```
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

file_path="Restaurant_Reviews.tsv"
df=pd.read_csv(file_path, delimiter = '\t', quoting = 3)

df.head()
```

```
                                              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
```

```
# Shape of dataframe
df.shape
```

```
(1000, 2)
```

Q1-> How to use PortStemmer in Preprocessing

Data Preprocessing

```
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.
```

```
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)

corpus[500]
```

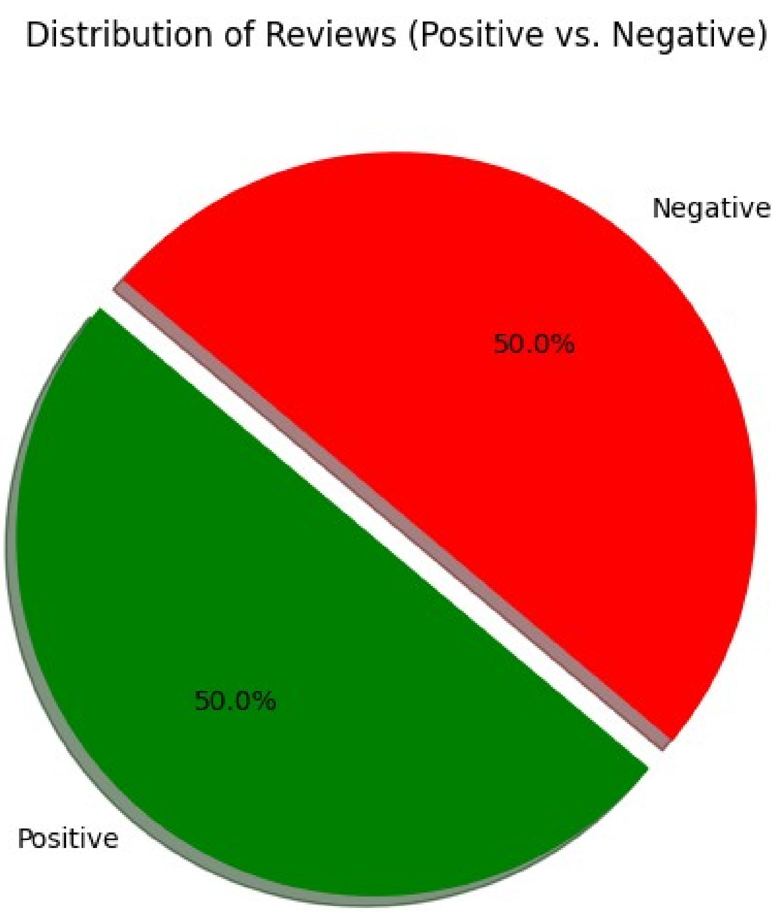
{"type":"string"}

Q2-> Top 3 most commom words in our dataset

```
# 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

```
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()
```



Data transformation

Q4->Used TF-IDF vectorization with max_features

```
# 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

Dividing dataset into training and test set

```
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)
```

altamash-ansari-22mmca023hy

December 12, 2023

[1]:
```python
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]:
```python
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]:
```python
#(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]:
```python
#(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]:
```python
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]:
```python
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]:
```python
df['Species'].unique()
```

[16]:
```
array(['Acer saccharum', 'Quercus alba', 'Quercus rubra',
       'Prunus serotina'], dtype=object)
```

[17]:
```python
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]:
```python
df['Event']=df['Event'].fillna(0)
```

[26]:
```python
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
```

```python
# import python libraries

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

# import csv file
df = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')

df.shape
```

```
(11251, 15)
```

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

```
   User_ID  Cust_name Product_ID Gender Age Group  Age  Marital_Status  \
0  1002903  Sanskriti  P00125942      F     26-35   28               0

1  1000732     Kartik  P00110942      F     26-35   35               1

2  1001990      Bindu  P00118542      F     26-35   35               1

3  1001425     Sudevi  P00237842      M      0-17   16               0

4  1000588       Joni  P00057942      M     26-35   28               1


            State      Zone       Occupation Product_Category  Orders  \
0     Maharashtra   Western       Healthcare             Auto       1

1  Andhra Pradesh  Southern             Govt             Auto       3

2   Uttar Pradesh   Central       Automobile             Auto       3

3       Karnataka  Southern     Construction             Auto       2

4         Gujarat   Western  Food Processing             Auto       2


    Amount  Status  unnamed1
0  23952.0     NaN       NaN
1  23934.0     NaN       NaN
2  23924.0     NaN       NaN
3  23912.0     NaN       NaN
4  23877.0     NaN       NaN
```

```python
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
```

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

#check for null values
pd.isnull(df).sum()
```

```
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
```

```python
# drop null values
df.dropna(inplace=True)

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

df['Amount'].dtypes
```

```
dtype('int32')
```

```
df.columns
```

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

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

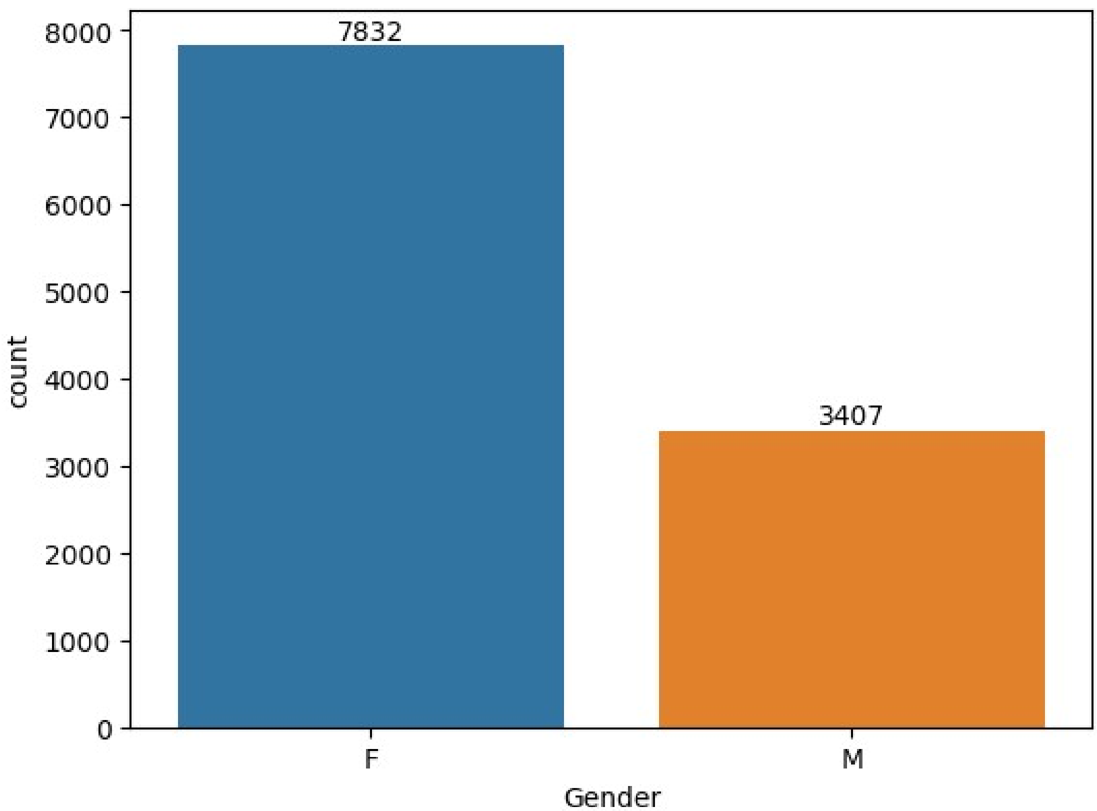
```
       User_ID    Cust_name Product_ID Gender Age Group  Age
Shaadi  \
0      1002903    Sanskriti  P00125942      F     26-35   28       0

1      1000732       Kartik  P00110942      F     26-35   35       1

2      1001990        Bindu  P00118542      F     26-35   35       1

3      1001425       Sudevi  P00237842      M      0-17   16       0

4      1000588         Joni  P00057942      M     26-35   28       1

...        ...          ...        ...    ...       ...  ...     ...

11246  1000695      Manning  P00296942      M     18-25   19       1

11247  1004089  Reichenbach  P00171342      M     26-35   33       0

11248  1001209        Oshin  P00201342      F     36-45   40       0

11249  1004023       Noonan  P00059442      M     36-45   37       0

11250  1002744      Brumley  P00281742      F     18-25   19       0


                State      Zone       Occupation Product_Category
Orders  \
0         Maharashtra   Western       Healthcare             Auto
1
1      Andhra Pradesh  Southern             Govt             Auto
3
2       Uttar Pradesh   Central       Automobile             Auto
3
3           Karnataka  Southern     Construction             Auto
2
4             Gujarat   Western  Food Processing             Auto
2
```

```
...                  ...       ...                ...               ...
...
11246     Maharashtra   Western          Chemical            Office
4
11247         Haryana  Northern        Healthcare        Veterinary
3
11248  Madhya Pradesh   Central           Textile            Office
4
11249       Karnataka  Southern       Agriculture            Office
3
11250     Maharashtra   Western        Healthcare            Office
3

       Amount
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]

# describe() method returns description of the data in the DataFrame
(i.e. count, mean, std, etc)
df.describe()

            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
```

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

	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

```python
# 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)
```

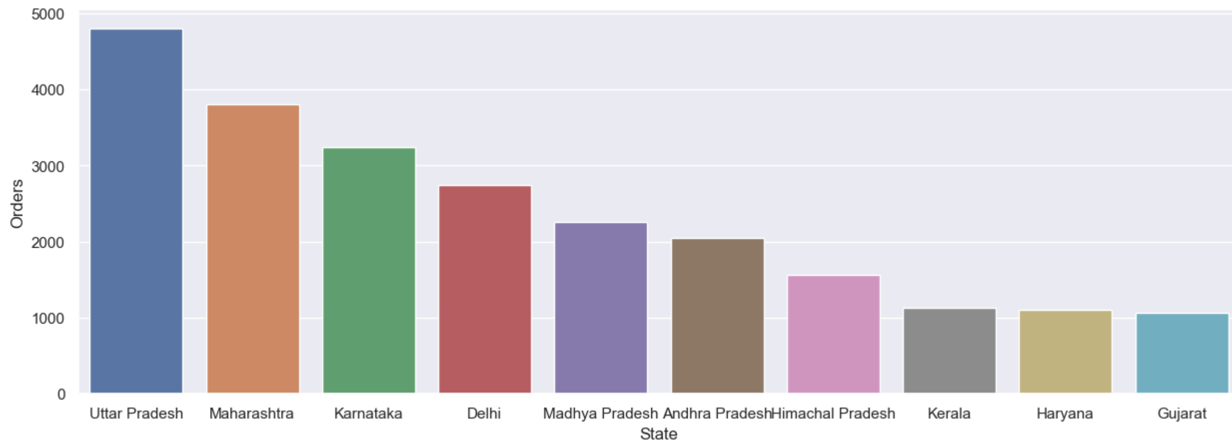


From above graphs we can see that most of the buyers are of age group between 26-35 yrs female

```
### State
```

```python
# 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')
```

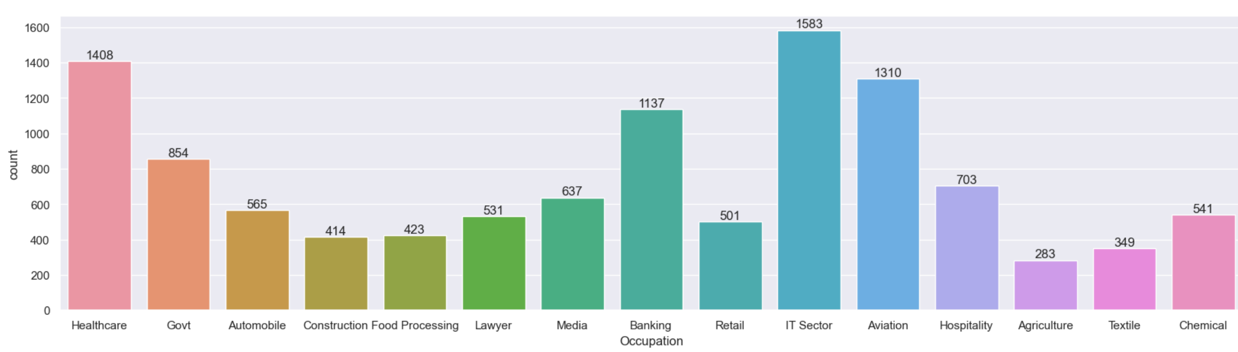
```
<Axes: xlabel='State', ylabel='Orders'>
```



