miles)'],y=df.Price,bins =12);
```


35000
30000
25000
20000
15000
10000
5000
0
Price
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Mileage(miles)

sales-prediction

December 11, 2023

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

[2]:
```
import os
for dirname, _, filenames in os.walk('""C:/Users/ahap0/Downloads/Advertising.
↪csv"""'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[3]:
```
df=pd.read_csv("C:/Users/ahap0/Downloads/Advertising.csv")
df
```

[3]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
..	…	…	…	…	…
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

[200 rows x 5 columns]

[4]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
df.shape
```

[4]: (200, 4)

ques1. Calculate the count,mean ,1st quartile,median , 3rd quartile,std,min max?

[5]:
```
df.describe()
```

[5]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000

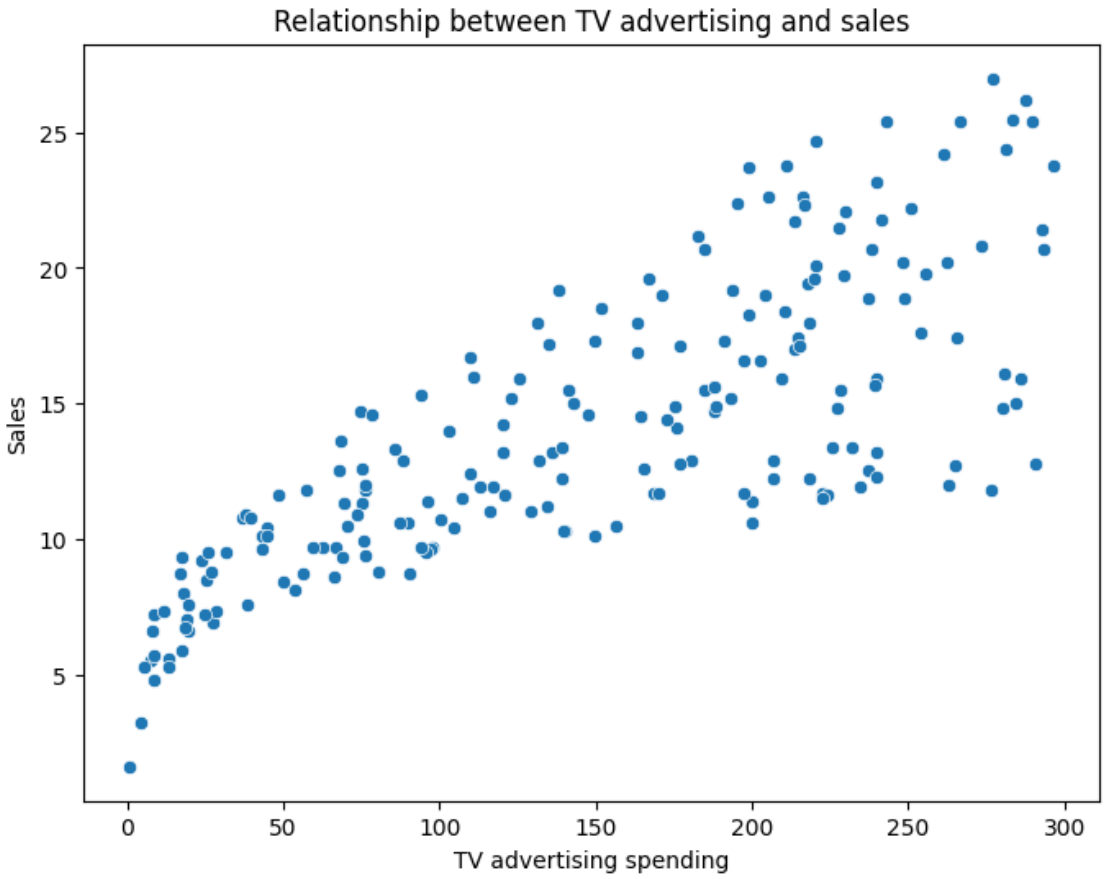
```
mean   147.042500   23.264000   30.554000   14.022500
std     85.854236   14.846809   21.778621    5.217457
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   10.375000
50%    149.750000   22.900000   25.750000   12.900000
75%    218.825000   36.525000   45.100000   17.400000
max    296.400000   49.600000  114.000000   27.000000
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

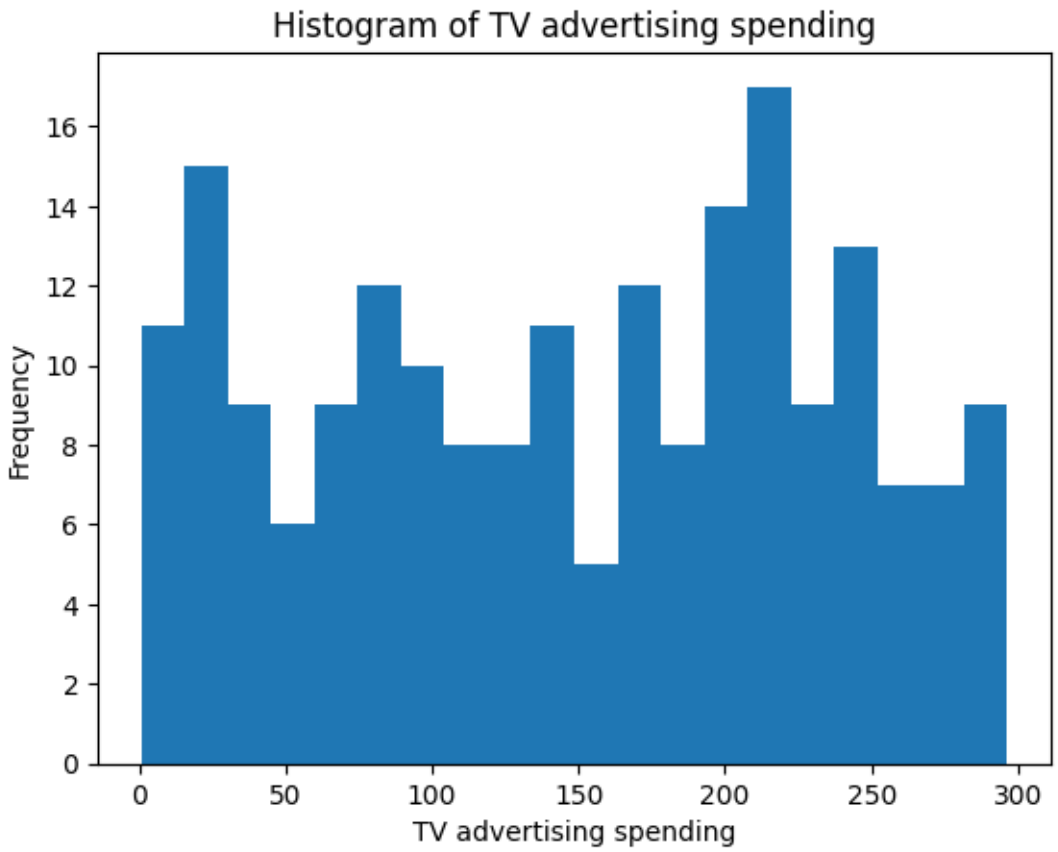
ques 2. plot the relationship between 'TV' advertising spending and 'Sales'?

[7]:
```
import seaborn as sns
import matplotlib.pyplot as plt
# Scatter plot between 'TV' advertising spending and 'Sales'
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='TV',y='Sales')
plt.xlabel('TV advertising spending')
plt.ylabel('Sales')
plt.title('Relationship between TV advertising and sales')
plt.show()
```



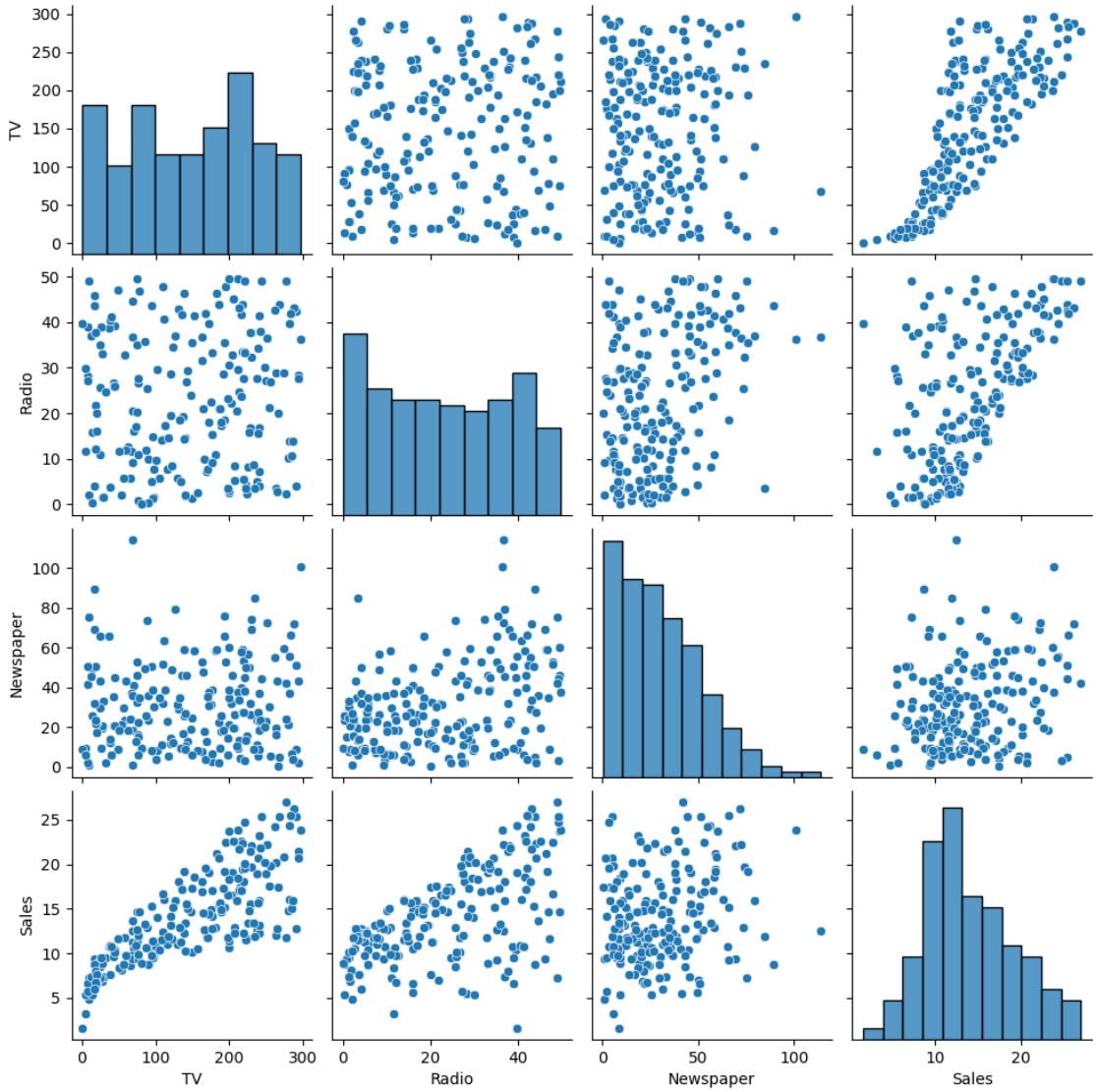
ques 3: Plot the Histogram of 'TV' advertising spending?

[8]:
```
plt.hist(df['TV'],bins=20)
plt.xlabel('TV advertising spending')
plt.ylabel('Frequency')
plt.title('Histogram of TV advertising spending')
plt.show()
```



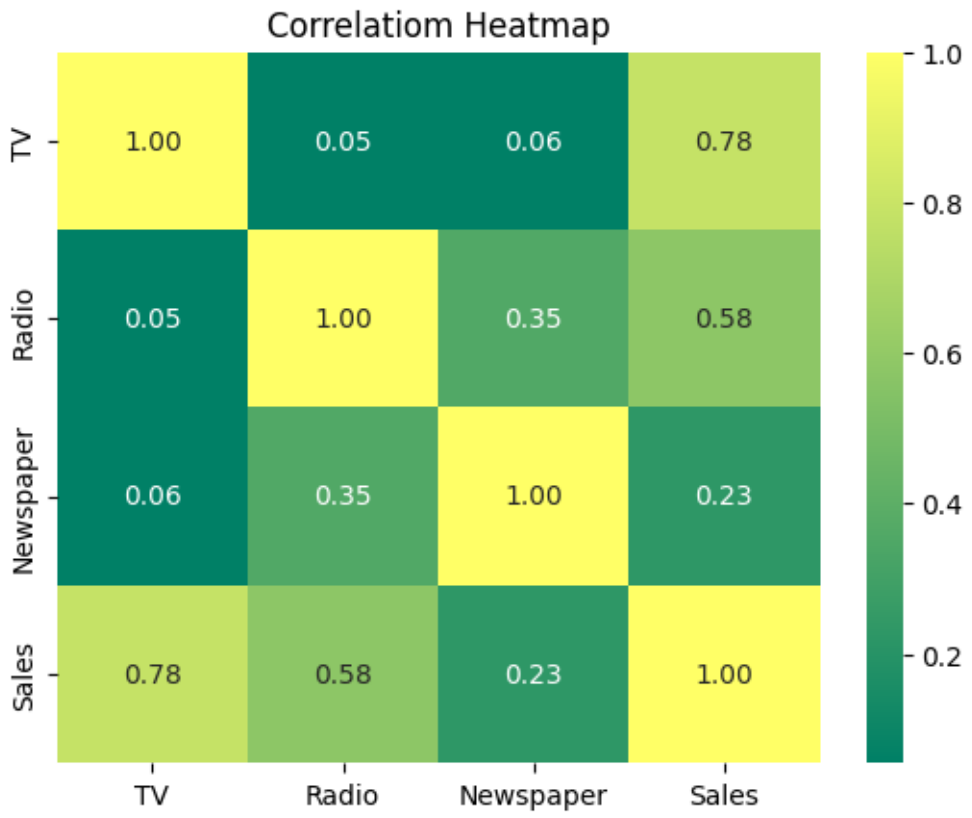
ques 4: PLot both relationship and histogram?

[9]:
```
sns.pairplot(df)
plt.show()
```



ques5: plot a correlation heatmap?