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

Occupation

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

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

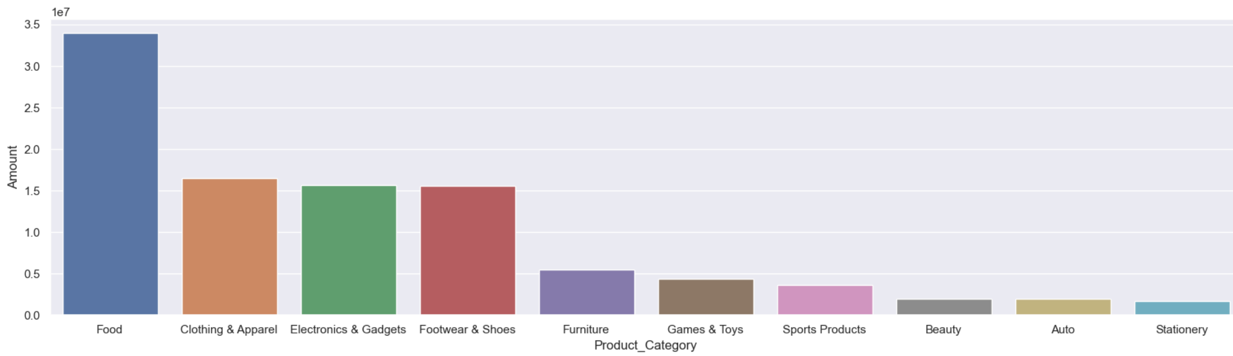


From above graphs we can see that most of the buyers are working in IT, Healthcare and Aviation sector

```python
### Product Category
```

```python
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')
```

```
<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

```
## 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!

```python
import numpy as np
import pandas as pd
data = pd.read_csv("C:/Users/ahap0/Downloads/Netflix TV Shows and Movies.csv")

data.head()
```

```
   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  
```

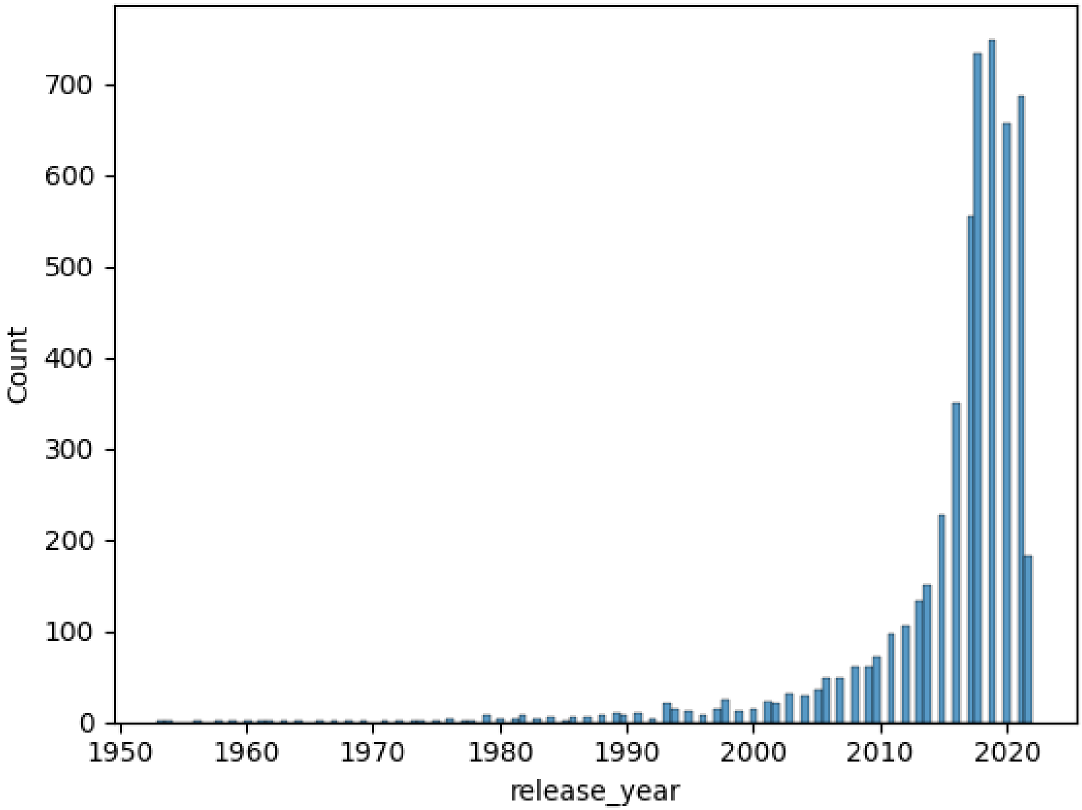
Q-1. What is the range of release years for the movies and shows, and can you identify any patterns or trends in the distribution of release years?

```python
data['release_year'].describe()
```

```
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
```

```python
import seaborn as sns
sns.histplot(data = data, x='release_year')
```

```
<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

```
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:

                                          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
```

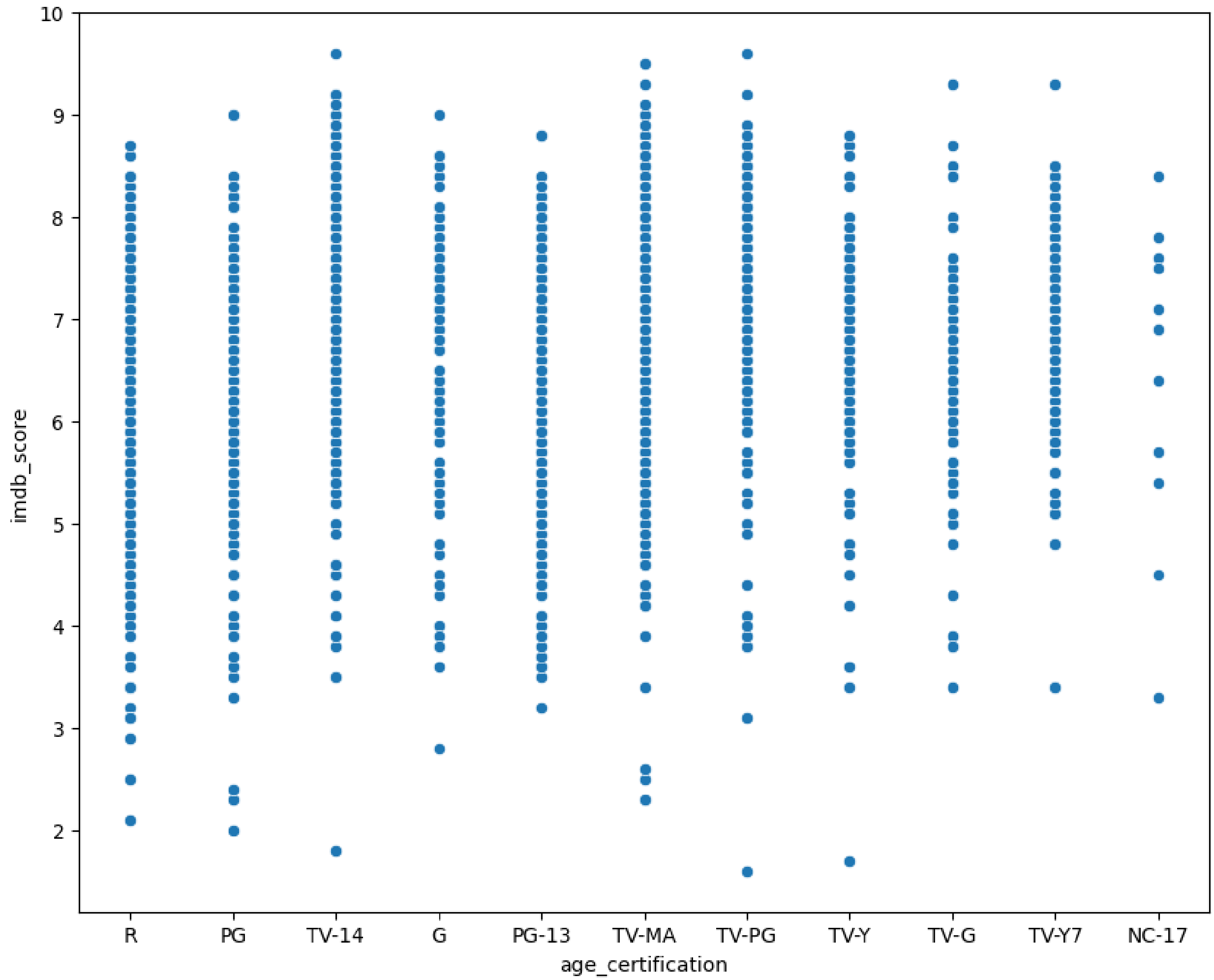
Q-3. What are the unique age certifications present in the dataset, and can you identify any relationship between IMDb score and age certification?

```
data["age_certification"].unique()
```

```
array(['R', 'PG', 'TV-14', 'G', 'PG-13', nan, 'TV-MA', 'TV-PG', 'TV-
Y',
       'TV-G', 'TV-Y7', 'NC-17'], dtype=object)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize = (10,8))
sns.scatterplot(data =data, x='age_certification', y='imdb_score')
```

```
<Axes: xlabel='age_certification', ylabel='imdb_score'>
```

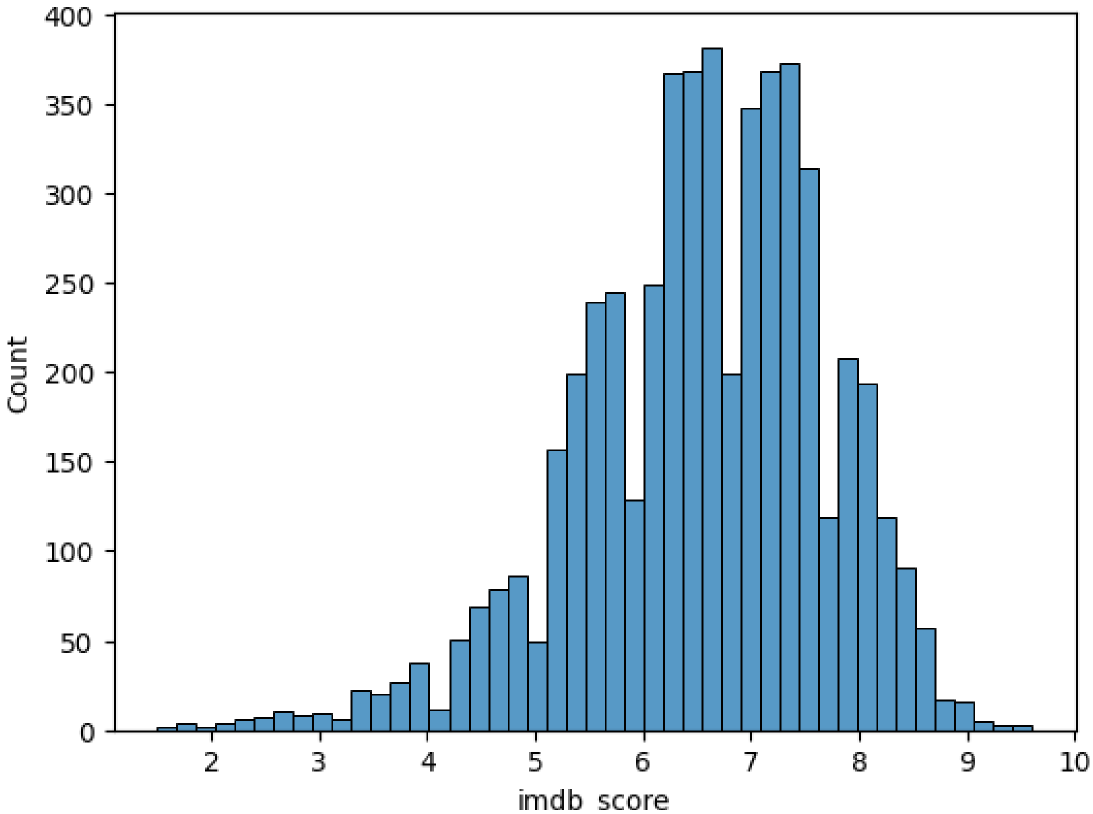


answer: the highest imdb scores went to tv-14 and tv-ma rating. The lowest went to pg rating. The ratings on TV-MA are wisespread

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.

```
sns.histplot(data = data, x='imdb_score')
```

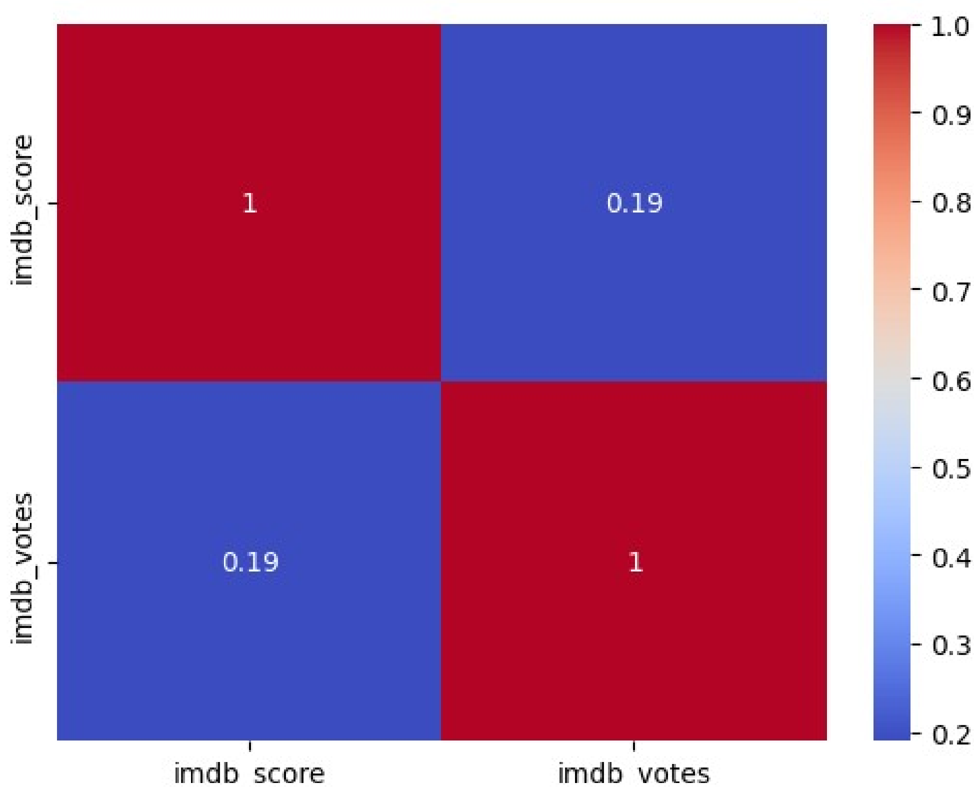
```
<Axes: xlabel='imdb_score', ylabel='Count'>
```



answer1: most imdb rating is between 6 to 7 range

```
df_temp = data[["imdb_score", "imdb_votes"]]
sns.heatmap( df_temp.corr(), cmap='coolwarm', annot=True)
```

```
<Axes: >
```

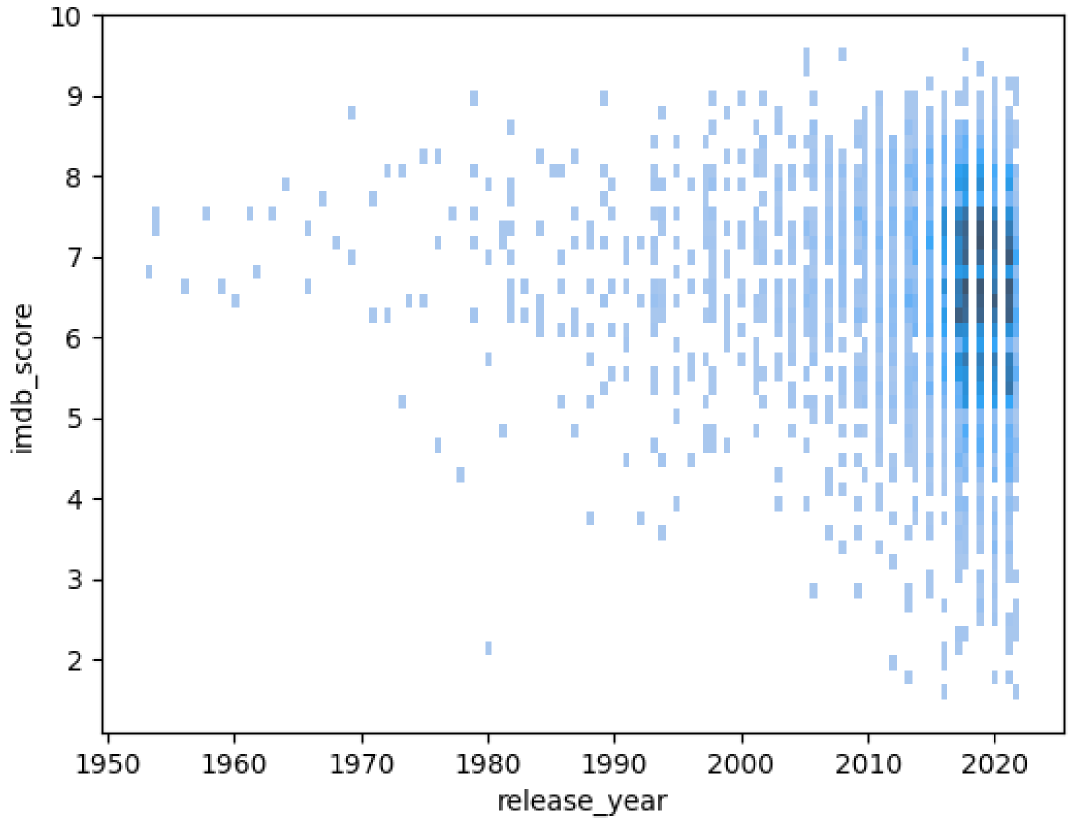


answer2: the correlation between them is 0.19, which is a weak positive corelation

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?

```
sns.histplot(data=data, x='release_year', y='imdb_score')
```

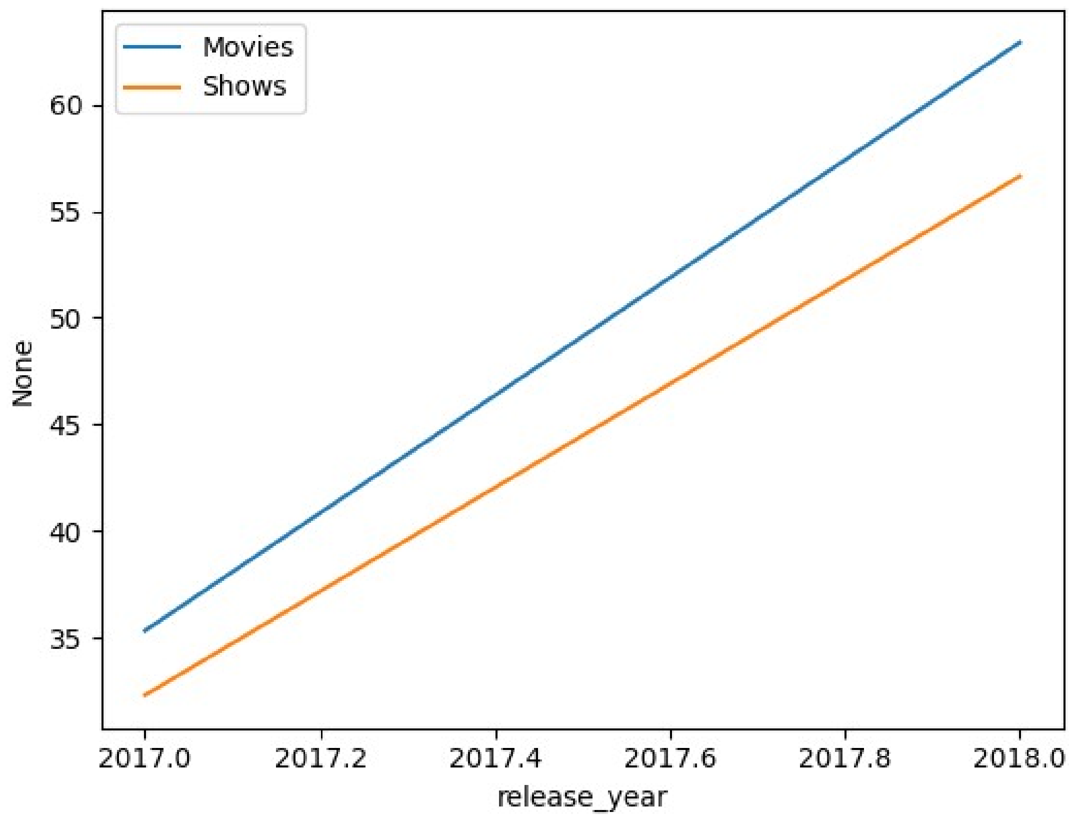
```
<Axes: xlabel='release_year', ylabel='imdb_score'>
```



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.

```
# 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)
```

```
<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

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

import os
for dirname, _, filenames in
os.walk('""C:/Users/ahap0/Downloads/Advertising.csv""'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

df=pd.read_csv("C:/Users/ahap0/Downloads/Advertising.csv")
df

     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]

df.drop('Unnamed: 0',axis=1,inplace=True)
df.shape

(200, 4)
```

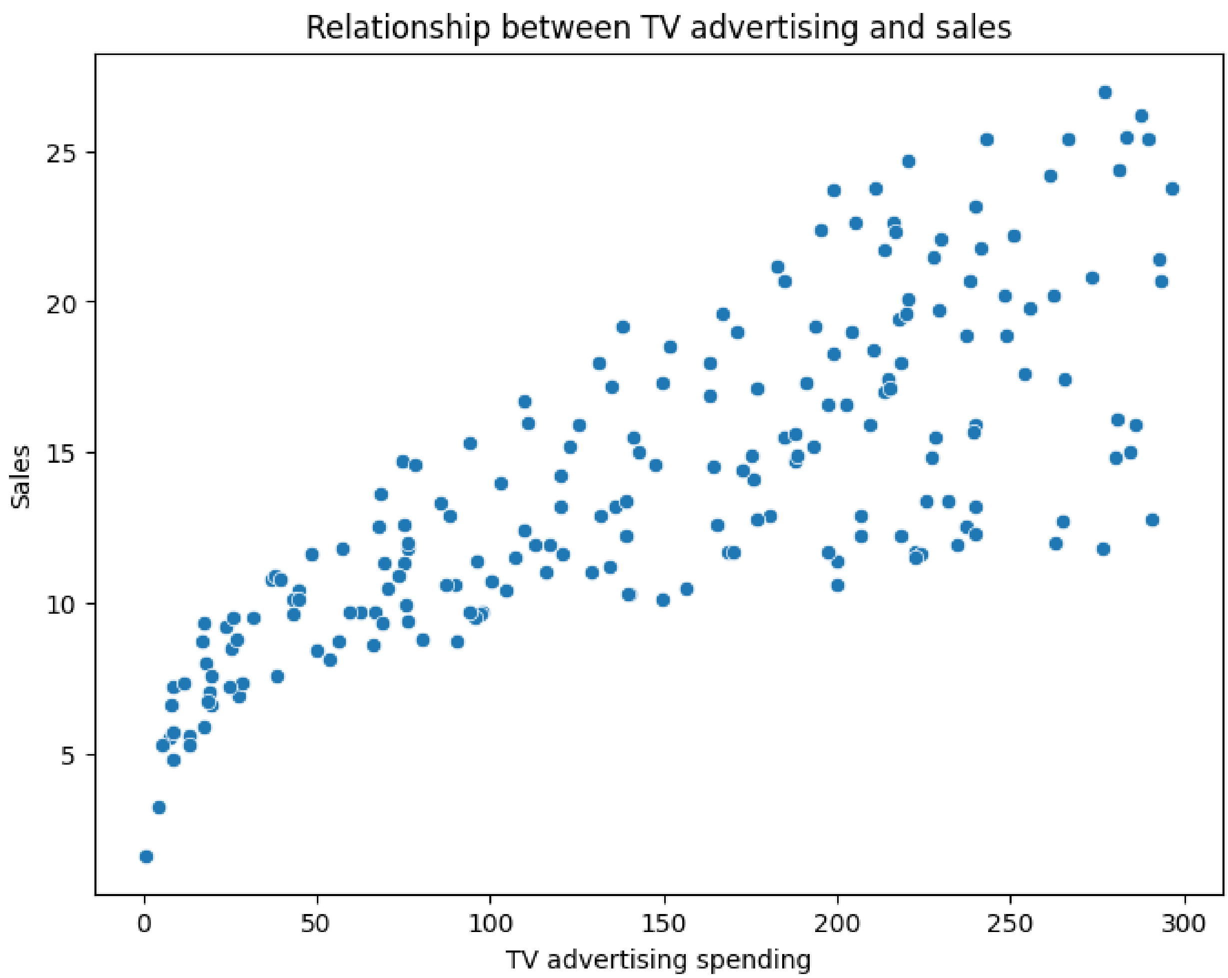
ques1. Calculate the count,mean ,1st quartile,median , 3rd quartile,std,min max?