[10]:
```
correlation_matrix=df.corr()

sns.heatmap(correlation_matrix,annot=True,cmap='summer',fmt='.2f')
plt.title('Correlatiom Heatmap')
plt.show()
```


Correlatiom Heatmap
TV
Radio
Newspaper
Sales
1.00
0.05
0.06
0.78
0.05
1.00
0.35
0.58
0.06
0.35
1.00
0.23
0.78
0.58
0.23
1.00
TV
Radio
Newspaper
Sales
1.0
0.8
0.6
0.4
0.2

[ ]:

tree-servival-dataset

December 11, 2023

[1]:
```
import numpy as np
import pandas as pd
```

Q1. Handling the null values in the dataset (a) Dropping the entire row (b) Dropping the row in which all attributes of particular row is null (c) Handling by forward fill, backward fill and interpolate

[2]:
```
df=pd.read_csv('/kaggle/input/tree-dataset/Tree_Data.csv')
df
```

[2]:

	No	Plot	Subplot	Species	Light_ISF	Light_Cat	Core	\
0	126	1	C	Acer saccharum	0.106	Med	2017	
1	11	1	C	Quercus alba	0.106	Med	2017	
2	12	1	C	Quercus rubra	0.106	Med	2017	
3	2823	7	D	Acer saccharum	0.080	Med	2016	
4	5679	14	A	Acer saccharum	0.060	Low	2017	
…	…	…	…	…	…	…	…	
2778	7165	17	B	Prunus serotina	0.111	Med	2017	
2779	7217	17	D	Quercus alba	0.118	Med	2017	
2780	7306	17	D	Quercus alba	0.118	Med	2017	
2781	7771	18	D	Quercus alba	0.161	High	2017	
2782	7401	18	A	Prunus serotina	0.141	High	2016	

	Soil	Adult	Sterile	…	AMF	EMF	Phenolics	\
0	Prunus serotina	I	Non-Sterile	…	22.00	NaN	-0.56	
1	Quercus rubra	970	Non-Sterile	…	15.82	31.07	5.19	
2	Prunus serotina	J	Non-Sterile	…	24.45	28.19	3.36	
3	Prunus serotina	J	Non-Sterile	…	22.23	NaN	-0.71	
4	Prunus serotina	689	Non-Sterile	…	21.15	NaN	-0.58	
…	…	…	…	…	…	…	…	
2778	Populus grandidentata	891	Non-Sterile	…	40.89	NaN	0.83	
2779	Acer rubrum	1468	Non-Sterile	…	15.47	32.82	4.88	
2780	Quercus rubra	1454	Non-Sterile	…	11.96	37.67	5.51	
2781	Sterile	1297	Sterile	…	16.99	22.51	4.28	
2782	Populus grandidentata	118	Non-Sterile	…	60.46	NaN	1.00	

	Lignin	NSC	Census	Time	Event	Harvest	Alive
0	13.86	12.15	4	14.0	1.0	NaN	NaN

```
1     20.52  19.29      33  115.5    0.0      NaN      X
2     24.74  15.01      18   63.0    1.0      NaN    NaN
3     14.29  12.36       4   14.0    1.0      NaN    NaN
4     10.85  11.20       4   14.0    1.0      NaN    NaN
…       …      …       …     …      …        …      …
2778   9.15  11.88      16   56.0    1.0      NaN    NaN
2779  19.01  23.50      16   56.0    1.0      NaN    NaN
2780  21.13  19.10      16   56.0    1.0      NaN    NaN
2781  19.38  21.36      33  115.5    NaN      NaN    NaN
2782   9.04  11.82      16   56.0    1.0      NaN    NaN

[2783 rows x 24 columns]
```

[3]:
```
#(a) Dropping the entire row
a=df.dropna()
a
```

[3]:
```
Empty DataFrame
Columns: [No, Plot, Subplot, Species, Light_ISF, Light_Cat, Core, Soil, Adult,
Sterile, Conspecific, Myco, SoilMyco, PlantDate, AMF, EMF, Phenolics, Lignin,
NSC, Census, Time, Event, Harvest, Alive]
Index: []

[0 rows x 24 columns]
```

[4]:
```
#(b) Dropping the row in which all attributes of particular row is null
b=df.dropna(how='all')
b
```

[4]:
```
         No  Plot Subplot          Species  Light_ISF Light_Cat  Core  \
0       126     1       C   Acer saccharum      0.106       Med  2017
1        11     1       C     Quercus alba      0.106       Med  2017
2        12     1       C    Quercus rubra      0.106       Med  2017
3      2823     7       D   Acer saccharum      0.080       Med  2016
4      5679    14       A   Acer saccharum      0.060       Low  2017
…     …   …   …       …              …        …       …   …
2778   7165    17       B  Prunus serotina      0.111       Med  2017
2779   7217    17       D     Quercus alba      0.118       Med  2017
2780   7306    17       D     Quercus alba      0.118       Med  2017
2781   7771    18       D     Quercus alba      0.161      High  2017
2782   7401    18       A  Prunus serotina      0.141      High  2016

                     Soil Adult      Sterile  …    AMF    EMF Phenolics  \
0           Prunus serotina     I  Non-Sterile  …  22.00    NaN     -0.56
1             Quercus rubra   970  Non-Sterile  …  15.82  31.07      5.19
2           Prunus serotina     J  Non-Sterile  …  24.45  28.19      3.36
3           Prunus serotina     J  Non-Sterile  …  22.23    NaN     -0.71
```

```
4           Prunus serotina   689  Non-Sterile  …  21.15    NaN     -0.58
…                       …   …          …   …    …    …       …
2778  Populus grandidentata   891  Non-Sterile  …  40.89    NaN      0.83
2779            Acer rubrum  1468  Non-Sterile  …  15.47  32.82      4.88
2780          Quercus rubra  1454  Non-Sterile  …  11.96  37.67      5.51
2781                Sterile  1297      Sterile  …  16.99  22.51      4.28
2782  Populus grandidentata   118  Non-Sterile  …  60.46    NaN      1.00

     Lignin    NSC  Census   Time  Event  Harvest  Alive
0     13.86  12.15       4   14.0    1.0      NaN    NaN
1     20.52  19.29      33  115.5    0.0      NaN      X
2     24.74  15.01      18   63.0    1.0      NaN    NaN
3     14.29  12.36       4   14.0    1.0      NaN    NaN
4     10.85  11.20       4   14.0    1.0      NaN    NaN
…     …    …      …    …     …       …    …
2778   9.15  11.88      16   56.0    1.0      NaN    NaN
2779  19.01  23.50      16   56.0    1.0      NaN    NaN
2780  21.13  19.10      16   56.0    1.0      NaN    NaN
2781  19.38  21.36      33  115.5    NaN      NaN    NaN
2782   9.04  11.82      16   56.0    1.0      NaN    NaN

[2783 rows x 24 columns]
```

[6]:
```
#(c) Handling by forward fill, backward fill and interpolate
c1=df.fillna(method='ffill')
c2=df.fillna(method='bfill')
c3=df.interpolate()
```

[7]:
```
c1
```

[7]:
```
        No  Plot Subplot          Species  Light_ISF Light_Cat  Core  \
0      126     1       C   Acer saccharum      0.106       Med  2017
1       11     1       C     Quercus alba      0.106       Med  2017
2       12     1       C    Quercus rubra      0.106       Med  2017
3     2823     7       D   Acer saccharum      0.080       Med  2016
4     5679    14       A   Acer saccharum      0.060       Low  2017
…     …    …      …              …         …       …    …
2778  7165    17       B  Prunus serotina      0.111       Med  2017
2779  7217    17       D     Quercus alba      0.118       Med  2017
2780  7306    17       D     Quercus alba      0.118       Med  2017
2781  7771    18       D     Quercus alba      0.161      High  2017
2782  7401    18       A  Prunus serotina      0.141      High  2016

                      Soil Adult      Sterile  …    AMF    EMF Phenolics  \
0           Prunus serotina     I  Non-Sterile  …  22.00    NaN     -0.56
1             Quercus rubra   970  Non-Sterile  …  15.82  31.07      5.19
2           Prunus serotina     J  Non-Sterile  …  24.45  28.19      3.36
```

```
3           Prunus serotina     J  Non-Sterile  …  22.23  28.19     -0.71
4           Prunus serotina   689  Non-Sterile  …  21.15  28.19     -0.58
…                       …   …          …  …    …    …       …         …
2778  Populus grandidentata   891  Non-Sterile  …  40.89  39.00      0.83
2779            Acer rubrum  1468  Non-Sterile  …  15.47  32.82      4.88
2780          Quercus rubra  1454  Non-Sterile  …  11.96  37.67      5.51
2781                Sterile  1297      Sterile  …  16.99  22.51      4.28
2782  Populus grandidentata   118  Non-Sterile  …  60.46  22.51      1.00

     Lignin    NSC  Census   Time  Event  Harvest  Alive
0     13.86  12.15       4   14.0    1.0      NaN    NaN
1     20.52  19.29      33  115.5    0.0      NaN      X
2     24.74  15.01      18   63.0    1.0      NaN      X
3     14.29  12.36       4   14.0    1.0      NaN      X
4     10.85  11.20       4   14.0    1.0      NaN      X
…       …    …     …     …    …    …       …    …
2778   9.15  11.88      16   56.0    1.0        X      X
2779  19.01  23.50      16   56.0    1.0        X      X
2780  21.13  19.10      16   56.0    1.0        X      X
2781  19.38  21.36      33  115.5    1.0        X      X
2782   9.04  11.82      16   56.0    1.0        X      X

[2783 rows x 24 columns]
```

[8]:
```
c2
```

[8]:
```
        No  Plot Subplot          Species  Light_ISF Light_Cat  Core  \
0      126     1       C   Acer saccharum      0.106       Med  2017
1       11     1       C     Quercus alba      0.106       Med  2017
2       12     1       C    Quercus rubra      0.106       Med  2017
3     2823     7       D   Acer saccharum      0.080       Med  2016
4     5679    14       A   Acer saccharum      0.060       Low  2017
…    …   …     …             …        …       …   …
2778  7165    17       B  Prunus serotina      0.111       Med  2017
2779  7217    17       D     Quercus alba      0.118       Med  2017
2780  7306    17       D     Quercus alba      0.118       Med  2017
2781  7771    18       D     Quercus alba      0.161      High  2017
2782  7401    18       A  Prunus serotina      0.141      High  2016

                        Soil Adult      Sterile  …    AMF    EMF Phenolics  \
0           Prunus serotina     I  Non-Sterile  …  22.00  31.07     -0.56
1             Quercus rubra   970  Non-Sterile  …  15.82  31.07      5.19
2           Prunus serotina     J  Non-Sterile  …  24.45  28.19      3.36
3           Prunus serotina     J  Non-Sterile  …  22.23  20.00     -0.71
4           Prunus serotina   689  Non-Sterile  …  21.15  20.00     -0.58
…                       …   …          …  …    …    …       …         …
2778  Populus grandidentata   891  Non-Sterile  …  40.89  32.82      0.83
```

```
2779            Acer rubrum  1468  Non-Sterile  …  15.47  32.82      4.88
2780          Quercus rubra  1454  Non-Sterile  …  11.96  37.67      5.51
2781                Sterile  1297      Sterile  …  16.99  22.51      4.28
2782  Populus grandidentata   118  Non-Sterile  …  60.46    NaN      1.00

     Lignin    NSC  Census   Time  Event  Harvest  Alive
0     13.86  12.15       4   14.0    1.0        X      X
1     20.52  19.29      33  115.5    0.0        X      X
2     24.74  15.01      18   63.0    1.0        X      X
3     14.29  12.36       4   14.0    1.0        X      X
4     10.85  11.20       4   14.0    1.0        X      X
…       …      …       …      …      …        …      …
2778   9.15  11.88      16   56.0    1.0      NaN    NaN
2779  19.01  23.50      16   56.0    1.0      NaN    NaN
2780  21.13  19.10      16   56.0    1.0      NaN    NaN
2781  19.38  21.36      33  115.5    1.0      NaN    NaN
2782   9.04  11.82      16   56.0    1.0      NaN    NaN

[2783 rows x 24 columns]
```

[9]:
```
c3
```

[9]:
```
         No  Plot Subplot          Species  Light_ISF Light_Cat  Core  \
0       126     1       C   Acer saccharum      0.106       Med  2017
1        11     1       C     Quercus alba      0.106       Med  2017
2        12     1       C    Quercus rubra      0.106       Med  2017
3      2823     7       D   Acer saccharum      0.080       Med  2016
4      5679    14       A   Acer saccharum      0.060       Low  2017
…     …   …      …              …        …       …   …
2778  7165    17       B  Prunus serotina      0.111       Med  2017
2779  7217    17       D     Quercus alba      0.118       Med  2017
2780  7306    17       D     Quercus alba      0.118       Med  2017
2781  7771    18       D     Quercus alba      0.161      High  2017
2782  7401    18       A  Prunus serotina      0.141      High  2016

                      Soil Adult      Sterile  …    AMF      EMF Phenolics  \
0           Prunus serotina     I  Non-Sterile  …  22.00      NaN     -0.56
1             Quercus rubra   970  Non-Sterile  …  15.82  31.0700      5.19
2           Prunus serotina     J  Non-Sterile  …  24.45  28.1900      3.36
3           Prunus serotina     J  Non-Sterile  …  22.23  26.1425     -0.71
4           Prunus serotina   689  Non-Sterile  …  21.15  24.0950     -0.58
…                     …   …          …  …    …      …       …
2778  Populus grandidentata   891  Non-Sterile  …  40.89  35.9100      0.83
2779            Acer rubrum  1468  Non-Sterile  …  15.47  32.8200      4.88
2780          Quercus rubra  1454  Non-Sterile  …  11.96  37.6700      5.51
2781                Sterile  1297      Sterile  …  16.99  22.5100      4.28
2782  Populus grandidentata   118  Non-Sterile  …  60.46  22.5100      1.00
```

```
      Lignin    NSC  Census   Time  Event  Harvest  Alive
0      13.86  12.15       4   14.0    1.0      NaN    NaN
1      20.52  19.29      33  115.5    0.0      NaN      X
2      24.74  15.01      18   63.0    1.0      NaN    NaN
3      14.29  12.36       4   14.0    1.0      NaN    NaN
4      10.85  11.20       4   14.0    1.0      NaN    NaN
…      …    …      …    …     …       …    …
2778    9.15  11.88      16   56.0    1.0      NaN    NaN
2779   19.01  23.50      16   56.0    1.0      NaN    NaN
2780   21.13  19.10      16   56.0    1.0      NaN    NaN
2781   19.38  21.36      33  115.5    1.0      NaN    NaN
2782    9.04  11.82      16   56.0    1.0      NaN    NaN

[2783 rows x 24 columns]
```

Q2. Assigning some values as null values

[11]:
```
df['Adult'].unique()
```

[11]:
```
array(['I', '970', 'J', '689', '1332', '891', '1595', '1323', '394',
       '561', '1478', '1320', '1454', '921', '984', '118', '1757', '1384',
       '1688', '961', '1715', '50', '1468', '1201', '1386', '277', '415',
       '285', '275', '1205', '1330', '1297', '1326', 'H', '1027', 'G'],
      dtype=object)
```

[15]:
```
missing_value=["I", "J", "H", "G",np.nan]
df=pd.read_csv('/kaggle/input/tree-dataset/Tree_Data.
  ↪csv',na_values=missing_value)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2783 entries, 0 to 2782
Data columns (total 24 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   No           2783 non-null   int64
 1   Plot         2783 non-null   int64
 2   Subplot      2783 non-null   object
 3   Species      2783 non-null   object
 4   Light_ISF    2783 non-null   float64
 5   Light_Cat    2783 non-null   object
 6   Core         2783 non-null   int64
 7   Soil         2783 non-null   object
 8   Adult        2433 non-null   float64
 9   Sterile      2783 non-null   object
 10  Conspecific  2783 non-null   object
 11  Myco         2783 non-null   object
```

```
 12  SoilMyco     2783 non-null   object
 13  PlantDate    2783 non-null   object
 14  AMF          2783 non-null   float64
 15  EMF          1283 non-null   float64
 16  Phenolics    2783 non-null   float64
 17  Lignin       2783 non-null   float64
 18  NSC          2783 non-null   float64
 19  Census       2783 non-null   int64
 20  Time         2783 non-null   float64
 21  Event        2782 non-null   float64
 22  Harvest      704 non-null    object
 23  Alive        491 non-null    object
dtypes: float64(9), int64(4), object(11)
memory usage: 521.9+ KB
```

Q3. Find out the types of species and soil

[16]:
```
df['Species'].unique()
```

[16]:
```
array(['Acer saccharum', 'Quercus alba', 'Quercus rubra',
       'Prunus serotina'], dtype=object)
```

[17]:
```
df['Soil'].unique()
```

[17]:
```
array(['Prunus serotina', 'Quercus rubra', 'Acer rubrum',
       'Populus grandidentata', 'Sterile', 'Acer saccharum',
       'Quercus alba'], dtype=object)
```

Q4. Change the data type of "Event" attribute and find out the mean of "Time" column

[25]:
```
df['Event']=df['Event'].fillna(0)
```

[26]:
```
df['Event']=df['Event'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2783 entries, 0 to 2782
Data columns (total 24 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   No           2783 non-null   int64
 1   Plot         2783 non-null   int64
 2   Subplot      2783 non-null   object
 3   Species      2783 non-null   object
 4   Light_ISF    2783 non-null   float64
 5   Light_Cat    2783 non-null   object
 6   Core         2783 non-null   int64
 7   Soil         2783 non-null   object
```

```
 8   Adult         2433 non-null   float64
 9   Sterile       2783 non-null   object
 10  Conspecific   2783 non-null   object
 11  Myco          2783 non-null   object
 12  SoilMyco      2783 non-null   object
 13  PlantDate     2783 non-null   object
 14  AMF           2783 non-null   float64
 15  EMF           1283 non-null   float64
 16  Phenolics     2783 non-null   float64
 17  Lignin        2783 non-null   float64
 18  NSC           2783 non-null   float64
 19  Census        2783 non-null   int64
 20  Time          2783 non-null   float64
 21  Event         2783 non-null   int64
 22  Harvest       704 non-null    object
 23  Alive         491 non-null    object
dtypes: float64(8), int64(5), object(11)
memory usage: 521.9+ KB
```

[27]:
```
df['Time'].mean()
```

[27]:
```
53.487243981315125
```

Q5. Find the number of Sterile and Non-Sterile trees.

[28]:
```
desired_value = 'Sterile'
rows_with_desired_value = len(df[df['Sterile'] == desired_value])

# Display the number of rows with the specified attribute value
print(f"Number of rows with '{desired_value}': {rows_with_desired_value}")
```

```
Number of rows with 'Sterile': 423
```

[29]:
```
desired_value = 'Non-Sterile'
rows_with_desired_value = len(df[df['Sterile'] == desired_value])

# Display the number of rows with the specified attribute value
print(f"Number of rows with '{desired_value}': {rows_with_desired_value}")
```

```
Number of rows with 'Non-Sterile': 2360
```

```
import pandas as pd
import numpy as np
import datetime
from time import strftime
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

# Reading the dataset

base_data = pd.read_csv('Data.csv')

base_data
```

```
           PatientId  AppointmentID Gender          ScheduledDay  \
0       2.987250e+13        5642903      F  2016-04-29T18:38:08Z
1       5.589978e+14        5642503      M  2016-04-29T16:08:27Z
2       4.262962e+12        5642549      F  2016-04-29T16:19:04Z
3       8.679512e+11        5642828      F  2016-04-29T17:29:31Z
4       8.841186e+12        5642494      F  2016-04-29T16:07:23Z
...              ...            ...    ...                   ...
110522  2.572134e+12        5651768      F  2016-05-03T09:15:35Z
110523  3.596266e+12        5650093      F  2016-05-03T07:27:33Z
110524  1.557663e+13        5630692      F  2016-04-27T16:03:52Z
110525  9.213493e+13        5630323      F  2016-04-27T15:09:23Z
110526  3.775115e+14        5629448      F  2016-04-27T13:30:56Z

              AppointmentDay  Age      Neighbourhood  Scholarship  \
0       2016-04-29T00:00:00Z   62    JARDIM DA PENHA            0
1       2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0
2       2016-04-29T00:00:00Z   62      MATA DA PRAIA            0
3       2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI            0
4       2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0
...                      ...  ...                ...          ...
110522  2016-06-07T00:00:00Z   56        MARIA ORTIZ            0
110523  2016-06-07T00:00:00Z   51        MARIA ORTIZ            0
110524  2016-06-07T00:00:00Z   21        MARIA ORTIZ            0
110525  2016-06-07T00:00:00Z   38        MARIA ORTIZ            0
110526  2016-06-07T00:00:00Z   54        MARIA ORTIZ            0

        Hipertension  Diabetes  Alcoholism  Handcap  SMS_received No-
show
0                  1         0           0        0             0
No
1                  0         0           0        0             0
No
2                  0         0           0        0             0
No
3                  0         0           0        0             0
```

```
No
4                  1         1           0        0             0
No
...              ...       ...         ...      ...           ...
...
110522             0         0           0        0             1
No
110523             0         0           0        0             1
No
110524             0         0           0        0             1
No
110525             0         0           0        0             1
No
110526             0         0           0        0             1
No

[110527 rows x 14 columns]
```

```
base_data.shape
```

```
(110527, 14)
```

```
base_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

```
#modifying the date and time into standard form
base_data['ScheduledDay'] =
pd.to_datetime(base_data['ScheduledDay']).dt.date.astype('datetime64[n
s]')
```

```
base_data['AppointmentDay'] =
pd.to_datetime(base_data['AppointmentDay']).dt.date.astype('datetime64
[ns]')

base_data.head(5)
```

```
      PatientId  AppointmentID Gender ScheduledDay AppointmentDay  Age
\
0  2.987250e+13        5642903      F   2016-04-29     2016-04-29   62

1  5.589978e+14        5642503      M   2016-04-29     2016-04-29   56

2  4.262962e+12        5642549      F   2016-04-29     2016-04-29   62

3  8.679512e+11        5642828      F   2016-04-29     2016-04-29    8

4  8.841186e+12        5642494      F   2016-04-29     2016-04-29   56


       Neighbourhood  Scholarship  Hipertension  Diabetes  Alcoholism
\
0    JARDIM DA PENHA            0             1         0           0

1    JARDIM DA PENHA            0             0         0           0

2      MATA DA PRAIA            0             0         0           0

3  PONTAL DE CAMBURI            0             0         0           0

4    JARDIM DA PENHA            0             1         1           0


   Handcap  SMS_received No-show
0        0             0      No
1        0             0      No
2        0             0      No
3        0             0      No
4        0             0      No
```

for the schedule day and appointment day storing the weekdays only into a variable

```
# 5 is Saturday, 6 is Sunday

base_data['sch_weekday'] = base_data['ScheduledDay'].dt.dayofweek

base_data['app_weekday'] = base_data['AppointmentDay'].dt.dayofweek

base_data['sch_weekday'].value_counts()
```

```
1    26168
2    24262
```

```
0    23085
4    18915
3    18073
5       24
Name: sch_weekday, dtype: int64
```

```
base_data['app_weekday'].value_counts()
```

```
2    25867
1    25640
0    22715
4    19019
3    17247
5       39
Name: app_weekday, dtype: int64
```

```
base_data.columns
```

```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship',
'Hipertension',
       'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-show',
       'sch_weekday', 'app_weekday'],
      dtype='object')
```

```
#changing the name of some cloumns
base_data= base_data.rename(columns={'Hipertension': 'Hypertension',
'Handcap': 'Handicap', 'SMS_received': 'SMSReceived', 'No-show':
'NoShow'})
```

```
base_data.columns
```

```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship',
'Hypertension',
       'Diabetes', 'Alcoholism', 'Handicap', 'SMSReceived', 'NoShow',
       'sch_weekday', 'app_weekday'],
      dtype='object')
```

```
base_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 16 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64
 2   Gender          110527 non-null  object
 3   ScheduledDay    110527 non-null  datetime64[ns]
 4   AppointmentDay  110527 non-null  datetime64[ns]
```

```
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hypertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handicap        110527 non-null  int64
 12  SMSReceived     110527 non-null  int64
 13  NoShow          110527 non-null  object
 14  sch_weekday     110527 non-null  int64
 15  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(10), object(3)
memory usage: 13.5+ MB
```

```
# dropping some columns which have no significance
base_data.drop(['PatientId', 'AppointmentID', 'Neighbourhood'],
axis=1, inplace=True)

base_data
```

```
       Gender ScheduledDay AppointmentDay  Age  Scholarship
Hypertension  \
0           F   2016-04-29     2016-04-29   62            0
1
1           M   2016-04-29     2016-04-29   56            0
0
2           F   2016-04-29     2016-04-29   62            0
0
3           F   2016-04-29     2016-04-29    8            0
0
4           F   2016-04-29     2016-04-29   56            0
1
...       ...          ...            ...  ...          ...
...
110522      F   2016-05-03     2016-06-07   56            0
0
110523      F   2016-05-03     2016-06-07   51            0
0
110524      F   2016-04-27     2016-06-07   21            0
0
110525      F   2016-04-27     2016-06-07   38            0
0
110526      F   2016-04-27     2016-06-07   54            0
0

        Diabetes  Alcoholism  Handicap  SMSReceived NoShow
sch_weekday  \
0              0           0         0            0     No
4
1              0           0         0            0     No
```

```
4
2               0            0          0             0      No
4
3               0            0          0             0      No
4
4               1            0          0             0      No
4
...           ...          ...        ...           ...    ...           ..
.
110522          0            0          0             1      No
1
110523          0            0          0             1      No
1
110524          0            0          0             1      No
2
110525          0            0          0             1      No
2
110526          0            0          0             1      No
2

        app_weekday
0                 4
1                 4
2                 4
3                 4
4                 4
...             ...
110522            1
110523            1
110524            1
110525            1
110526            1

[110527 rows x 13 columns]

base_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns]
 2   AppointmentDay  110527 non-null  datetime64[ns]
 3   Age             110527 non-null  int64
 4   Scholarship     110527 non-null  int64
 5   Hypertension    110527 non-null  int64
 6   Diabetes        110527 non-null  int64
 7   Alcoholism      110527 non-null  int64
```

```
 8   Handicap        110527 non-null  int64
 9   SMSReceived     110527 non-null  int64
 10  NoShow          110527 non-null  object
 11  sch_weekday     110527 non-null  int64
 12  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), int64(9), object(2)
memory usage: 11.0+ MB
```

```
base_data.describe()
```

	Age	Scholarship	Hypertension	Diabetes
count	110527.000000	110527.000000	110527.000000	110527.000000
mean	37.088874	0.098266	0.197246	0.071865
std	23.110205	0.297675	0.397921	0.258265
min	-1.000000	0.000000	0.000000	0.000000
25%	18.000000	0.000000	0.000000	0.000000
50%	37.000000	0.000000	0.000000	0.000000
75%	55.000000	0.000000	0.000000	0.000000
max	115.000000	1.000000	1.000000	1.000000

	Alcoholism	Handicap	SMSReceived	sch_weekday
count	110527.000000	110527.000000	110527.000000	110527.000000
mean	0.030400	0.022248	0.321026	1.851955
std	0.171686	0.161543	0.466873	1.378520
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	2.000000
75%	0.000000	0.000000	1.000000	3.000000
max	1.000000	4.000000	1.000000	5.000000

	app_weekday
count	110527.000000
mean	1.858243
std	1.371672
min	0.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	5.000000

```
# calculating the % of appointments or not
100*base_data['NoShow'].value_counts()/len(base_data['NoShow'])
```

```
No     79.806744
Yes    20.193256
Name: NoShow, dtype: float64
```

```
base_data['NoShow'].value_counts()
```

```
No     88208
Yes    22319
Name: NoShow, dtype: int64
```

Missing Data - Initial Intuition

- Here, we don't have any missing data.

General Thumb Rules:

- For features with less missing values- can use regression to predict the missing values or fill with the mean of the values present, depending on the feature.
- For features with very high number of missing values- it is better to drop those columns as they give very less insight on analysis.
- As there's no thumb rule on what criteria do we delete the columns with high number of missing values, but generally you can delete the columns, if you have more than 30-40% of missing values.

Data Cleaning

1. Create a copy of base data for manupulation & processing

```
new_data = base_data.copy()

new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns]
 2   AppointmentDay  110527 non-null  datetime64[ns]
 3   Age             110527 non-null  int64
 4   Scholarship     110527 non-null  int64
 5   Hypertension    110527 non-null  int64
 6   Diabetes        110527 non-null  int64
 7   Alcoholism      110527 non-null  int64
 8   Handicap        110527 non-null  int64
 9   SMSReceived     110527 non-null  int64
 10  NoShow          110527 non-null  object
 11  sch_weekday     110527 non-null  int64
 12  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), int64(9), object(2)
memory usage: 11.0+ MB
```

As we don't have any null records, there's no data cleaning required

```
# Get the max tenure
print(base_data['Age'].max()) #72
```

```
115
```

```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 20) for i in range(1, 118, 20)]

base_data['Age_group'] = pd.cut(base_data.Age, range(1, 130, 20),
right=False, labels=labels)

base_data.drop(['Age'], axis=1, inplace=True)
```

Data Exploration

```
list(base_data.columns)
```

```
['Gender',
 'ScheduledDay',
 'AppointmentDay',
 'Scholarship',
 'Hypertension',
 'Diabetes',
 'Alcoholism',
 'Handicap',
 'SMSReceived',
 'NoShow',
 'sch_weekday',
 'app_weekday',
 'Age_group']
```

```
base_data['NoShow'] = np.where(base_data.NoShow == 'Yes',1,0)

base_data.NoShow.value_counts()
```

```
0    88208
1    22319
Name: NoShow, dtype: int64
```

Convert all the categorical variables into dummy variables