```
df.describe()

               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

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
```

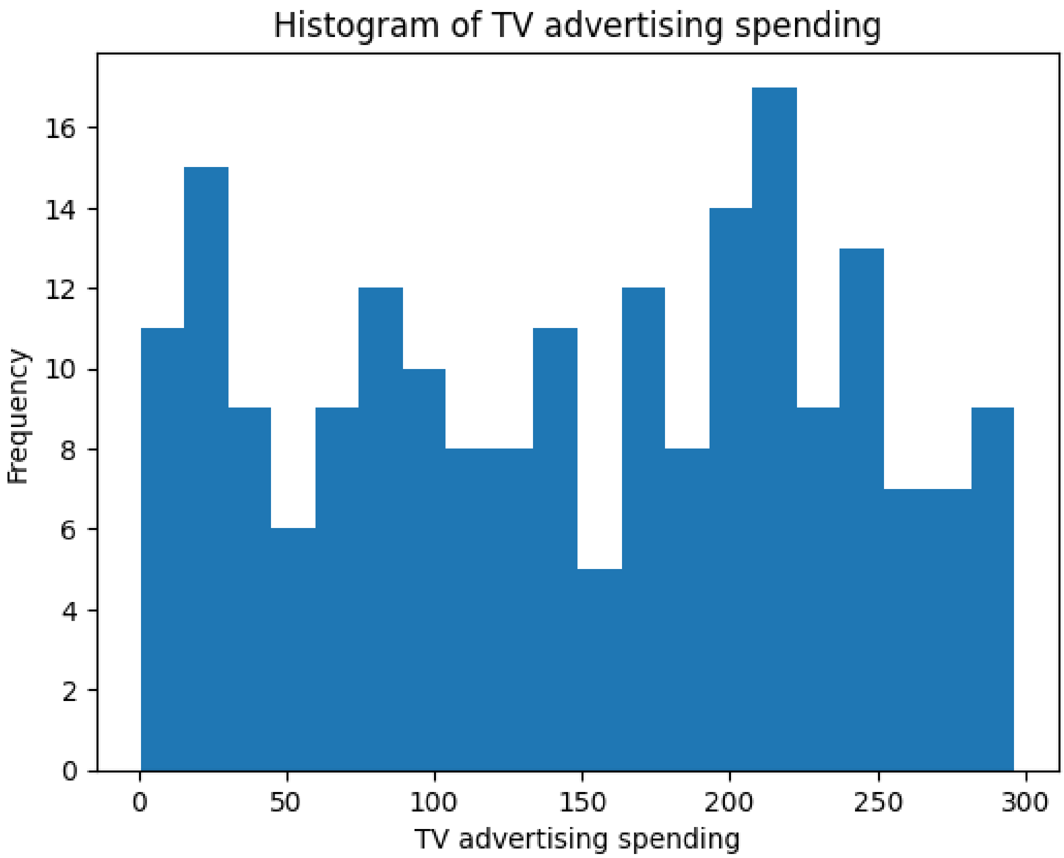
ques 2. plot the relationship between 'TV' advertising spending and 'Sales'?

```
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()
```



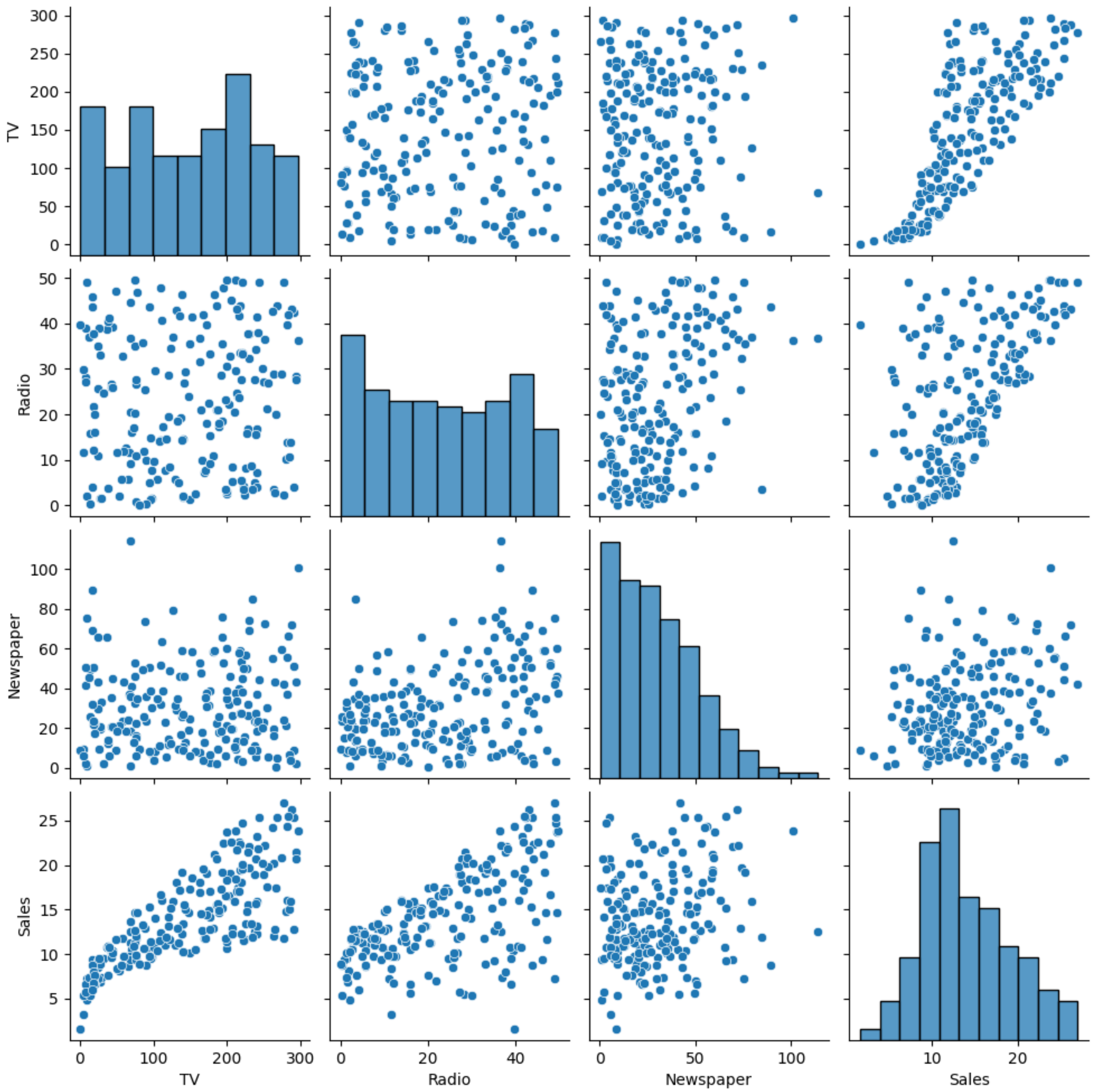
ques 3: Plot the Histogram of 'TV' advertising spending?

```
plt.hist(df['TV'],bins=20)
plt.xlabel('TV advertising spending')
plt.ylabel('Frequency')
plt.title('Histogram of TV advertising spending')
plt.show()
```



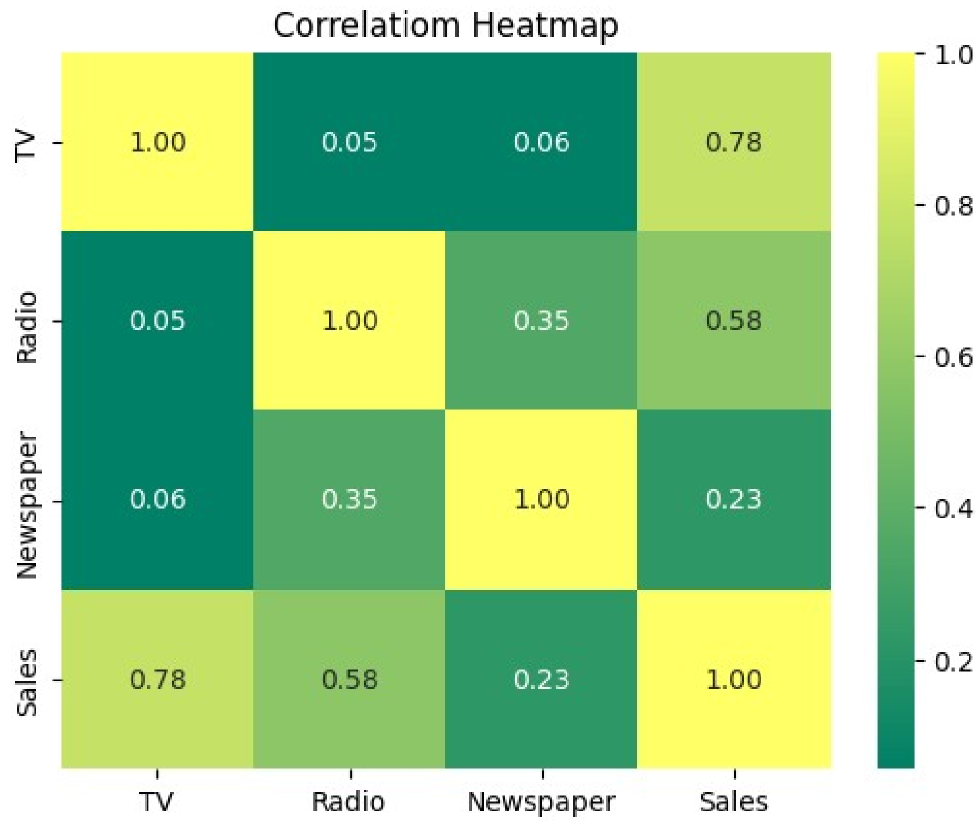
ques 4: PLot both relationship and histogram?

```
sns.pairplot(df)
plt.show()
```



ques5: plot a correlation heatmap?

```
correlation_matrix=df.corr()

sns.heatmap(correlation_matrix,annot=True,cmap='summer',fmt='.2f')
plt.title('Correlatiom Heatmap')
plt.show()
```

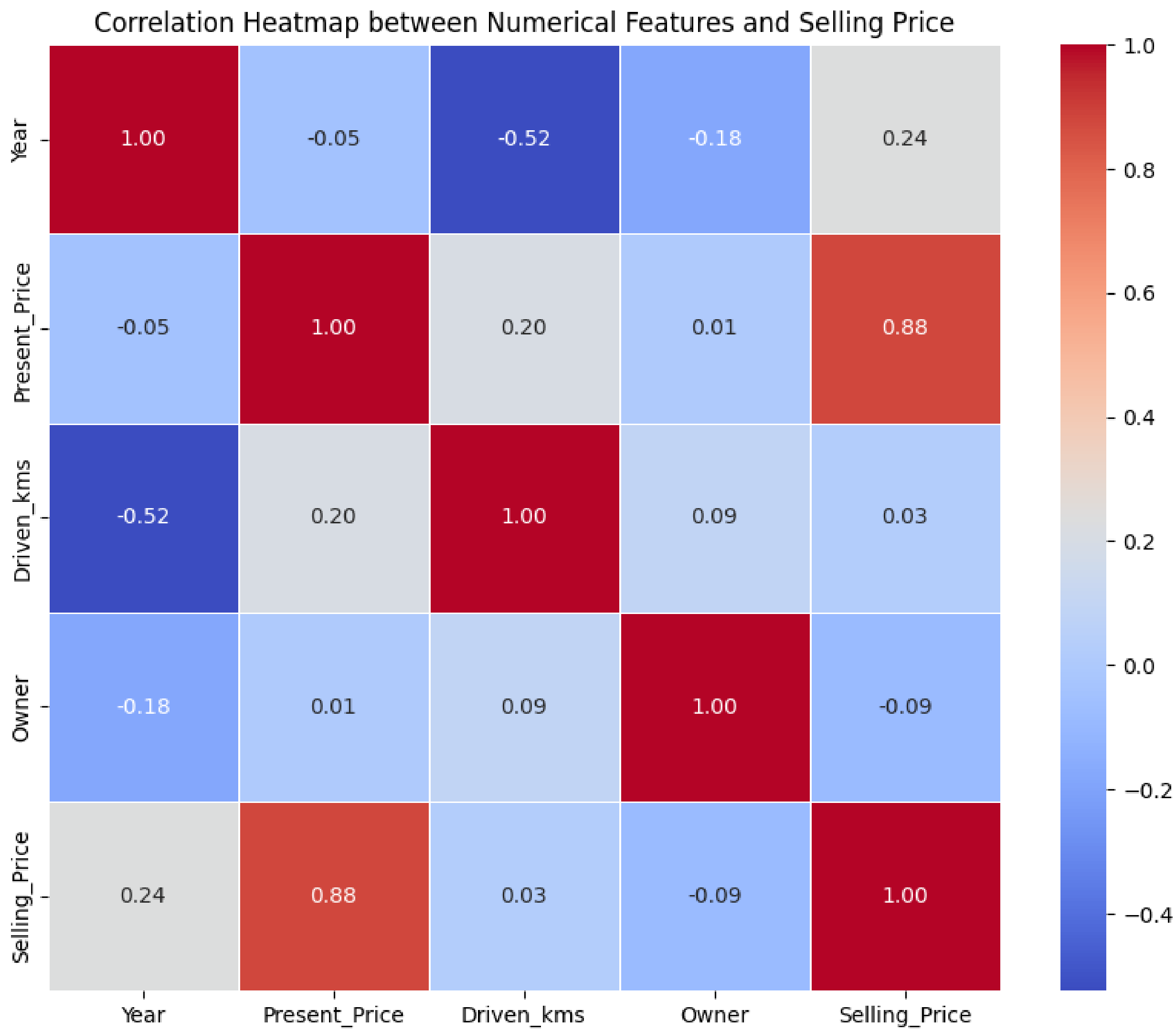

Correlatiom Heatmap
TV
Radio
Newspaper
Sales
1.00
0.05
0.06
0.78
0.05
1.00
0.35
0.58
0.06
0.35
1.00
0.23
0.78
0.58
0.23
1.00
TV
Radio
Newspaper
Sales
1.0
0.8
0.6
0.4
0.2