```
base_data_dummies = pd.get_dummies(base_data)
base_data_dummies.head()
```

```
  ScheduledDay AppointmentDay  Scholarship  Hypertension  Diabetes  \
0   2016-04-29     2016-04-29            0             1         0
1   2016-04-29     2016-04-29            0             0         0
2   2016-04-29     2016-04-29            0             0         0
3   2016-04-29     2016-04-29            0             0         0
4   2016-04-29     2016-04-29            0             1         1

   Alcoholism  Handicap  SMSReceived  NoShow  sch_weekday  app_weekday
\
```

```
0            0         0              0       0              4              4

1            0         0              0       0              4              4

2            0         0              0       0              4              4

3            0         0              0       0              4              4

4            0         0              0       0              4              4


   Gender_F  Gender_M  Age_group_1 - 21  Age_group_21 - 41
Age_group_41 - 61  \
0         1         0                 0                  0
0
1         0         1                 0                  0
1
2         1         0                 0                  0
0
3         1         0                 1                  0
0
4         1         0                 0                  0
1

   Age_group_61 - 81  Age_group_81 - 101  Age_group_101 - 121
0                  1                   0                    0
1                  0                   0                    0
2                  1                   0                    0
3                  0                   0                    0
4                  0                   0                    0
```

Build a corelation of all predictors with 'NoShow'

Findings

1. Female patients have taken more appointments then male patients
2. Ratio of Nohow and Show is almost equal for age group except Age 0 and Age 1 with 80% show rate for each age group
3. Each Neighbourhood have almost 80% show rate
4. There are 99666 patients without Scholarship and out of them around 80% have come for the visit and out of the 21801 patients with Scholarship around 75% of them have come for the visit.
5. there are around 88,726 patients without Hypertension and out of them around 78% have come for the visit and Out of the 21801 patients with Hypertension around 85% of them have come for the visit.
6. there are around 102,584 patients without Diabetes and out of them around 80% have come for the visit and Out of the 7,943 patients with Diabetes around 83% of them have come for the visit.

7. there are around 75,045 patients who have not received SMS and out of them around 84% have come for the visit and out of the 35,482 patients who have received SMS around 72% of them have come for the visit.
8. there is no appointments on sunday and on saturday appointments are very less in comparision to other week days

titanic

December 11, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
d = pd.read_csv(r"E:\DATA SET\titanic_train.csv")
d
```

[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0					
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1					
4	5	0	3	Allen, Mr. William Henry	male	35.0	0					
..	…	…	…	…	…	…	…	…	…	…	…	…
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0					
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0					
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1					
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0					
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0					

```
2          0  STON/O2. 3101282   7.9250   NaN        S
3          0            113803  53.1000  C123        S
4          0            373450   8.0500   NaN        S
..       …                …       …   …       …
886        0            211536  13.0000   NaN        S
887        0            112053  30.0000   B42        S
888        2        W./C. 6607  23.4500   NaN        S
889        0            111369  30.0000  C148        C
890        0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

[2]:
```
d.isnull().sum()
```

[2]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[3]:
```
d['Age'] = d['Age'].fillna(d['Age'].mean())
```

[4]:
```
d.isnull().sum()
```

[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[5]:
```
d.drop(['PassengerId','Cabin'],axis=1, inplace = True)
```

[6]:
```
d.dropna(inplace = True)
```

[7]:
```
d.isnull().sum()
```

[7]:
```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    0
dtype: int64
```

[8]:
```
d
```

[8]:
```
     Survived  Pclass                                               Name  \
0           0       3                            Braund, Mr. Owen Harris
1           1       1  Cumings, Mrs. John Bradley (Florence Briggs Th…
2           1       3                             Heikkinen, Miss. Laina
3           1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4           0       3                           Allen, Mr. William Henry
..        …     …                                                    …
886         0       2                              Montvila, Rev. Juozas
887         1       1                       Graham, Miss. Margaret Edith
888         0       3           Johnston, Miss. Catherine Helen "Carrie"
889         1       1                              Behr, Mr. Karl Howell
890         0       3                                Dooley, Mr. Patrick

        Sex        Age  SibSp  Parch            Ticket     Fare Embarked
0      male  22.000000      1      0         A/5 21171   7.2500        S
1    female  38.000000      1      0          PC 17599  71.2833        C
2    female  26.000000      0      0  STON/O2. 3101282   7.9250        S
3    female  35.000000      1      0            113803  53.1000        S
4      male  35.000000      0      0            373450   8.0500        S
..      …        …    …    …                …      …      …
886    male  27.000000      0      0            211536  13.0000        S
887  female  19.000000      0      0            112053  30.0000        S
888  female  29.699118      1      2        W./C. 6607  23.4500        S
889    male  26.000000      0      0            111369  30.0000        C
890    male  32.000000      0      0            370376   7.7500        Q

[889 rows x 10 columns]
```

[9]:
```
# d['Age'] = d['Age'].astype(int)
```

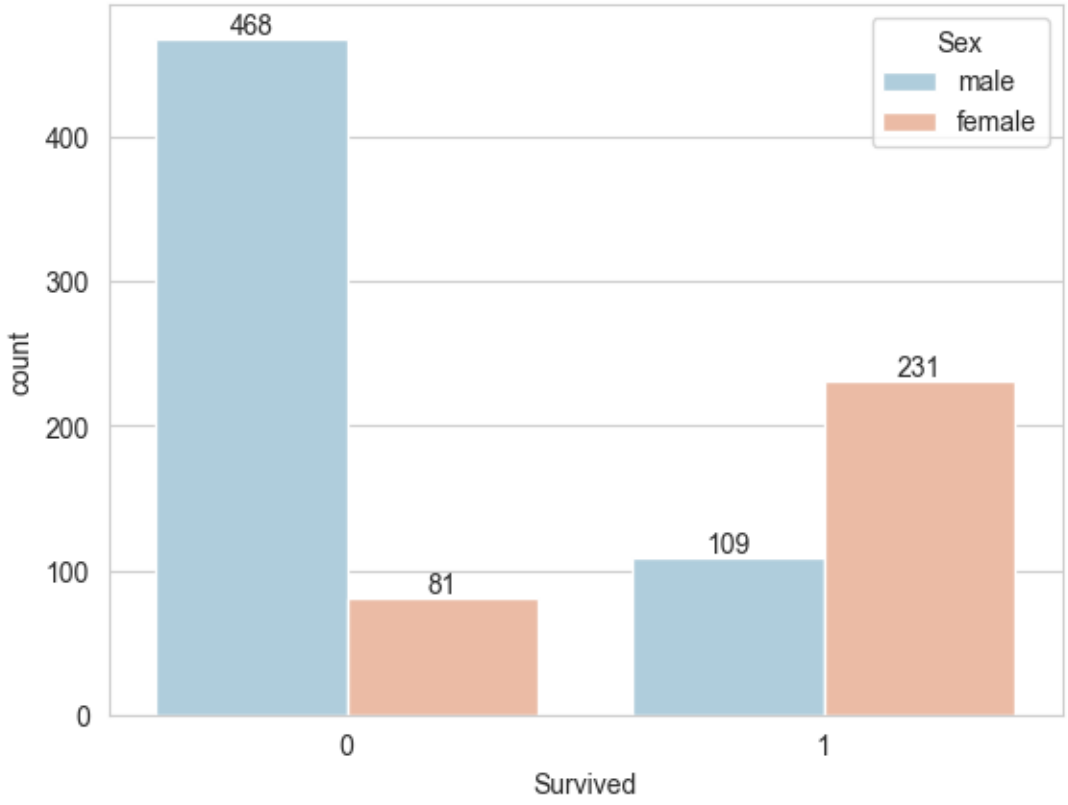
1 1. Find the maximum age and corresponding name

[10]:
```
# Find the maximum age and corresponding name
max_age_row = d.loc[d['Age'].idxmax()]

max_age = max_age_row['Age']
name_of_max_age = max_age_row['Name']

print(f"The maximum age in the Titanic dataset is {max_age} and the␣
 ↪corresponding name is {name_of_max_age}.")
```

```
The maximum age in the Titanic dataset is 80.0 and the corresponding name is
Barkworth, Mr. Algernon Henry Wilson.
```

2 2. The distribution of survivors and non-survivors across gender

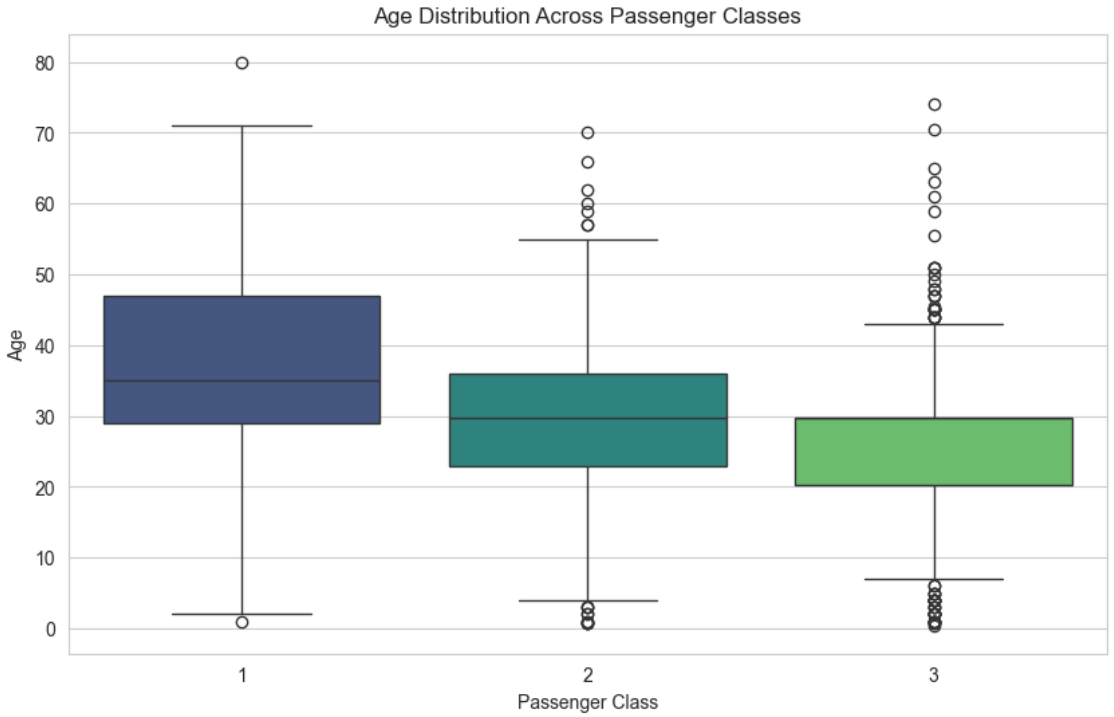
[11]:
```
sns.set_style('whitegrid')
dx = sns.countplot(x='Survived',hue='Sex',data=d,palette='RdBu_r')
for bars in dx.containers:
    dx.bar_label(bars)
```



3 3. The age distribution vary across different passenger classes

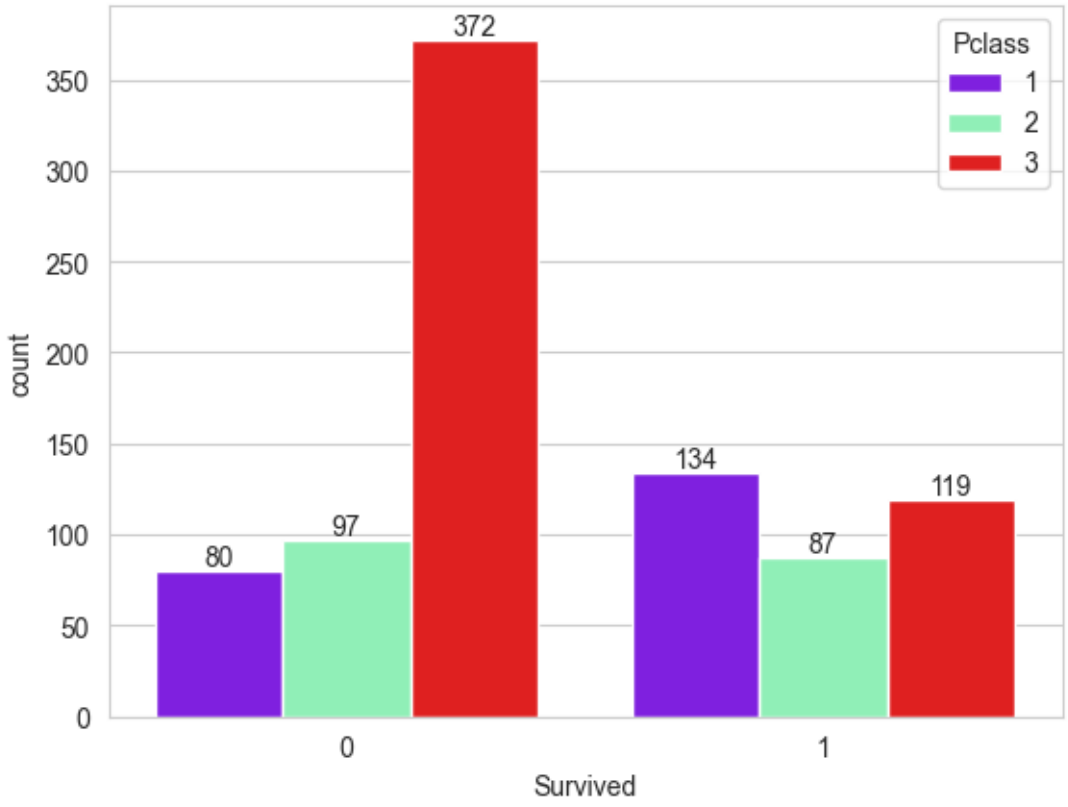
[12]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Pclass', y='Age', data=d, palette='viridis')
plt.title('Age Distribution Across Passenger Classes')
plt.xlabel('Passenger Class')
plt.ylabel('Age')
plt.show()
```

```
C:\Users\computer\AppData\Local\Temp\ipykernel_12188\41317869.py:2:
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.boxplot(x='Pclass', y='Age', data=d, palette='viridis')
```



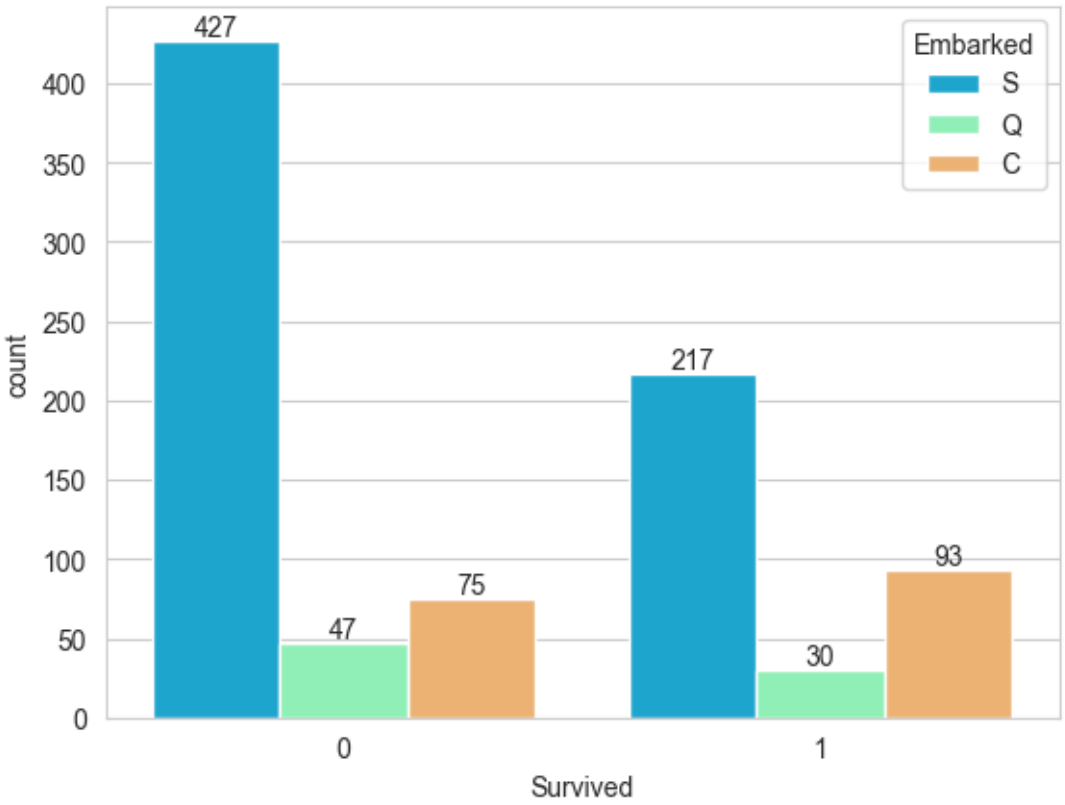
4 4. You visualize the distribution of survivors and non-survivors across different classes

[13]:
```
sns.set_style('whitegrid')
dx=sns.countplot(x='Survived',hue='Pclass',data=d,palette='rainbow')
for bars in dx.containers:
    dx.bar_label(bars)
```



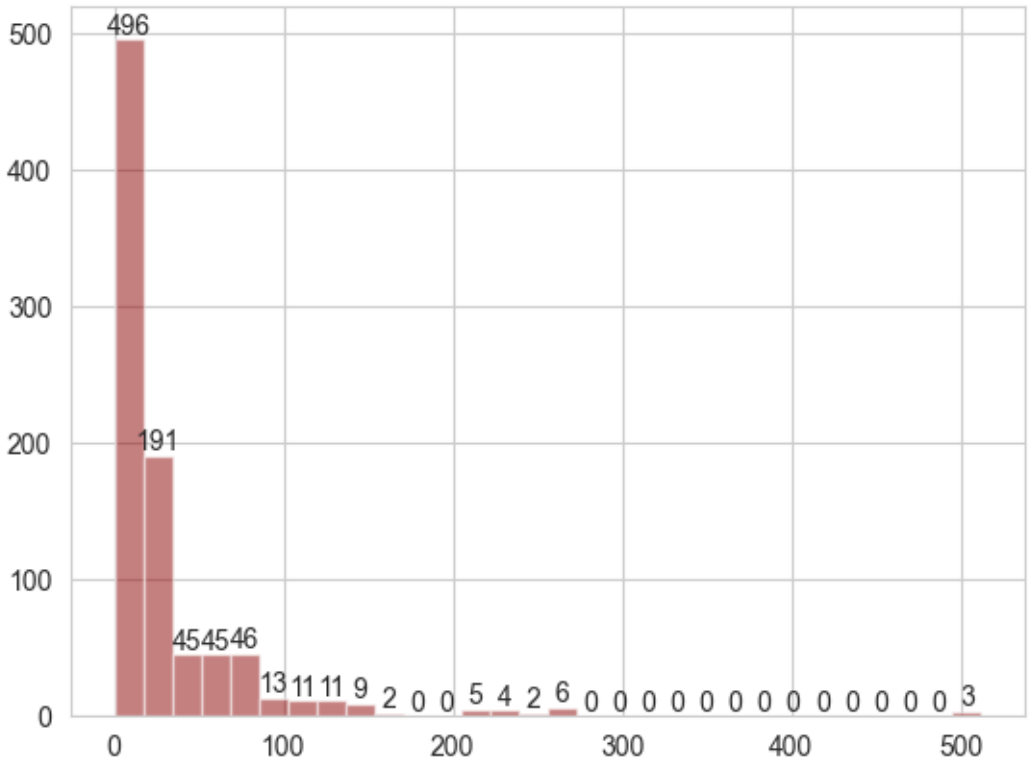
5 5. Correlation between the port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton) and survival

[14]:
```
sns.set_style('whitegrid')
dx=sns.countplot(x='Survived',hue='Embarked',data=d,palette='rainbow')
for bars in dx.containers:
    dx.bar_label(bars)
```



6 6. The fare distributed among passengers

[15]:
```
x = d['Fare'].hist(bins=30,color='darkred',alpha=0.5)
for bars in x.containers:
    x.bar_label(bars)
```


500
400
300
200
100
0
496
191
45 45 46
13 11 11 9 2 0 0 5 4 2 6 0 0 0 0 0 0 0 0 0 0 0 0 0 3
0
100
200
300
400
500

netflix

December 11, 2023

1. Data Preprocessing

[107]:
```
import pandas as pd
import numpy as np
```

[108]:
```
nf=pd.read_csv('/kaggle/input/netflix/NetFlix.csv')
nf
```

[108]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R…	Brazil	14-Aug-20	2020	TV-MA	4
1	s10	Movie	1920	Vikram Bhatt	Rajneesh Duggal, Adah Sharma, Indraneil Sengup…	India	15-Dec-17	2008	TV-MA	143
2	s100	Movie	3 Heroines	Iman Brotoseno	Reza Rahadian, Bunga Citra Lestari, Tara Basro…	Indonesia	5-Jan-19	2016	TV-PG	124
3	s1000	Movie	Blue Mountain State: The Rise of Thadland	Lev L. Spiro	Alan Ritchson, Darin Brooks, James Cade, Rob R…	United States	1-Jan-16	2016	TV-MA	90
4	s1001	TV Show	Blue Planet II	NaN	David Attenborough	United Kingdom	30-Dec-17	2017	TV-PG	1
…	…	…	…	…	…	…	…	…	…	…
7782	s995	TV Show	Blown Away	NaN	NaN	United States	1-Jan-19	2019	TV-14	4
7783	s996	TV Show	Blue Exorcist	NaN	Nobuhiko Okamoto, Jun Fukuyama, Kana Hanazawa,…	Japan	1-Jan-19	2019	TV-14	4
7784	s997	Movie	Blue Is the Warmest Color	Abdellatif Kechiche	Léa Seydoux, Adèle Exarchopoulos, Salim Kechio…	France	1-Jan-19	2019	TV-MA	4
7785	s998	Movie	Blue Jasmine	Woody Allen	Cate Blanchett, Sally Hawkins, Alec Baldwin, L…	United States	1-Jan-19	2019	TV-14	4
7786	s999	Movie	Blue Jay	Alex Lehmann	Sarah Paulson, Mark Duplass, Clu Gulager	United States	1-Jan-19	2019	TV-14	4

```
3              United States   1-Mar-16          2016      R        90
4             United Kingdom   3-Dec-18          2017   TV-G         1
…                        …        …           …     …       …
7782                 Canada  12-Jul-19          2019  TV-14         1
7783                  Japan   1-Sep-20          2017  TV-MA         2
7784  France, Belgium, Spain  26-Aug-16          2013  NC-17       180
7785           United States   8-Mar-19          2013  PG-13        98
7786           United States   6-Dec-16          2016  TV-MA        81

                                                  genres  \
0     International TV Shows, TV Dramas, TV Sci-Fi &…
1        Horror Movies, International Movies, Thrillers
2           Dramas, International Movies, Sports Movies
3                                               Comedies
4     British TV Shows, Docuseries, Science & Nature TV
…                                                    …
7782                International TV Shows, Reality TV
7783               Anime Series, International TV Shows
7784   Dramas, Independent Movies, International Movies
7785               Comedies, Dramas, Independent Movies
7786        Dramas, Independent Movies, Romantic Movies

                                              description
0     In a future where the elite inhabit an island …
1     An architect and his wife move into a castle t…
2     Three Indonesian women break records by becomi…
3     New NFL star Thad buys his old teammates' belo…
4     This sequel to the award-winning nature series…
…                                                   …
7782  Ten master artists turn up the heat in glassbl…
7783  Determined to throw off the curse of being Sat…
7784  Determined to fall in love, 15-year-old Adele …
7785  The high life leads to high anxiety for a fash…
7786  Two former high school sweethearts unexpectedl…

[7787 rows x 12 columns]
```

[109]:
```
nf.head()
```

[109]:
```
  show_id     type                                    title        director  \
0      s1  TV Show                                       3%             NaN
1     s10    Movie                                     1920   Vikram Bhatt
2    s100    Movie                                3 Heroines  Iman Brotoseno
3   s1000    Movie  Blue Mountain State: The Rise of Thadland   Lev L. Spiro
4   s1001  TV Show                             Blue Planet II             NaN

                                                 cast         country  \
```

```
0  João Miguel, Bianca Comparato, Michel Gomes, R…          Brazil
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…           India
2  Reza Rahadian, Bunga Citra Lestari, Tara Basro…       Indonesia
3  Alan Ritchson, Darin Brooks, James Cade, Rob R…   United States
4                               David Attenborough  United Kingdom

  date_added  release_year rating  duration  \
0  14-Aug-20          2020  TV-MA         4
1  15-Dec-17          2008  TV-MA       143
2   5-Jan-19          2016  TV-PG       124
3   1-Mar-16          2016      R        90
4   3-Dec-18          2017   TV-G         1

                                          genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers
2        Dramas, International Movies, Sports Movies
3                                         Comedies
4  British TV Shows, Docuseries, Science & Nature TV

                                        description
0  In a future where the elite inhabit an island …
1  An architect and his wife move into a castle t…
2  Three Indonesian women break records by becomi…
3  New NFL star Thad buys his old teammates' belo…
4  This sequel to the award-winning nature series…
```

[110]:
```
nf.tail()
```

[110]:
```
     show_id     type                     title             director  \
7782    s995  TV Show                 Blown Away                  NaN
7783    s996  TV Show              Blue Exorcist                  NaN
7784    s997    Movie  Blue Is the Warmest Color  Abdellatif Kechiche
7785    s998    Movie               Blue Jasmine          Woody Allen
7786    s999    Movie                   Blue Jay         Alex Lehmann

                                                cast  \
7782                                             NaN
7783  Nobuhiko Okamoto, Jun Fukuyama, Kana Hanazawa,…
7784  Léa Seydoux, Adèle Exarchopoulos, Salim Kechio…
7785  Cate Blanchett, Sally Hawkins, Alec Baldwin, L…
7786           Sarah Paulson, Mark Duplass, Clu Gulager

                    country date_added  release_year rating  duration  \
7782                 Canada  12-Jul-19          2019  TV-14         1
7783                  Japan   1-Sep-20          2017  TV-MA         2
7784  France, Belgium, Spain  26-Aug-16          2013  NC-17       180
```

```
7785          United States   8-Mar-19          2013  PG-13        98
7786          United States   6-Dec-16          2016  TV-MA        81

                                                 genres  \
7782               International TV Shows, Reality TV
7783             Anime Series, International TV Shows
7784  Dramas, Independent Movies, International Movies
7785             Comedies, Dramas, Independent Movies
7786       Dramas, Independent Movies, Romantic Movies

                                           description
7782  Ten master artists turn up the heat in glassbl…
7783  Determined to throw off the curse of being Sat…
7784  Determined to fall in love, 15-year-old Adele …
7785  The high life leads to high anxiety for a fash…
7786  Two former high school sweethearts unexpectedl…
```

[111]:
```
nf.shape
```

[111]:
```
(7787, 12)
```

[112]:
```
nf.size
```

[112]:
```
93444
```

[113]:
```
nf.columns
```

[113]:
```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'genres', 'description'],
      dtype='object')
```

[114]:
```
nf.dtypes
```

[114]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration         int64
genres          object
description     object
dtype: object
```

[115]:
```
nf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       7787 non-null   object
 1   type          7787 non-null   object
 2   title         7787 non-null   object
 3   director      5398 non-null   object
 4   cast          7069 non-null   object
 5   country       7280 non-null   object
 6   date_added    7777 non-null   object
 7   release_year  7787 non-null   int64 
 8   rating        7780 non-null   object
 9   duration      7787 non-null   int64 
 10  genres        7787 non-null   object
 11  description   7787 non-null   object
dtypes: int64(2), object(10)
memory usage: 730.2+ KB
```

2. Data Cleaning

[116]:
```
nf.duplicated()
```

[116]:
```
0       False
1       False
2       False
3       False
4       False
        …
7782    False
7783    False
7784    False
7785    False
7786    False
Length: 7787, dtype: bool
```

[117]:
```
nf[nf.duplicated()]
```

[117]:
```
Empty DataFrame
Columns: [show_id, type, title, director, cast, country, date_added,
release_year, rating, duration, genres, description]
Index: []
```

[118]:
```
#nf.drop_duplicates(inplace=True)
```

[119]:
```
nf.isnull()
```

[119]:
```
      show_id   type  title  director   cast  country  date_added  \
0       False  False  False      True  False    False       False
1       False  False  False     False  False    False       False
2       False  False  False     False  False    False       False
3       False  False  False     False  False    False       False
4       False  False  False      True  False    False       False
…         …    …      …         …     …        …           …
7782    False  False  False      True   True    False       False
7783    False  False  False      True  False    False       False
7784    False  False  False     False  False    False       False
7785    False  False  False     False  False    False       False
7786    False  False  False     False  False    False       False

      release_year  rating  duration  genres  description
0            False   False     False   False        False
1            False   False     False   False        False
2            False   False     False   False        False
3            False   False     False   False        False
4            False   False     False   False        False
…              …     …         …       …            …
7782         False   False     False   False        False
7783         False   False     False   False        False
7784         False   False     False   False        False
7785         False   False     False   False        False
7786         False   False     False   False        False