```
Topic.  Car prediction

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()

  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
```

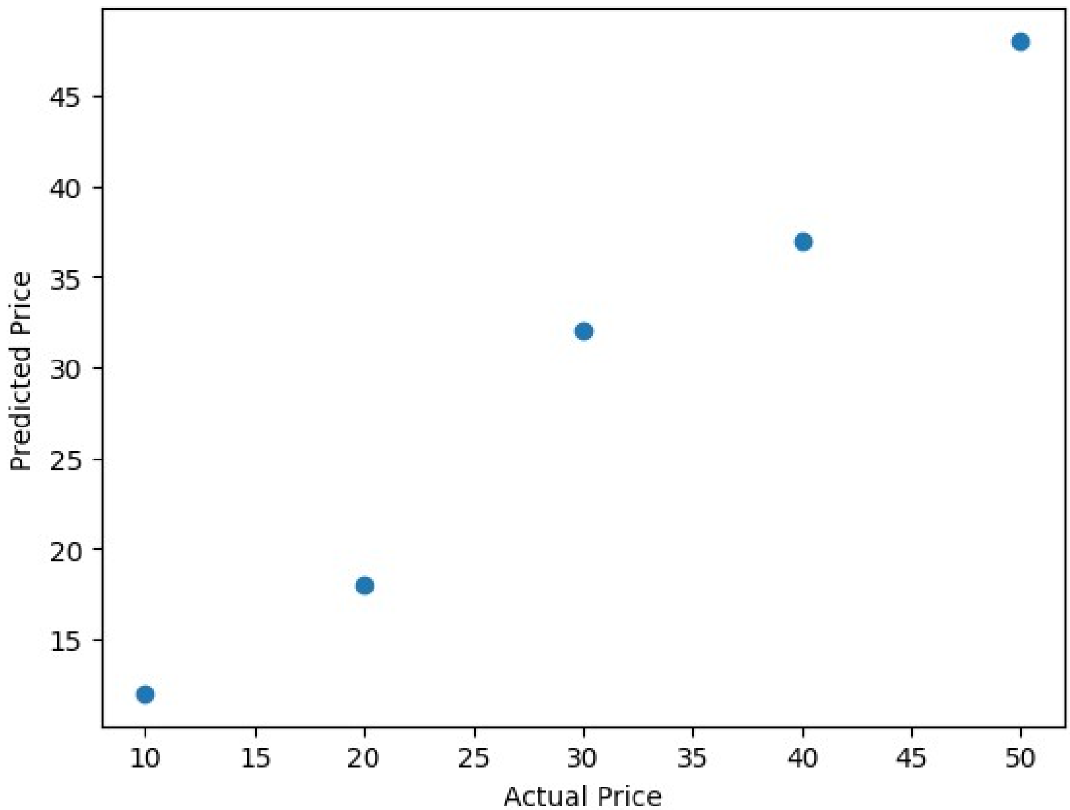
question 1--> "How many types of fuel are there for cars—petrol, diesel, and CNG?"

```
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?

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)
```

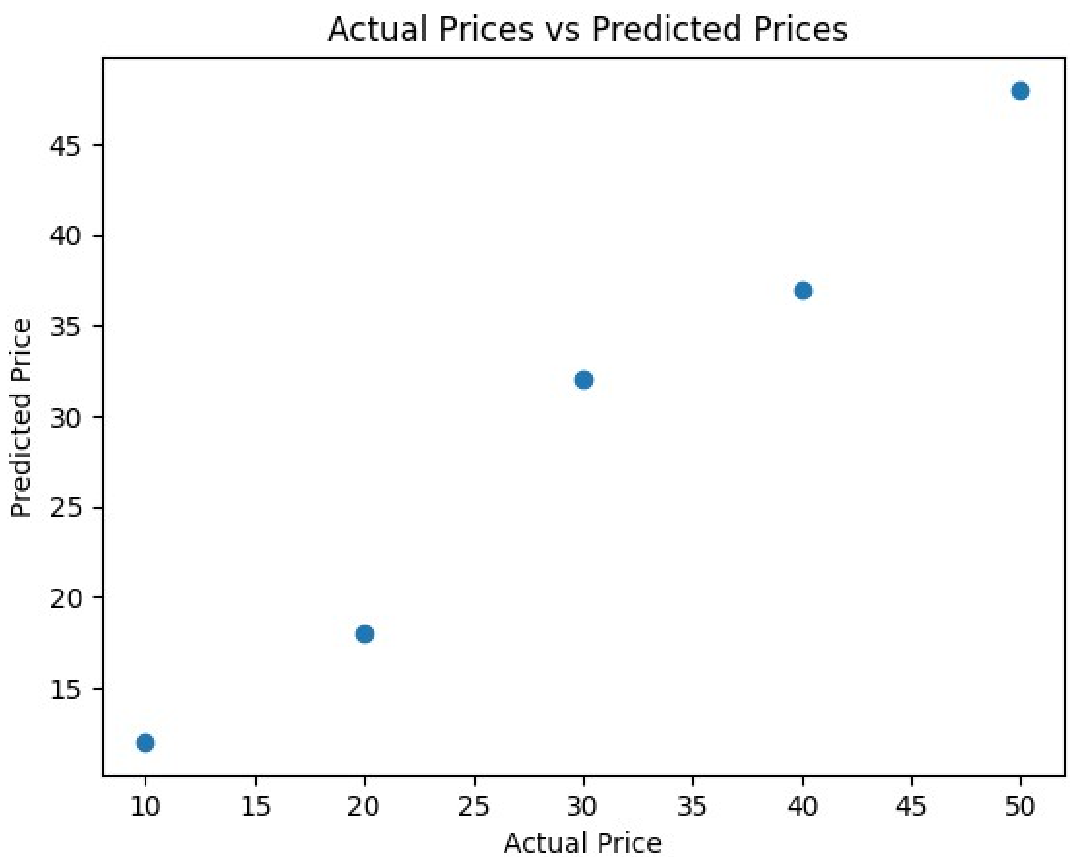



```
Question 3--> How closely do the predicted prices align with the
actual prices?

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()
```




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

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]

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

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
45
40
35
30
25
20
15
10
15
20
25
30
35
40
45
50

```
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

Data Science Lab

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

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

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

Data Loading and Inspection

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

```
file_path = 'healthcare_dataset.csv'
df = pd.read_csv(file_path)
```

Display the first 5 rows of the DataFrame

```
df.head()
```

```
                  Name  Age  Gender Blood Type Medical Condition  \
0      Tiffany Ramirez   81  Female         O-          Diabetes   
1          Ruben Burns   35    Male         O+            Asthma   
2            Chad Byrd   61    Male         B-           Obesity   
3    Antonio Frederick   49    Male         B-            Asthma   
4  Mrs. Brandy Flowers   51    Male         O-         Arthritis   

  Date of Admission          Doctor                    Hospital  \
0        2022-11-17  Patrick Parker            Wallace-Hamilton   
1        2023-06-01   Diane Jackson  Burke, Griffin and Cooper   
2        2019-01-09      Paul Baker                  Walton LLC   
3        2020-05-02  Brian Chandler                  Garcia Ltd   
4        2021-07-09  Dustin Griffin     Jones, Brown and Murray   

  Insurance Provider  Billing Amount  Room Number Admission Type  \
0           Medicare    37490.983364          146       Elective   
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
```

Display the last 5 rows of the DataFrame

```
df.tail()
```

```
                      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
```

Data Inspection and Exploration

Display the basic information about the DataFrame

```
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
```

This will show all column name in an array

```
df.columns.tolist()
```

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

```
df.shape
```

```
(10000, 15)
```

Check for missing values

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

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

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

Data Analysis

Question1: Calculate average billing amount and total amount.

```
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
```

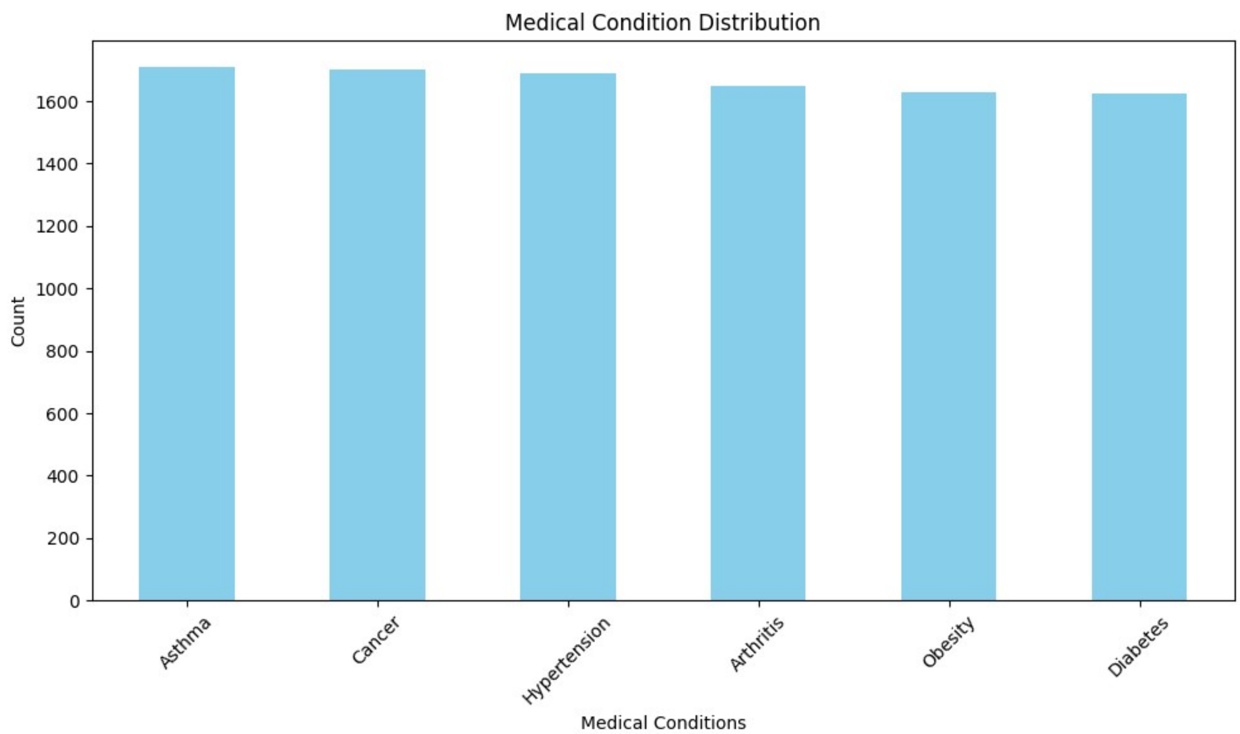
Question2: How to cacuclate medical condition from dataset and plot barchart.

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

condition_count
```

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

Plotted a pie chart of medical condition.