[7787 rows x 12 columns]
```

[120]:
```
nf.isnull().sum()
```

[120]:
```
show_id           0
type              0
title             0
director       2389
cast            718
country         507
date_added       10
release_year      0
rating            7
duration          0
genres            0
description       0
dtype: int64
```

[121]:
```
nf[nf.director.isnull()]
```

[121]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	genres
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R…	Brazil	14-Aug-20	2020	TV-MA	4	International TV Shows, TV Dramas, TV Sci-Fi &…
4	s1001	TV Show	Blue Planet II	NaN	David Attenborough	United Kingdom	3-Dec-18	2017	TV-G	1	British TV Shows, Docuseries, Science & Nature TV
10	s1007	TV Show	BNA	NaN	Sumire Morohoshi, Yoshimasa Hosoya, Maria Naga…	Japan	30-Jun-20	2020	TV-14	1	Anime Series, International TV Shows
15	s1011	TV Show	Bo on the Go!	NaN	Catherine O'Connor, Andrew Sabiston, Jim Fowler	Canada	21-Mar-19	2007	TV-Y	1	Kids' TV
17	s1013	TV Show	Bob Ross: Beauty Is Everywhere	NaN	Bob Ross	NaN	1-Jun-16	1991	TV-G	1	TV Shows
…	…	…	…	…	…	…	…	…	…	…	…
7777	s990	TV Show	Blood Pact	NaN	Guilherme Fontes, Ravel Cabral, Jonathan Haage…	Brazil	10-Oct-18	2018	TV-MA	1	Crime TV Shows, International TV Shows, TV Dramas
7779	s992	TV Show	Bloodline	NaN	Kyle Chandler, Ben Mendelsohn, Sissy Spacek, L…	United States	26-May-17	2017	TV-MA	3	
7780	s993	TV Show	Bloodride	NaN	Ine Marie Wilmann, Bjørnar Teigen, Emma Spetal…	Norway	13-Mar-20	2020	TV-MA	1	
7782	s995	TV Show	Blown Away	NaN	NaN	Canada	12-Jul-19	2019	TV-14	1	
7783	s996	TV Show	Blue Exorcist	NaN	Nobuhiko Okamoto, Jun Fukuyama, Kana Hanazawa,…	Japan	1-Sep-20	2017	TV-MA	2	

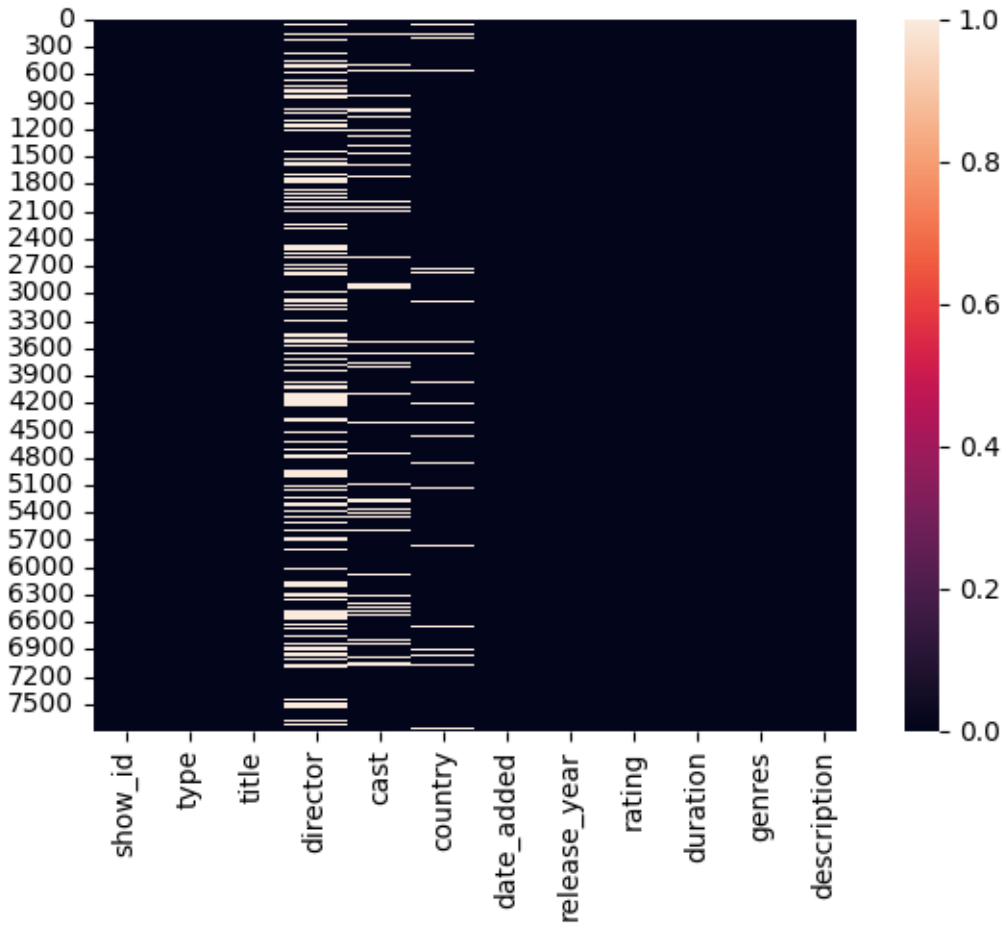
```
7779              TV Dramas, TV Mysteries, TV Thrillers
7780    International TV Shows, TV Horror, TV Mysteries
7782                 International TV Shows, Reality TV
7783               Anime Series, International TV Shows

                                                description
0     In a future where the elite inhabit an island …
4     This sequel to the award-winning nature series…
10    Morphed into a raccoon beastman, Michiru seeks…
15    Staying at home doesn't mean sitting still for…
17    "The Joy of Painting" host Bob Ross brings his…
…                                                   …
7777  An ambitious TV reporter uses risky and ethica…
7779  When the black sheep son of a respected family…
7780  The doomed passengers aboard a spectral bus he…
7782  Ten master artists turn up the heat in glassbl…
7783  Determined to throw off the curse of being Sat…

[2389 rows x 12 columns]
```

Seasborn Library (Heat-map)

[122]:
```
import seaborn as sns
```

[123]:
```
sns.heatmap(nf.isnull())      #To show null values count
```

[123]:
```
<Axes: >
```



Q1. For 'House of Cards', What is the Show Id and Who is the Director of this Show?

[124]:
```
nf[nf['title'].isin(['House of Cards'])]
```

[124]:
```
     show_id     type           title director  \
2038   s2833  TV Show  House of Cards      NaN

                                                 cast        country  \
2038  Kevin Spacey, Robin Wright, Kate Mara, Corey S…  United States

     date_added  release_year rating  duration                   genres  \
2038   2-Nov-18          2018  TV-MA         6  TV Dramas, TV Thrillers

                                          description
2038  A ruthless politician will stop at nothing to …
```

[125]:
```
nf[nf['title'].str.contains('House of Cards')]
```

[125]:
```
      show_id     type           title director  \
2038   s2833  TV Show  House of Cards      NaN

                                                  cast        country  \
2038  Kevin Spacey, Robin Wright, Kate Mara, Corey S…  United States

     date_added  release_year rating  duration                   genres  \
2038   2-Nov-18          2018  TV-MA         6  TV Dramas, TV Thrillers

                                          description
2038  A ruthless politician will stop at nothing to …
```

Q2. In which year the highest number of TV Shows and Movies were released?

[126]:
```
nf.dtypes
```

[126]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration         int64
genres          object
description     object
dtype: object
```

[127]:
```
#nf['Date_N']=pd.to_datetime(nf['release_year'])
```

[128]:
```
nf['Date_N'] = pd.to_datetime(nf['release_year'], format='%Y')
```

[129]:
```
nf['Year'] = nf['Date_N'].dt.year
```

[130]:
```
nf.head()
```

[130]:
```
  show_id     type                                  title        director  \
0      s1  TV Show                                     3%             NaN
1     s10    Movie                                   1920    Vikram Bhatt
2    s100    Movie                            3 Heroines  Iman Brotoseno
3   s1000    Movie  Blue Mountain State: The Rise of Thadland    Lev L. Spiro
4   s1001  TV Show                            Blue Planet II             NaN

                                              cast         country  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…          Brazil
```

```
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…           India
2  Reza Rahadian, Bunga Citra Lestari, Tara Basro…       Indonesia
3  Alan Ritchson, Darin Brooks, James Cade, Rob R…   United States
4                               David Attenborough  United Kingdom

  date_added  release_year rating  duration  \
0  14-Aug-20          2020  TV-MA         4
1  15-Dec-17          2008  TV-MA       143
2   5-Jan-19          2016  TV-PG       124
3   1-Mar-16          2016      R        90
4   3-Dec-18          2017   TV-G         1

                                          genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers
2        Dramas, International Movies, Sports Movies
3                                        Comedies
4  British TV Shows, Docuseries, Science & Nature TV

                                        description     Date_N  Year
0  In a future where the elite inhabit an island … 2020-01-01  2020
1  An architect and his wife move into a castle t… 2008-01-01  2008
2  Three Indonesian women break records by becomi… 2016-01-01  2016
3  New NFL star Thad buys his old teammates' belo… 2016-01-01  2016
4  This sequel to the award-winning nature series… 2017-01-01  2017
```

[131]:
```
nf.dtypes
```

[131]:
```
show_id                 object
type                    object
title                   object
director                object
cast                    object
country                 object
date_added              object
release_year             int64
rating                  object
duration                 int64
genres                  object
description             object
Date_N          datetime64[ns]
Year                     int32
dtype: object
```

[132]:
```
nf.drop('Date_N', axis=1, inplace=True)
```

[133]:
```
nf.dtypes
```

[133]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration         int64
genres          object
description     object
Year             int32
dtype: object
```

[134]:
```
nf['Year'] = pd.to_datetime(nf['Year'], format='%Y')
```

[135]:
```
nf['Year'].info()
```

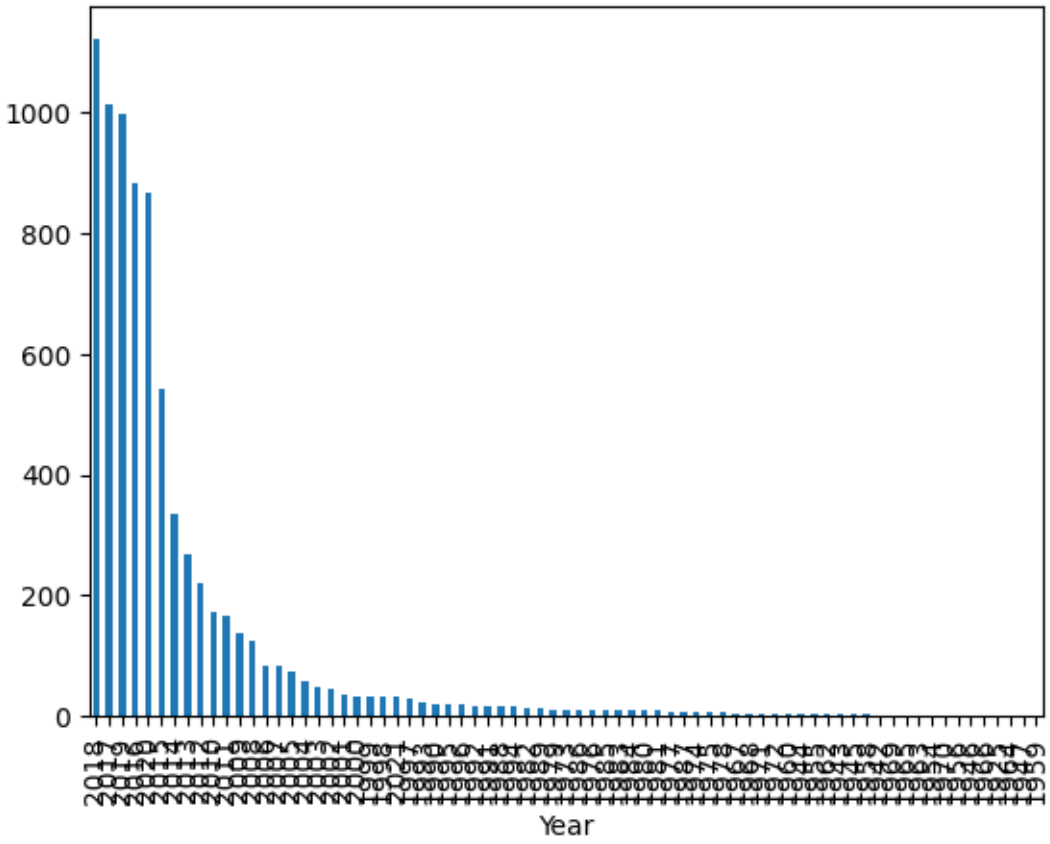
```
<class 'pandas.core.series.Series'>
RangeIndex: 7787 entries, 0 to 7786
Series name: Year
Non-Null Count  Dtype
--------------  -----
7787 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 61.0 KB
```

[136]:
```
nf['Year'].dt.year.value_counts()
```

[136]:
```
Year
2018    1121
2017    1012
2019     996
2016     882
2020     868
        …
1966       1
1925       1
1964       1
1947       1
1959       1
Name: count, Length: 73, dtype: int64
```

[137]:
```
nf['Year'].dt.year.value_counts().plot(kind='bar')
```

[137]:
```
<Axes: xlabel='Year'>
```



Q3. How many Movies and TV Shows are there in the dataset?

[138]:
```
nf.head(2)
```

[138]:
```
  show_id     type title      director  \
0      s1  TV Show    3%           NaN
1     s10    Movie  1920  Vikram Bhatt

                                                cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143

                                             genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers
```

```
                                          description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

[139]:
```
nf.groupby('type').type.count()
```

[139]:
```
type
Movie      5377
TV Show    2410
Name: type, dtype: int64
```

[140]:
```
nf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       7787 non-null   object
 1   type          7787 non-null   object
 2   title         7787 non-null   object
 3   director      5398 non-null   object
 4   cast          7069 non-null   object
 5   country       7280 non-null   object
 6   date_added    7777 non-null   object
 7   release_year  7787 non-null   int64
 8   rating        7780 non-null   object
 9   duration      7787 non-null   int64
 10  genres        7787 non-null   object
 11  description   7787 non-null   object
 12  Year          7787 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(10)
memory usage: 791.0+ KB
```

[141]:
```
nf.head(2)
```

[141]:
```
  show_id     type title      director  \
0      s1  TV Show    3%           NaN
1     s10    Movie  1920  Vikram Bhatt

                                                cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143
```

```
                                                  genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers

                                           description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

[142]:
```
nf.rename(columns={'type': 'category'}, inplace=True)
```

[143]:
```
nf.head(2)
```

[143]:
```
   show_id category title      director  \
0       s1  TV Show    3%           NaN
1      s10    Movie  1920  Vikram Bhatt