```
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()
```



Data Exploratory

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

```
df.columns

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')
```

Percentage of male and female patient.

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

Female    5075
Male      4925
Name: Gender, dtype: int64

total_count = gender_counts.sum()
total_count
```

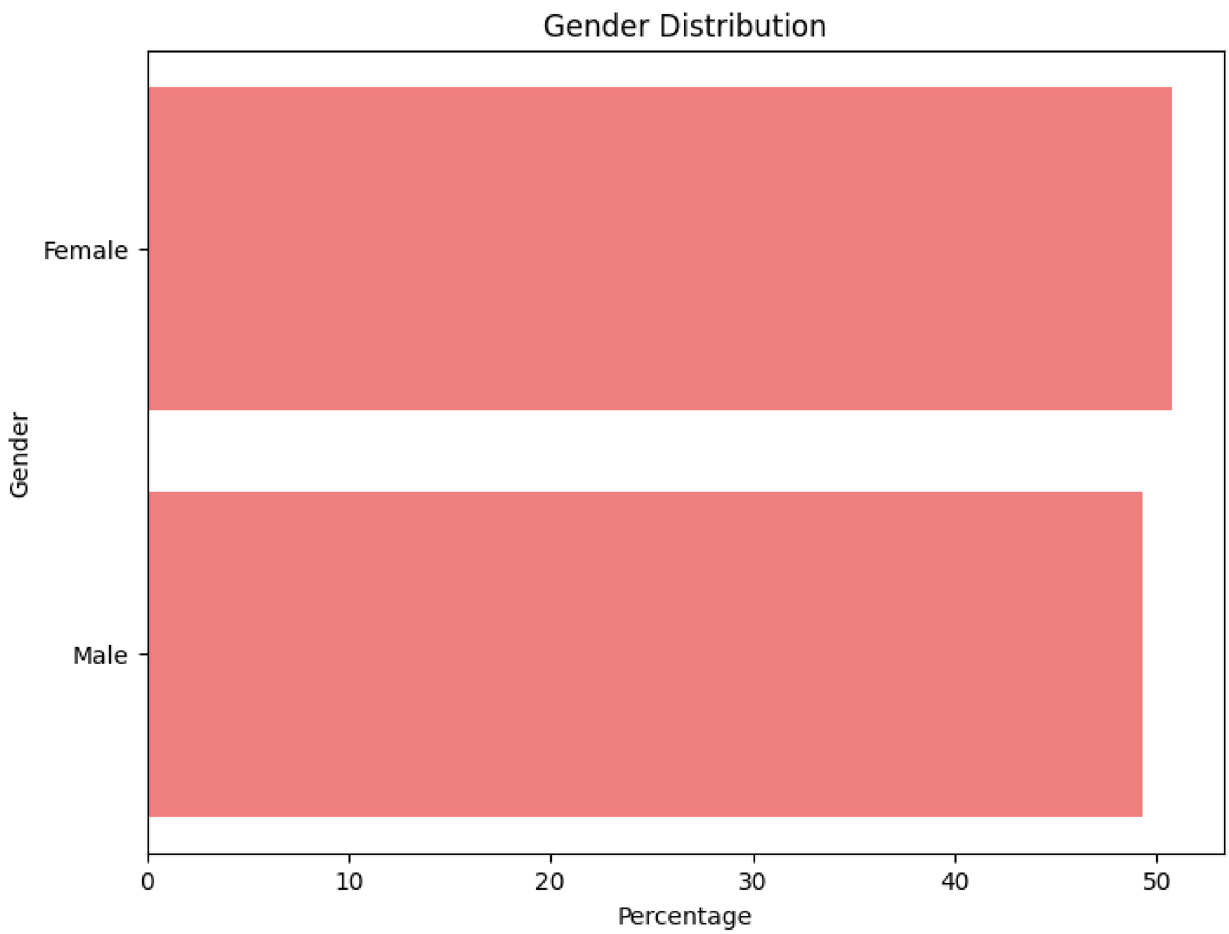
```
10000
```

```
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
```

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

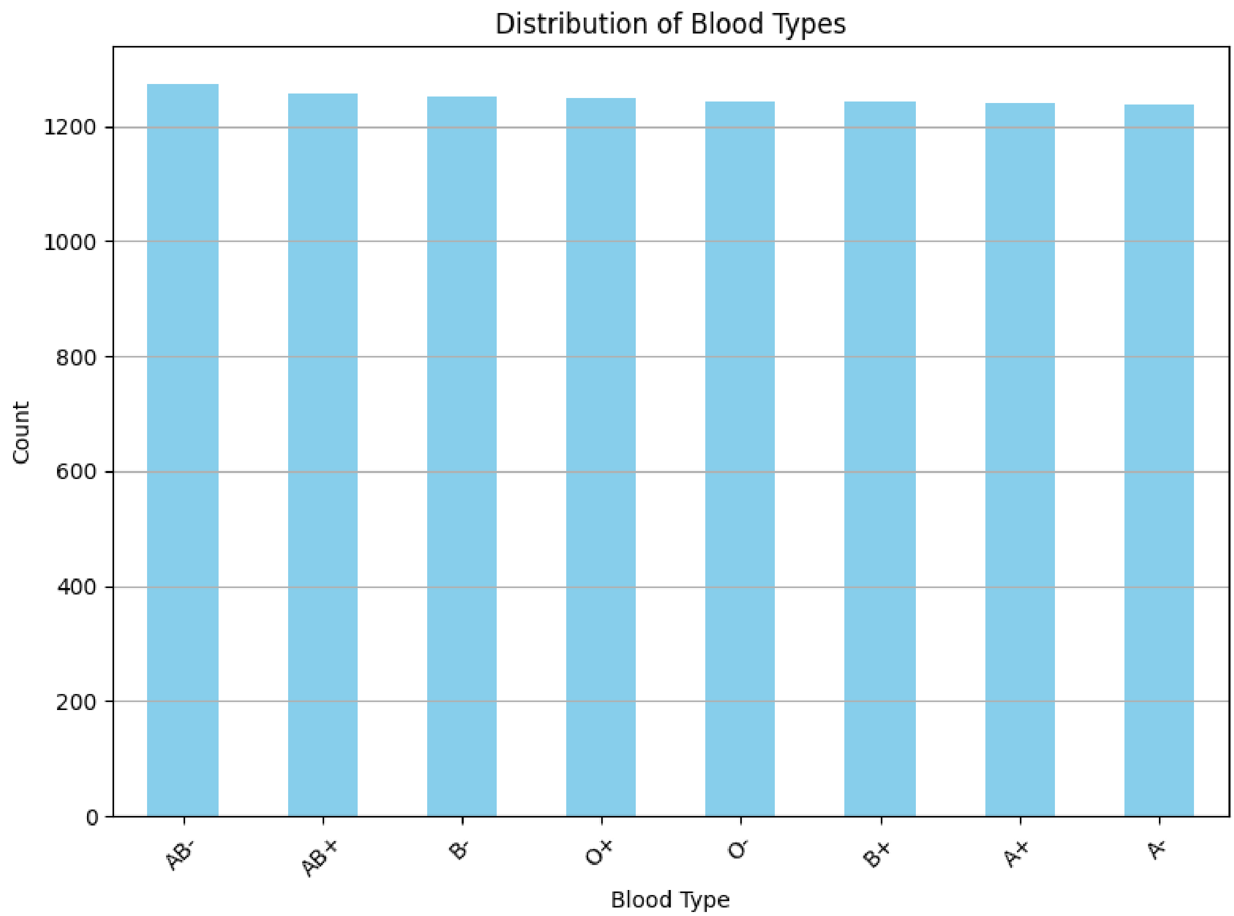
```
values = [percentage_male, percentage_female]
labels = ['Male', 'Female']

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()
```



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

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

Blood Type
AB-     1275
AB+     1258
B-      1252
O+      1248
O-      1244
B+      1244
A+      1241
A-      1238
Name: count, dtype: int64

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()
```



```python
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')

df = pd.read_csv('weatherAUS.csv')

df.head()
```

```
         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]
```

Q1 How many column are numerical and categorical

```
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
```

```
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'])
```

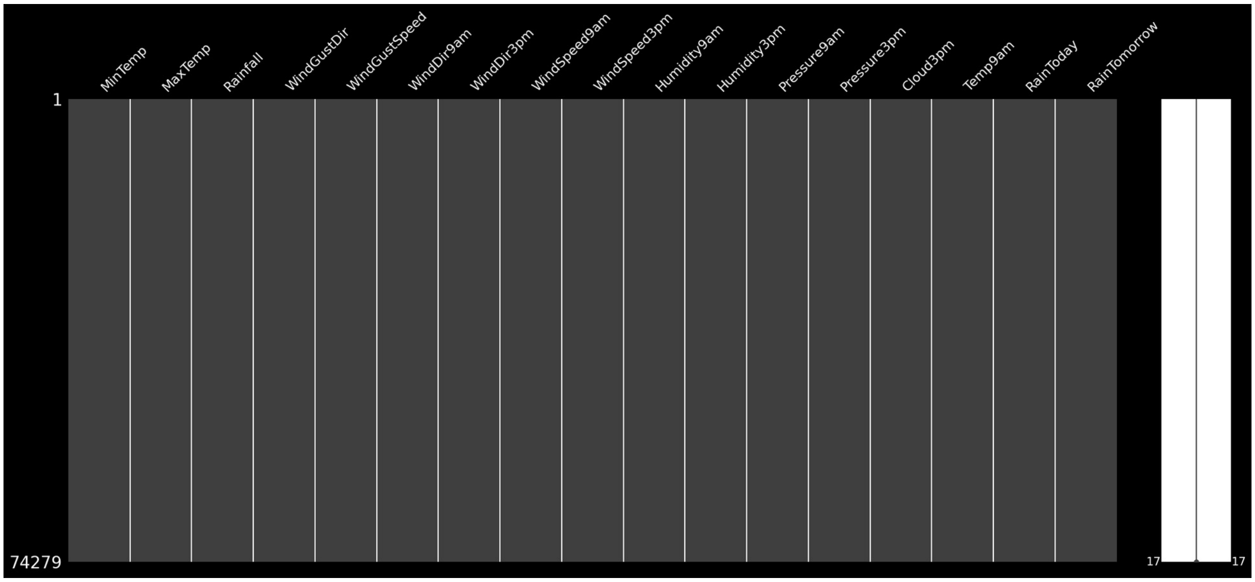
```
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])


df.head()

    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

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

x.head()

    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
```

Q2.vusualize the null values

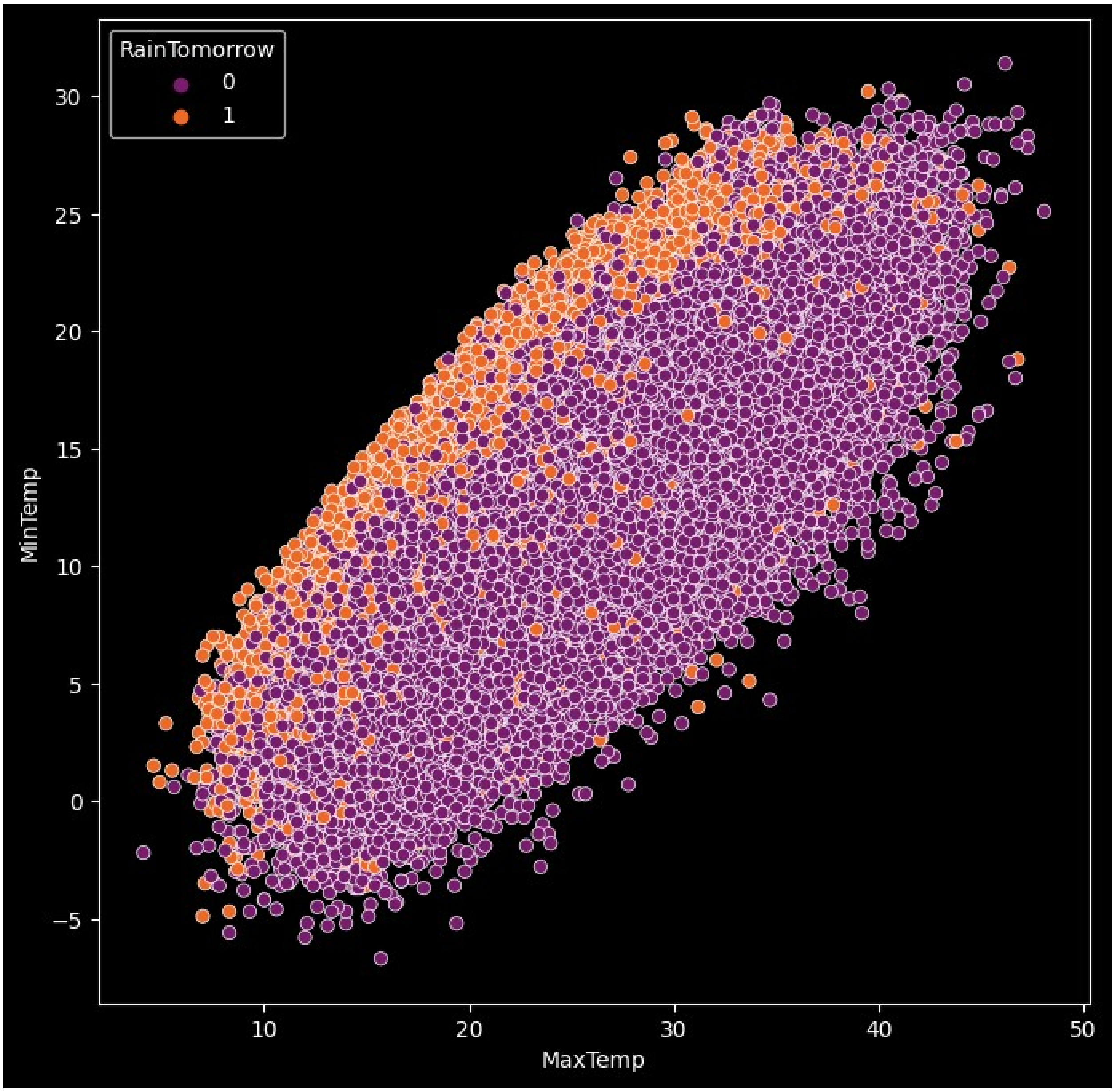
```
msno.matrix(df)
```

```
<Axes: >
```

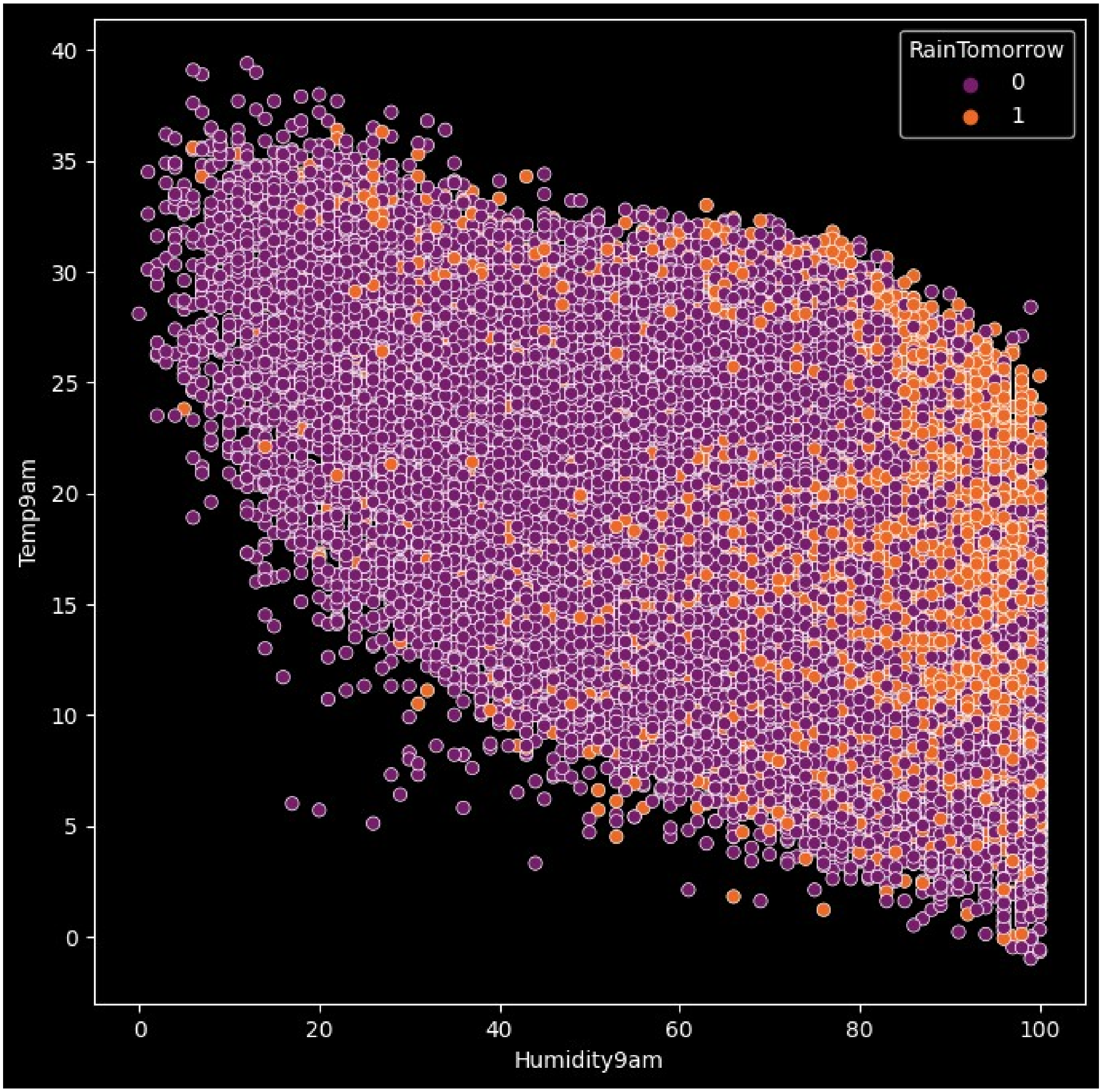



```
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'MaxTemp', y = 'MinTemp', hue = 'RainTomorrow', palette = 'inferno',data = df)
```

```
<Axes: xlabel='MaxTemp', ylabel='MinTemp'>
```



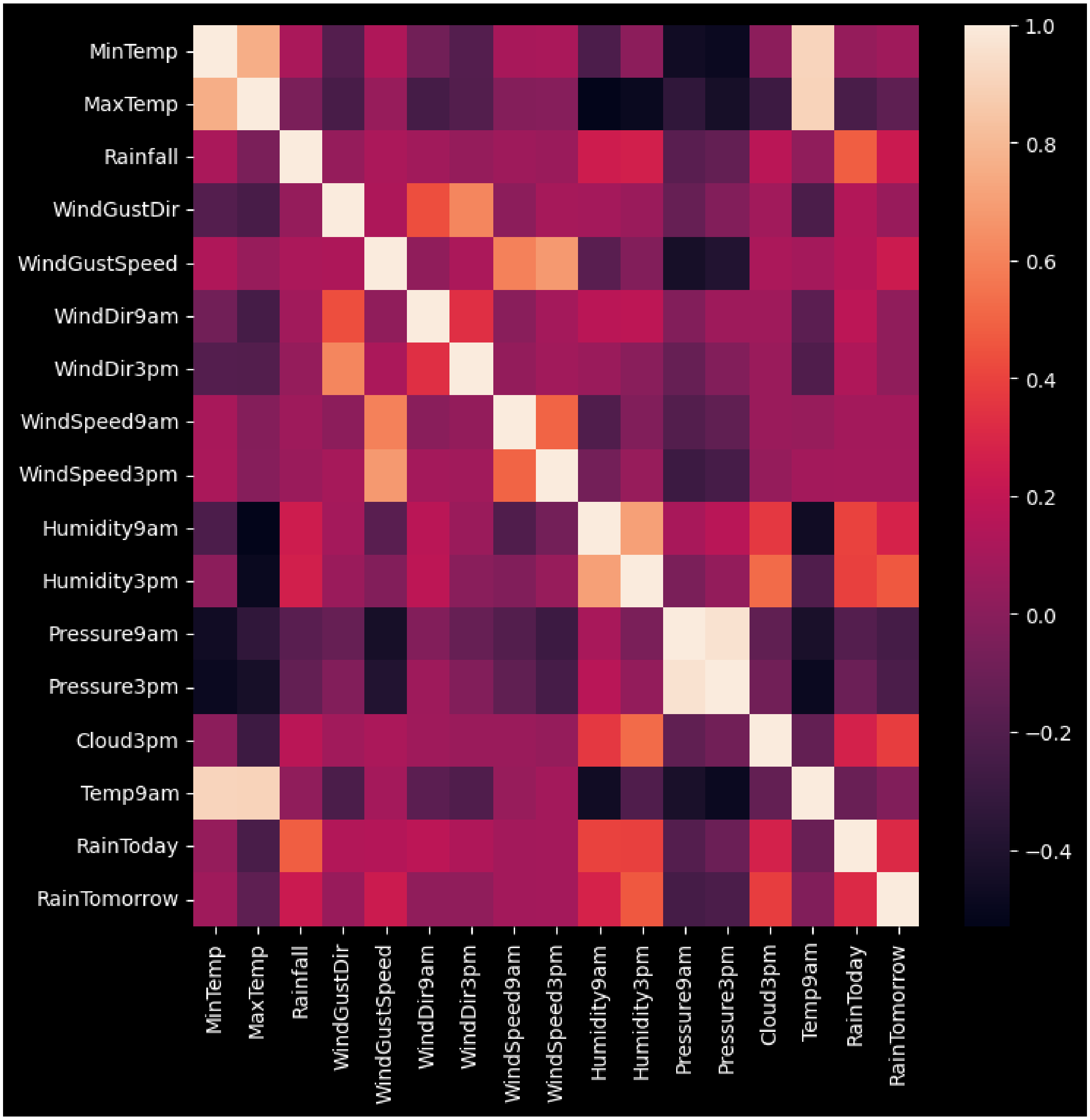

```
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'Humidity9am', y = 'Temp9am', hue =
'RainTomorrow', palette = 'inferno',data = df)

<Axes: xlabel='Humidity9am', ylabel='Temp9am'>
```



Q3 Draw a Heatmap

```
plt.figure(figsize = (8,8))
sns.heatmap(df.corr())
```

```
<Axes: >
```



Q4.How to convert yes or no into numerical

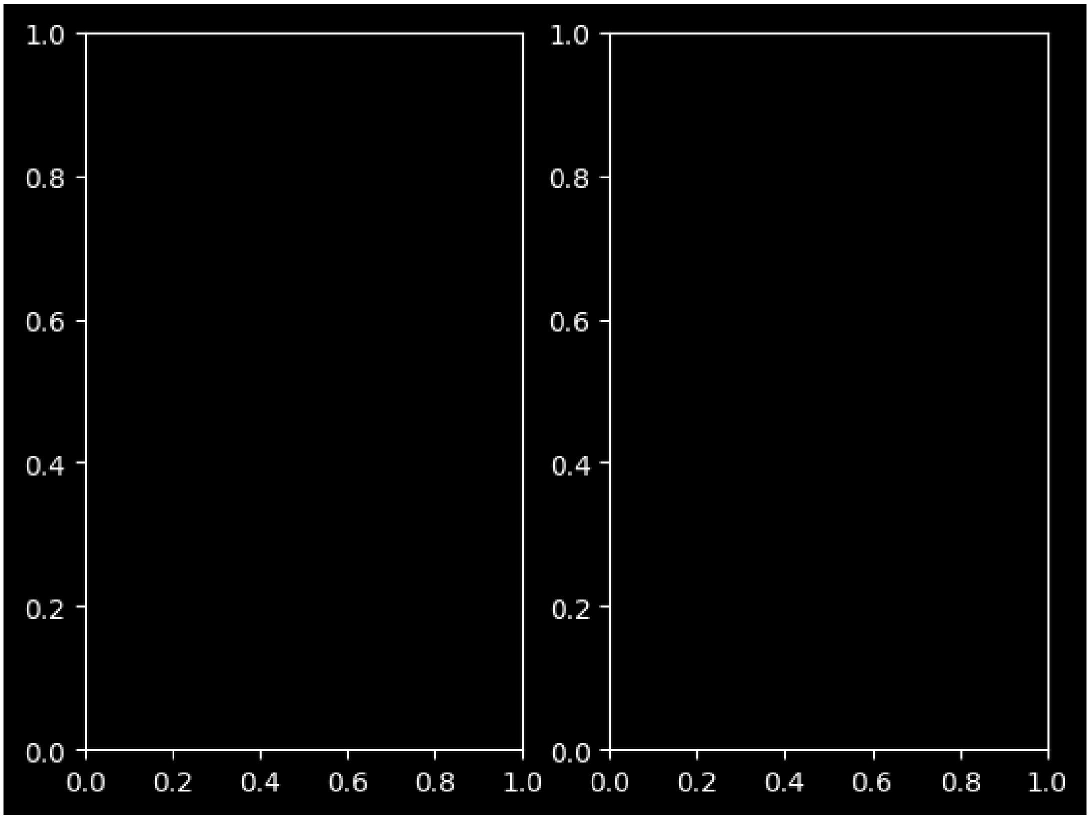
```
df['RainTomorrow'] = df['RainTomorrow'].map({'Yes':1, 'No':0})
df['RainToday'] = df['RainToday'].map({'Yes':1, 'No':0})
print(df.RainToday)
print(df.RainTomorrow)

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

```python
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)

<Axes: >
```




```
&lt;Figure size 2000x2000 with 0 Axes&gt;
```

home-loan-approval

December 12, 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  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             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')
```

```python
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Pre-Processing

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

df
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  
chlorides  \
0               7.4             0.700         0.00             1.9  
0.076
1               7.8             0.880         0.00             2.6  
0.098
2               7.8             0.760         0.04             2.3  
0.092
3              11.2             0.280         0.56             1.9  
0.075
4               7.4             0.700         0.00             1.9  
0.076
...             ...               ...          ...             ...  
...
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  \
0                    11.0                  34.0  0.99780  3.51  
0.56
1                    25.0                  67.0  0.99680  3.20  
0.68
2                    15.0                  54.0  0.99700  3.26  
0.65
3                    17.0                  60.0  0.99800  3.16  
0.58
4                    11.0                  34.0  0.99780  3.51  
0.56
```

```
...                   ...                   ...      ...   ...
...
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
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]
```

```
df.head()
```

```
   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
```

```
df.tail()
```

```
      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
```

```
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
```

```
df.describe()
```

```
       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

```
df.dtypes
```

```
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
```

```
df.size
```

```
14859
```

```
df.shape
```

```
(1143, 13)
```

```
df.columns
```

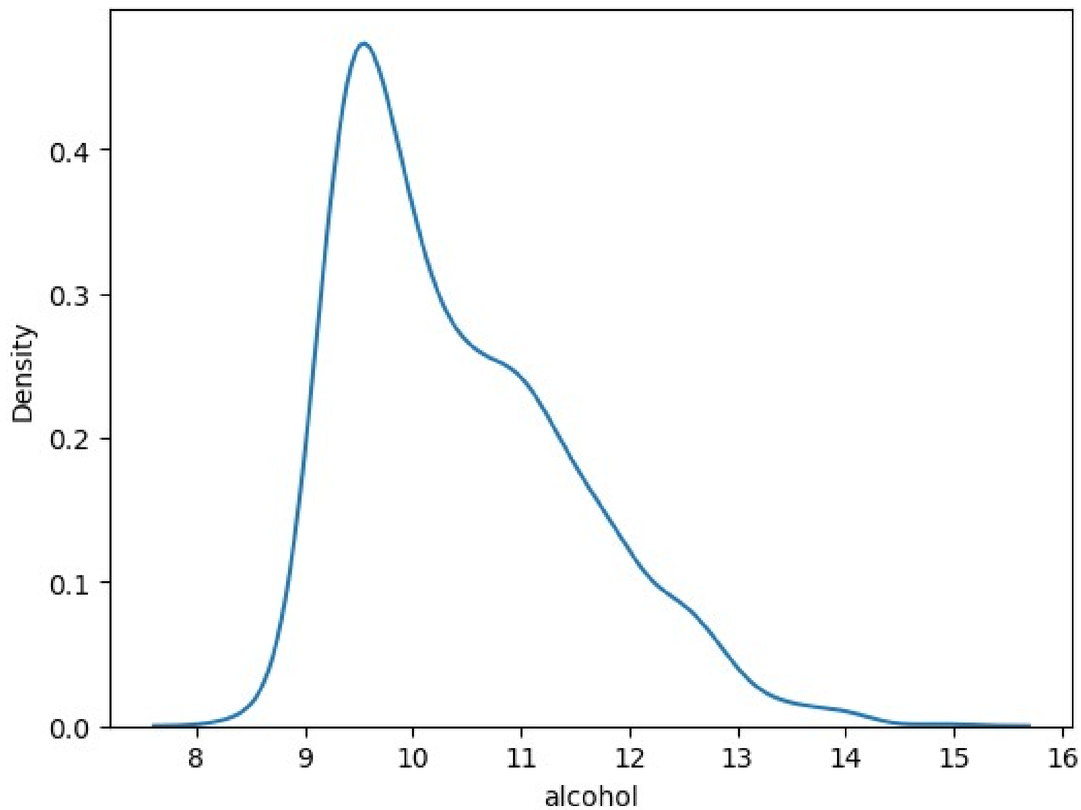
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual
sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',
'density',
```

```
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

How to visualize the single variable 'alcohol' to visualize its distribution?

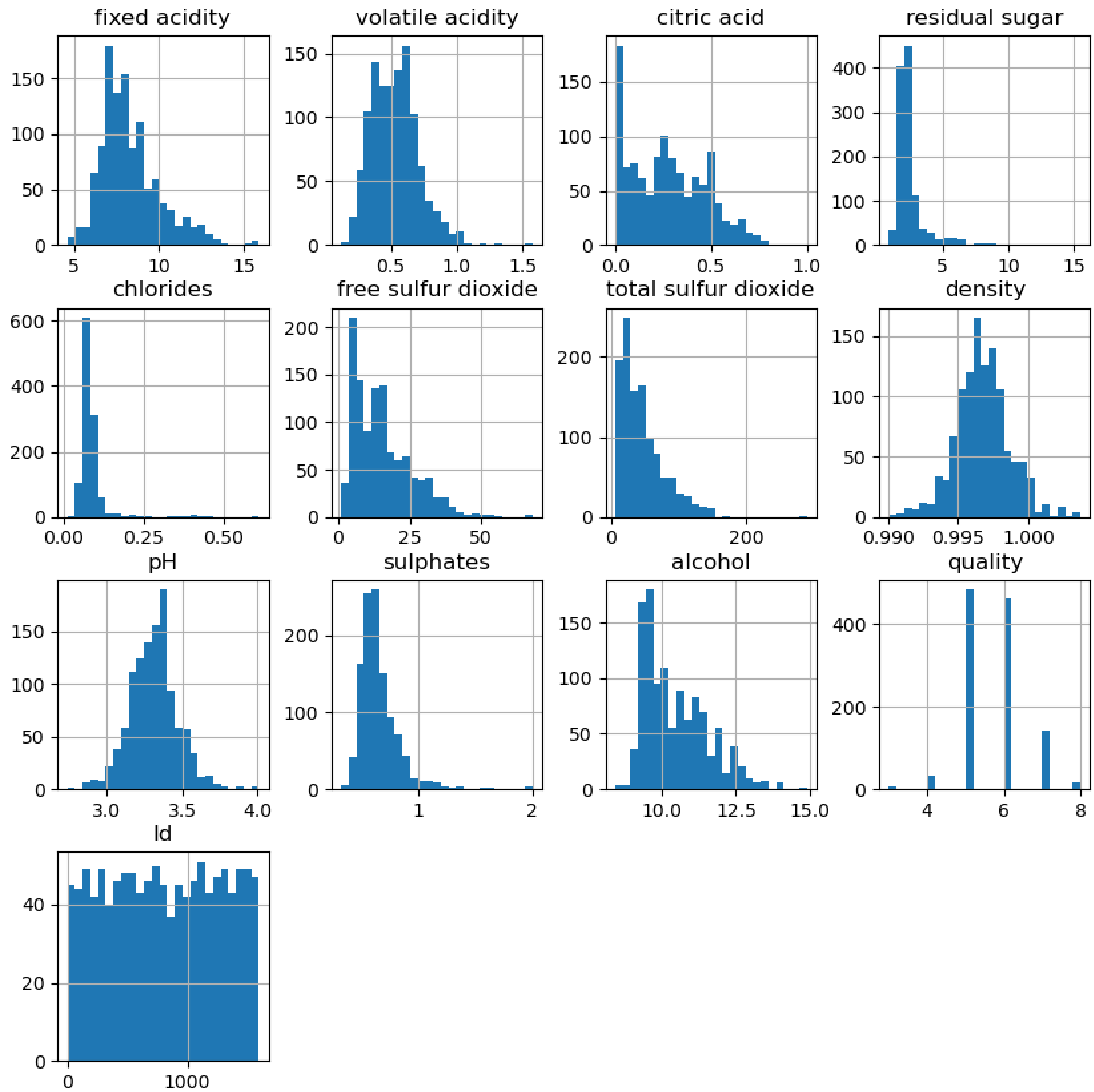
```
sns.kdeplot(df['alcohol'])
```

```
<Axes: xlabel='alcohol', ylabel='Density'>
```



Display histogram for all feature variables.

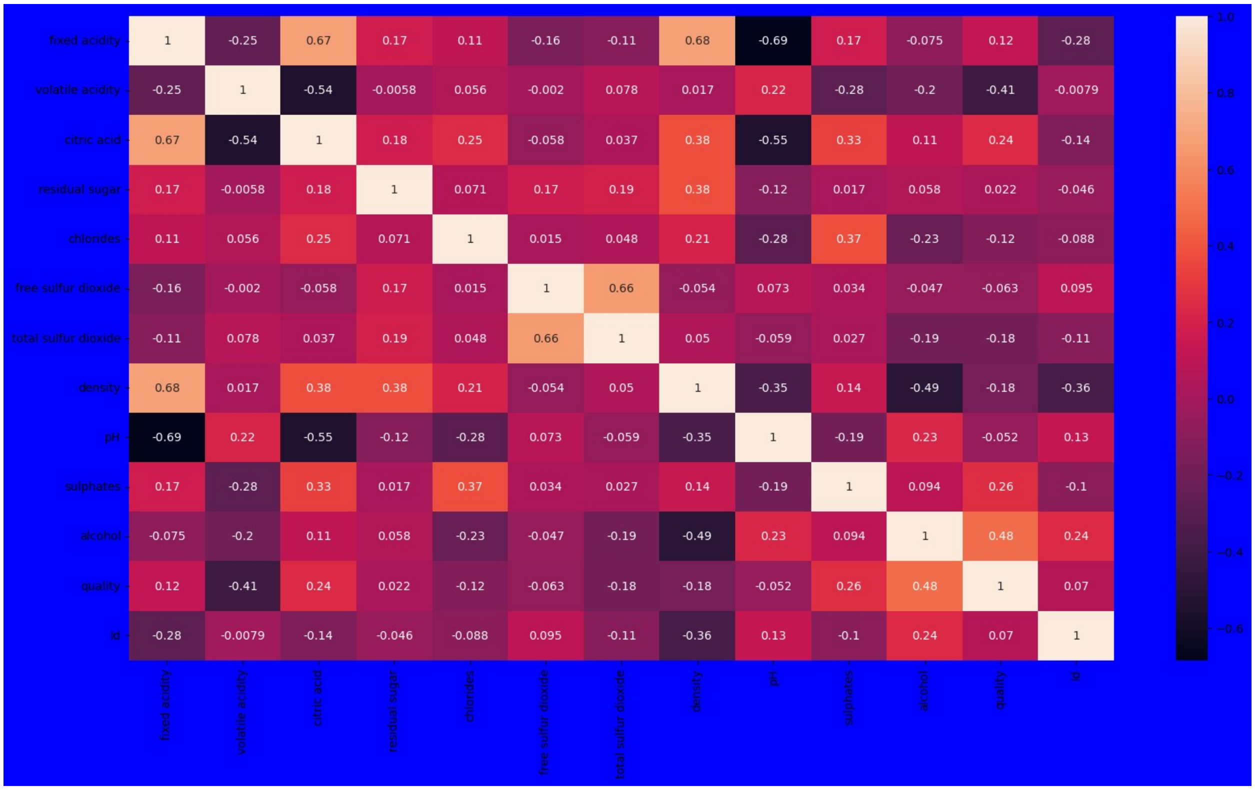
```
df.hist(bins=25,figsize=(10,10))
plt.show()
```



Use a statistical method that finds the bonding and relationship between two features.

```
# ploting heatmap
plt.figure(figsize=[19,10],facecolor='blue')
sns.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```



What is the average value of the 'alcohol' column?

```
average_alcohol = df['alcohol'].mean()
print(f"The average alcohol content is: {average_alcohol}")
```

```
The average alcohol content is: 10.442111402741325
```

What is the correlation between 'fixed acidity' and 'pH'?

```
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
```

How many records have 'density' greater than 0.998?

```
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.
```

Project Title - Cardiac Patients Medical Report Analysis

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')
```

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

- 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
```

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()

# 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()

# 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=No
ne)

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_f
rame(name=None)
```

```
df4 = df4.transpose()

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

# 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

# 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();
```

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()
```

```python
# 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()
```

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();
```

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()

# 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();
```

1. Data Preprocessing

```
import pandas as pd
import numpy as np

nf=pd.read_csv('/kaggle/input/netflix/NetFlix.csv')
nf
```

```
     show_id     type                                             title  \
0         s1  TV Show                                                3%
1        s10    Movie                                              1920
2       s100    Movie                                       3 Heroines
3      s1000    Movie  Blue Mountain State: The Rise of Thadland
4      s1001  TV Show                                    Blue Planet II
...      ...      ...                                               ...
7782    s995  TV Show                                        Blown Away
7783    s996  TV Show                                     Blue Exorcist
7784    s997    Movie                       Blue Is the Warmest Color
7785    s998    Movie                                      Blue Jasmine
7786    s999    Movie                                          Blue Jay

                 director
cast  \
0                     NaN  João Miguel, Bianca Comparato, Michel
Gomes, R...
1            Vikram Bhatt  Rajneesh Duggal, Adah Sharma, Indraneil
Sengup...
2          Iman Brotoseno  Reza Rahadian, Bunga Citra Lestari, Tara
Basro...
3            Lev L. Spiro  Alan Ritchson, Darin Brooks, James Cade,
Rob R...
4                     NaN                                    David
Attenborough
...                   ...
...
7782                  NaN
NaN
7783                  NaN  Nobuhiko Okamoto, Jun Fukuyama, Kana
Hanazawa,...
7784  Abdellatif Kechiche  Léa Seydoux, Adèle Exarchopoulos, Salim
Kechio...
7785          Woody Allen  Cate Blanchett, Sally Hawkins, Alec
Baldwin, L...
7786         Alex Lehmann           Sarah Paulson, Mark Duplass, Clu
Gulager

                     country date_added  release_year rating  duration
\
0                     Brazil  14-Aug-20          2020  TV-MA         4
```

```
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

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]

nf.head()
```

```
  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...
```

```
nf.tail()
```

```
     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...
```

```