                                                 cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143

                                                  genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers

                                           description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

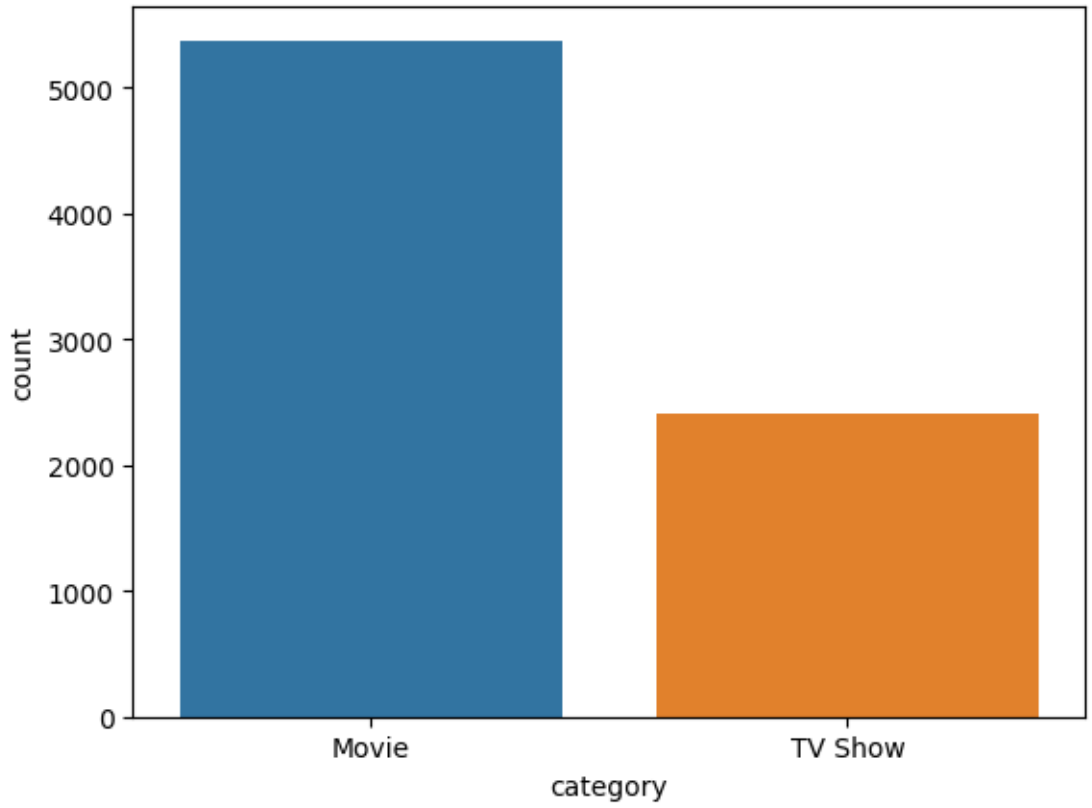
[144]:
```
nf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       7787 non-null   object
 1   category      7787 non-null   object
 2   title         7787 non-null   object
 3   director      5398 non-null   object
 4   cast          7069 non-null   object
 5   country       7280 non-null   object
 6   date_added    7777 non-null   object
```

```
 7   release_year  7787 non-null   int64
 8   rating        7780 non-null   object
 9   duration      7787 non-null   int64
 10  genres        7787 non-null   object
 11  description   7787 non-null   object
 12  Year          7787 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(10)
memory usage: 791.0+ KB
```

[145]:
```
nf.rename(columns={'type': 'category'}, inplace=True)
nf['category'] = nf['category'].astype('category')
sns.countplot(data=nf, x='category')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[145], line 4
      2 nf['category'] = nf['category'].astype('category')
      3 sns.countplot(data=nf, x='category')
----> 4 plt.show()

NameError: name 'plt' is not defined
```



Q4. Show all the movies that were released in the year 2000.

[146]:
```
nf.head()
```

[146]:
```
  show_id category                                        title        director  \
0      s1  TV Show                                           3%             NaN
1     s10    Movie                                         1920    Vikram Bhatt
2    s100    Movie                                  3 Heroines  Iman Brotoseno
3   s1000    Movie  Blue Mountain State: The Rise of Thadland    Lev L. Spiro
4   s1001  TV Show                               Blue Planet II             NaN

                                                cast         country  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…          Brazil
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…           India
2  Reza Rahadian, Bunga Citra Lestari, Tara Basro…       Indonesia
3  Alan Ritchson, Darin Brooks, James Cade, Rob R…   United States
4                              David Attenborough  United Kingdom

  date_added  release_year rating  duration  \
0  14-Aug-20          2020  TV-MA         4
1  15-Dec-17          2008  TV-MA       143
2   5-Jan-19          2016  TV-PG       124
3   1-Mar-16          2016      R        90
4   3-Dec-18          2017   TV-G         1

                                          genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers
2        Dramas, International Movies, Sports Movies
3                                         Comedies
4  British TV Shows, Docuseries, Science & Nature TV

                                         description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
2  Three Indonesian women break records by becomi… 2016-01-01
3  New NFL star Thad buys his old teammates' belo… 2016-01-01
4  This sequel to the award-winning nature series… 2017-01-01
```

[147]:
```
nf.head(2)
```

[147]:
```
  show_id category title      director  \
0      s1  TV Show    3%           NaN
1     s10    Movie  1920  Vikram Bhatt
```

```
                                                  cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143

                                              genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers

                                         description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

[148]:
```
nf[(nf['category']=='Movie') & (nf['Year']==2000)]
```

[148]:
```
Empty DataFrame
Columns: [show_id, category, title, director, cast, country, date_added,
release_year, rating, duration, genres, description, Year]
Index: []
```

Q5. Show only the Titles of all TV Shows that were released only in India.

[149]:
```
nf.head(2)
```

[149]:
```
  show_id category title      director  \
0      s1  TV Show    3%           NaN
1     s10    Movie  1920  Vikram Bhatt

                                                  cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143

                                              genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers

                                         description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

[150]:
```
nf[(nf['category']=='TV Show') & (nf['country']=='India')]['title']
```

[150]:
```
354                                        Chhota Bheem
368                                           7 (Seven)
421     ChuChu TV Nursery Rhymes & Kids Songs (Hindi)
461                                     Classic Legends
517                                     College Romance
                              …
7629                                            Betaal
7643                      21 Sarfarosh: Saragarhi 1897
7655                                         Bh Se Bhade
7656                                  Bhaag Beanie Bhaag
7657                                      Bhaage Re Mann
Name: title, Length: 71, dtype: object
```

Q6. Show Top 10 Directors, who gave the highest number of TV Shows & Movies

[151]:
```
nf['director'].value_counts()
```

[151]:
```
director
Raúl Campos, Jan Suter    18
Marcus Raboy              16
Jay Karas                 14
Cathy Garcia-Molina       13
Youssef Chahine           12
                          ..
Mariano Baez               1
Ravikanth Perepu           1
Santram Varma              1
Oliver Blackburn           1
Woody Allen                1
Name: count, Length: 4049, dtype: int64
```

[152]:
```
nf['director'].value_counts().head(10)
```

[152]:
```
director
Raúl Campos, Jan Suter    18
Marcus Raboy              16
Jay Karas                 14
Cathy Garcia-Molina       13
Youssef Chahine           12
Martin Scorsese           12
Jay Chapman               12
Steven Spielberg          10
David Dhawan               9
Johnnie To                 8
Name: count, dtype: int64
```

Q8. Show all the Records, where "category is movie and type is comdies" or "country is United Kingdom

[153]:
```
nf.head(2)
```

[153]:
```
  show_id category title      director  \
0      s1  TV Show    3%           NaN
1     s10    Movie  1920  Vikram Bhatt

                                                cast country date_added  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…  Brazil  14-Aug-20
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…   India  15-Dec-17

   release_year rating  duration  \
0          2020  TV-MA         4
1          2008  TV-MA       143

                                          genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers

                                       description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
```

[154]:
```
nf[(nf['category']=='Movie') & (nf['genres']=='Comedies') |␣
 ↪(nf['country']=='United Kingdom')]
```

[154]:
```
     show_id category                                        title  \
3      s1000    Movie  Blue Mountain State: The Rise of Thadland
4      s1001  TV Show                               Blue Planet II
25     s1020    Movie          Bobby Robson: More Than a Manager
56     s1049  TV Show                                   Borderline
71     s1062  TV Show                             Botched Up Bodies
…      …      …                                             …
7702    s922    Movie                      Bill Hicks: Relentless
7703    s923    Movie                     Bill Hicks: Revelations
7721     s94    Movie                           27: Gone Too Soon
7733    s950  TV Show                          Black Earth Rising
7740    s957  TV Show                                Black Mirror

                           director  \
3                      Lev L. Spiro
4                               NaN
25    Gabriel Clarke, Torquil Jones
56                              NaN
71                              NaN
```

```
…                            …
7702                   Chris Bould
7703                   Chris Bould
7721             Simon Napier-Bell
7733                           NaN
7740                           NaN

                                                    cast         country  \
3     Alan Ritchson, Darin Brooks, James Cade, Rob R…   United States
4                                   David Attenborough  United Kingdom
25                                        Bobby Robson  United Kingdom
56    David Avery, Jackie Clune, David Elms, Liz Kin…  United Kingdom
71                         Charlie Brooks, Sue Johnston  United Kingdom
…                                                   …             …
7702                                        Bill Hicks  United Kingdom
7703                                        Bill Hicks  United Kingdom
7721  Janis Joplin, Jimi Hendrix, Amy Winehouse, Jim…  United Kingdom
7733  Michaela Coel, John Goodman, Abena Ayivor, Nom…  United Kingdom
7740  Jesse Plemons, Cristin Milioti, Jimmi Simpson,…  United Kingdom

             date_added  release_year rating  duration  \
3              1-Mar-16          2016      R        90
4              3-Dec-18          2017   TV-G         1
25             1-Dec-18          2018  TV-14        99
56      December 2, 2017          2017  TV-14         2
71             2-Oct-19          2013  TV-MA         1
…                 …            …    …       …
7702          31-Dec-18          1992  TV-MA        61
7703          31-Dec-18          1993  TV-MA        56
7721           1-May-18          2017  TV-MA        70
7733          25-Jan-19          2018  TV-MA         1
7740           5-Jun-19          2019  TV-MA         5

                                                  genres  \
3                                                Comedies
4     British TV Shows, Docuseries, Science & Nature TV
25    Documentaries, International Movies, Sports Mo…
56                        British TV Shows, TV Comedies
71    British TV Shows, Docuseries, International TV…
…                                                   …
7702                                 Stand-Up Comedy
7703                                 Stand-Up Comedy
7721                                   Documentaries
7733  British TV Shows, International TV Shows, TV D…
7740  British TV Shows, International TV Shows, TV D…

                                          description       Year
```

```
3     New NFL star Thad buys his old teammates' belo… 2016-01-01
4     This sequel to the award-winning nature series… 2017-01-01
25    Explore the life and times of legendary soccer… 2018-01-01
56    A team of inept border agents tackles immigrat… 2017-01-01
71    From lifting saggy skin to repairing shoddy br… 2013-01-01
…                                                   …          …
7702  In one of his most iconic performances, late c… 1992-01-01
7703  In his final recorded special, the iconoclasti… 1993-01-01
7721  Explore the circumstances surrounding the trag… 2017-01-01
7733  Adopted by a human rights attorney after the R… 2018-01-01
7740  This sci-fi anthology series explores a twiste… 2019-01-01

[485 rows x 13 columns]
```

[155]:
```
nf.isnull().sum()
```

[155]:
```
show_id            0
category           0
title              0
director        2389
cast             718
country          507
date_added        10
release_year       0
rating             7
duration           0
genres             0
description        0
Year               0
dtype: int64
```

Missing Values using mean

[156]:
```
nf.head()
```

[156]:
```
  show_id category                                     title        director  \
0      s1  TV Show                                       3%             NaN
1     s10    Movie                                     1920    Vikram Bhatt
2    s100    Movie                                3 Heroines  Iman Brotoseno
3   s1000    Movie  Blue Mountain State: The Rise of Thadland    Lev L. Spiro
4   s1001  TV Show                           Blue Planet II             NaN

                                                cast         country  \
0  João Miguel, Bianca Comparato, Michel Gomes, R…          Brazil
1  Rajneesh Duggal, Adah Sharma, Indraneil Sengup…           India
2  Reza Rahadian, Bunga Citra Lestari, Tara Basro…       Indonesia
3  Alan Ritchson, Darin Brooks, James Cade, Rob R…   United States
```

```
4                                David Attenborough  United Kingdom

  date_added  release_year rating  duration  \
0  14-Aug-20          2020  TV-MA         4
1  15-Dec-17          2008  TV-MA       143
2   5-Jan-19          2016  TV-PG       124
3   1-Mar-16          2016      R        90
4   3-Dec-18          2017   TV-G         1

                                              genres  \
0  International TV Shows, TV Dramas, TV Sci-Fi &…
1     Horror Movies, International Movies, Thrillers
2        Dramas, International Movies, Sports Movies
3                                          Comedies
4  British TV Shows, Docuseries, Science & Nature TV

                                         description       Year
0  In a future where the elite inhabit an island … 2020-01-01
1  An architect and his wife move into a castle t… 2008-01-01
2  Three Indonesian women break records by becomi… 2016-01-01
3  New NFL star Thad buys his old teammates' belo… 2016-01-01
4  This sequel to the award-winning nature series… 2017-01-01
```

[157]:
```
nf['rating'].dropna()
```

[157]:
```
0       TV-MA
1       TV-MA
2       TV-PG
3           R
4        TV-G
        …
7782    TV-14
7783    TV-MA
7784    NC-17
7785    PG-13
7786    TV-MA
Name: rating, Length: 7780, dtype: object
```

[158]:
```
nf.dropna( how='any',subset=['cast', 'director'])
```

[158]:
```
     show_id category                                     title  \
1        s10    Movie                                      1920
2       s100    Movie                                 3 Heroines
3      s1000    Movie  Blue Mountain State: The Rise of Thadland
5      s1002    Movie                                  Blue Ruin
6      s1003    Movie                                Blue Streak
…       …       …                                         …
```

```
7778    s991    Movie                           Blood Will Tell
7781    s994    Movie                                      Blow
7784    s997    Movie                 Blue Is the Warmest Color
7785    s998    Movie                              Blue Jasmine
7786    s999    Movie                                  Blue Jay

                      director  \
1                  Vikram Bhatt
2                 Iman Brotoseno
3                   Lev L. Spiro
5                Jeremy Saulnier
6                   Les Mayfield
…                             …
7778  Miguel Cohan, Miguel Cohan
7781                   Ted Demme
7784         Abdellatif Kechiche
7785                 Woody Allen
7786                Alex Lehmann

                                                 cast  \
1     Rajneesh Duggal, Adah Sharma, Indraneil Sengup…
2     Reza Rahadian, Bunga Citra Lestari, Tara Basro…
3     Alan Ritchson, Darin Brooks, James Cade, Rob R…
5     Macon Blair, Devin Ratray, Amy Hargreaves, Kev…
6     Martin Lawrence, Luke Wilson, Peter Greene, Da…
…                                                   …
7778  Oscar Martínez, Dolores Fonzi, Diego Velázquez…
7781  Johnny Depp, Penélope Cruz, Franka Potente, Ra…
7784  Léa Seydoux, Adèle Exarchopoulos, Salim Kechio…
7785  Cate Blanchett, Sally Hawkins, Alec Baldwin, L…
7786           Sarah Paulson, Mark Duplass, Clu Gulager

                      country date_added  release_year rating  duration  \
1                       India  15-Dec-17          2008  TV-MA       143
2                   Indonesia   5-Jan-19          2016  TV-PG       124
3               United States   1-Mar-16          2016      R        90
5        United States, France  25-Feb-19          2013      R        90
6       Germany, United States   1-Jan-21          1999  PG-13        94
…                          …          …           …     …       …
7778  Argentina, United States  21-Jun-19          2019  TV-MA       113
7781            United States   1-Oct-19          2001      R       123
7784    France, Belgium, Spain  26-Aug-16          2013  NC-17       180
7785            United States   8-Mar-19          2013  PG-13        98
7786            United States   6-Dec-16          2016  TV-MA        81

                                              genres  \
1       Horror Movies, International Movies, Thrillers
```

```
2          Dramas, International Movies, Sports Movies
3                                                Comedies
5                        Independent Movies, Thrillers
6                         Action & Adventure, Comedies
…                                                    …
7778  Dramas, Independent Movies, International Movies
7781                                             Dramas
7784  Dramas, Independent Movies, International Movies
7785              Comedies, Dramas, Independent Movies
7786       Dramas, Independent Movies, Romantic Movies

                                              description       Year
1     An architect and his wife move into a castle t… 2008-01-01
2     Three Indonesian women break records by becomi… 2016-01-01
3     New NFL star Thad buys his old teammates' belo… 2016-01-01
5     Bad news from the past unhinges vagabond Dwigh… 2013-01-01
6     A jewel thief returns to his hiding place afte… 1999-01-01
…                                                    …        …
7778  Family patriarch Elías begins to unravel after… 2019-01-01
7781  Cocaine smuggler George rises from poverty to … 2001-01-01
7784  Determined to fall in love, 15-year-old Adele … 2013-01-01
7785  The high life leads to high anxiety for a fash… 2013-01-01
7786  Two former high school sweethearts unexpectedl… 2016-01-01

[4979 rows x 13 columns]
```

[159]:
```
nf['country'].fillna('Missing',inplace=True)
nf['date_added'].fillna('Missing',inplace=True)
nf['rating'].fillna('Missing',inplace=True)
nf.isnull().sum().sum()
```

[159]: 3107

[160]:
```
nf['director'].fillna('Unknown', inplace=True)
```

[161]:
```
nf['cast'].fillna('Unknown', inplace=True)
```

[162]:
```
nf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       7787 non-null   object
 1   category      7787 non-null   category
 2   title         7787 non-null   object
```

```
 3   director      7787 non-null   object
 4   cast          7787 non-null   object
 5   country       7787 non-null   object
 6   date_added    7787 non-null   object
 7   release_year  7787 non-null   int64
 8   rating        7787 non-null   object
 9   duration      7787 non-null   int64
 10  genres        7787 non-null   object
 11  description   7787 non-null   object
 12  Year          7787 non-null   datetime64[ns]
dtypes: category(1), datetime64[ns](1), int64(2), object(9)
memory usage: 737.9+ KB
```

[ ]:

world-university-ranking

December 11, 2023

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[3]:
```
df=pd.read_csv(r"C:\Users\Md Raza Ashraf\Downloads\archive\World University␣
↪Rankings 2023.csv")
```

[5]:
```
df
```

[5]:

	University Rank	Name of University	Location	\
0	1	University of Oxford	United Kingdom	
1	2	Harvard University	United States	
2	3	University of Cambridge	United Kingdom	
3	3	Stanford University	United States	
4	5	Massachusetts Institute of Technology	United States	
…	…	…	…	
2336	-	University of the West of Scotland	NaN	
2337	-	University of Windsor	NaN	
2338	-	University of Wolverhampton	NaN	
2339	-	University of Wuppertal	NaN	
2340	-	Xi’an Jiaotong-Liverpool University	NaN	

	No of student	No of student per staff	International Student	\
0	20,965	10.6	42%	
1	21,887	9.6	25%	
2	20,185	11.3	39%	
3	16,164	7.1	24%	
4	11,415	8.2	33%	
…	…	…	…	
2336	NaN	NaN	NaN	
2337	NaN	NaN	NaN	
2338	NaN	NaN	NaN	
2339	NaN	NaN	NaN	
2340	NaN	NaN	NaN	

	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	\

```
0              48 : 52          96.4            92.3            99.7
1              50 : 50          95.2            94.8            99.0
2              47 : 53          94.8            90.9            99.5
3              46 : 54          94.8            94.2            96.7
4              40 : 60          94.2            90.7            93.6
…                 …             …               …               …
2336              NaN     34.0-39.2            24.1            15.5
2337              NaN     34.0-39.2            35.1            29.4
2338              NaN     34.0-39.2            18.2            14.3
2339              NaN     34.0-39.2            26.4            26.7
2340              NaN     34.0-39.2            17.8            14.8

      Citations Score  Industry Income Score  International Outlook Score
0                99.0                   74.9                         96.2
1                99.3                   49.5                         80.5
2                97.0                   54.2                         95.8
3                99.8                   65.0                         79.8
4                99.8                   90.9                         89.3
…                 …                      …                            …
2336             61.5                   37.9                         76.8
2337             34.5                   44.2                         88.7
2338             68.8                   37.3                         72.0
2339             52.8                   52.1                         47.6
2340             68.2                   38.2                         72.4

[2341 rows x 13 columns]
```

[6]:
```
df.isnull()
```

[6]:
```
      University Rank  Name of University  Location  No of student  \
0               False               False     False          False
1               False               False     False          False
2               False               False     False          False
3               False               False     False          False
4               False               False     False          False
…                 …                   …         …              …
2336            False               False      True           True
2337            False               False      True           True
2338            False               False      True           True
2339            False               False      True           True
2340            False               False      True           True

      No of student per staff  International Student  Female:Male Ratio  \
0                       False                  False              False
1                       False                  False              False
2                       False                  False              False
3                       False                  False              False
```

```
4                      False                  False              False
…                      …                      …                  …
2336                    True                   True               True
2337                    True                   True               True
2338                    True                   True               True
2339                    True                   True               True
2340                    True                   True               True

      OverAll Score  Teaching Score  Research Score  Citations Score  \
0             False           False           False            False
1             False           False           False            False
2             False           False           False            False
3             False           False           False            False
4             False           False           False            False
…             …               …               …                …
2336          False           False           False            False
2337          False           False           False            False
2338          False           False           False            False
2339          False           False           False            False
2340          False           False           False            False

      Industry Income Score  International Outlook Score
0                     False                        False
1                     False                        False
2                     False                        False
3                     False                        False
4                     False                        False
…                     …                            …
2336                  False                        False
2337                  False                        False
2338                  False                        False
2339                  False                        False
2340                  False                        False

[2341 rows x 13 columns]
```

[7]:
```
df.isnull().sum()
```

[7]:
```
University Rank                0
Name of University           108
Location                     294
No of student                132
No of student per staff      133
International Student        132
Female:Male Ratio            213
OverAll Score                542
Teaching Score               542
```

```
Research Score                 542
Citations Score                542
Industry Income Score          542
International Outlook Score    542
dtype: int64
```

[8]:
```
df.dropna(inplace=True)
```

[9]:
```
df.isnull().sum()
```

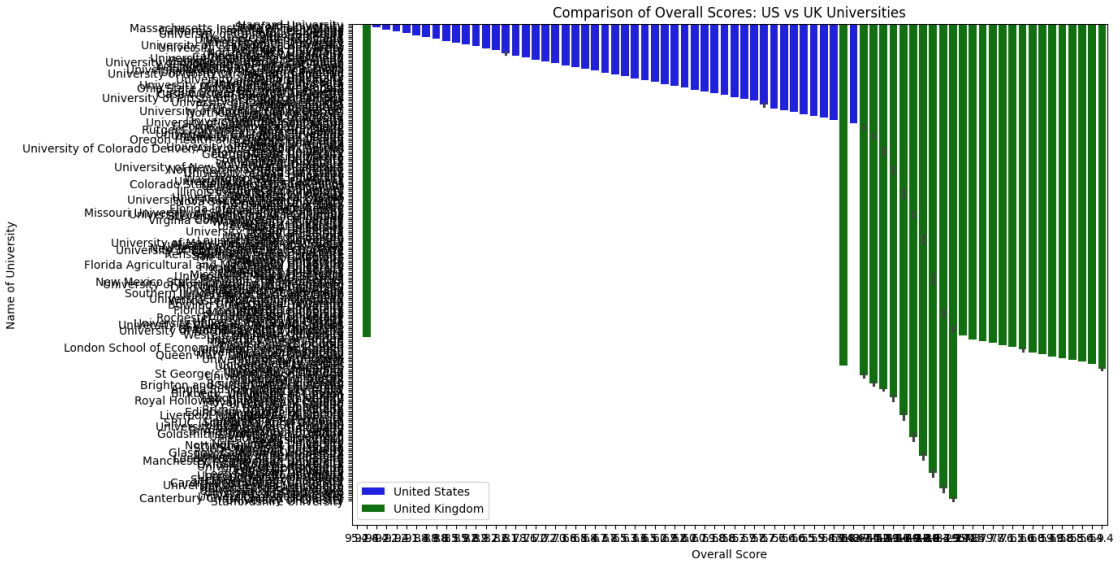
[9]:
```
University Rank                0
Name of University             0
Location                       0
No of student                  0
No of student per staff        0
International Student          0
Female:Male Ratio              0
OverAll Score                  0
Teaching Score                 0
Research Score                 0
Citations Score                0
Industry Income Score          0
International Outlook Score    0
dtype: int64
```

[12]:
```
# Find the university with the highest overall score
highest_score_university = df.loc[df['OverAll Score'].idxmax()]

# Print or display the information about the university with the highest␣
 ↪overall score
print("University with the highest overall score:")
print(highest_score_university[['Name of University', 'OverAll Score']])
```

```
University with the highest overall score:
Name of University    University of Oxford
OverAll Score                         96.4
Name: 0, dtype: object
```

[15]:
```
# How do universities in the United State compared to those in the United␣
 ↪Kingdom in terms of overall score
# Filter data for universities in the United States and the United Kingdom
us_universities = df[df['Location'] == 'United States']
uk_universities = df[df['Location'] == 'United Kingdom']

# Plot a bar chart to compare overall scores
plt.figure(figsize=(12, 8))
```

```
sns.barplot(x='OverAll Score', y='Name of University', data=us_universities,␣
 ↪color='blue', label='United States')
sns.barplot(x='OverAll Score', y='Name of University', data=uk_universities,␣
 ↪color='green', label='United Kingdom')
plt.title('Comparison of Overall Scores: US vs UK Universities')
plt.xlabel('Overall Score')
plt.ylabel('Name of University')
plt.legend()
plt.show()
```



[16]:
```
# what is the average No of student per staff ratio for the top-ranked␣
 ↪universities
# Sort the DataFrame by Overall Score in descending order
df_sorted = df.sort_values(by='OverAll Score', ascending=False)

# Select the top-ranked universities
top_universities = df_sorted.head(10)  # Adjust the number as needed

# Calculate the average No of student per staff ratio for the top-ranked␣
 ↪universities
average_ratio = top_universities['No of student per staff'].mean()

print(f"The average No of student per staff ratio for the top-ranked␣
 ↪universities is: {average_ratio:.2f}")
```

```
The average No of student per staff ratio for the top-ranked universities is:
9.65
```

[26]:
```
# which University has the highest Female:Male Ratio in above Dataset
# Convert Female:Male Ratio to a numerical value for correlation analysis
# Split the 'Female:Male Ratio' into separate columns
df[['Female Ratio', 'Male Ratio']] = df['Female:Male Ratio'].str.split(':',␣
 ↪expand=True)

# Convert the columns to numeric values
df['Female Ratio'] = pd.to_numeric(df['Female Ratio'], errors='coerce')
df['Male Ratio'] = pd.to_numeric(df['Male Ratio'], errors='coerce')

# Calculate the numerical ratio (e.g., Female Percentage / Male Percentage)
df['Numerical Ratio'] = df['Female Ratio'] / df['Male Ratio']

# Now you can use 'Numerical Ratio' for correlation analysis

# Display the DataFrame with the new numerical ratio
print(df[['University Rank', 'Name of University', 'Location', 'Female:Male␣
 ↪Ratio', 'Numerical Ratio']])
```

```
     University Rank                              Name of University  \
0                  1                             University of Oxford
1                  2                               Harvard University
2                  3                           University of Cambridge
3                  3                               Stanford University
4                  5           Massachusetts Institute of Technology
…                …                                                 …
1692           1501+  Wrocław University of Science and Technology
1693           1501+                              Yamaguchi University
1694           1501+                                Yanshan University
1695           1501+                               Yeditepe University
1696           1501+              Zonguldak Bülent Ecevit University

            Location Female:Male Ratio  Numerical Ratio
0     United Kingdom           48 : 52         0.923077
1      United States           50 : 50         1.000000
2     United Kingdom           47 : 53         0.886792
3      United States           46 : 54         0.851852
4      United States           40 : 60         0.666667
…                …                 …                …
1692          Poland           35 : 65         0.538462
1693           Japan           38 : 62         0.612903
1694           China           39 : 61         0.639344
1695          Turkey           55 : 45         1.222222
1696          Turkey           47 : 53         0.886792

[1488 rows x 5 columns]
```

[23]:
```
#is there a relationship between industry income and overall scores
# Check for missing values in the columns of interest
columns_of_interest = ['Industry Income Score', 'OverAll Score']
df[columns_of_interest].isnull().sum()

# Drop rows with missing values in the columns of interest
df_filtered = df.dropna(subset=columns_of_interest)

# Scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Industry Income Score', y='OverAll Score', data=df_filtered)
plt.title('Relationship Between Industry Income Score and Overall Score')
plt.xlabel('Industry Income Score')
plt.ylabel('Overall Score')
plt.show()
```

[25]:
```
top_ranked_universities = df[df['University Rank'] == 1]

# Count the number of top-ranked universities per country
top_ranked_by_country = top_ranked_universities['Location'].value_counts()

# Display the count of top-ranked universities per country
print(top_ranked_by_country)
```

```
Series([], Name: count, dtype: int64)
```

[ ]: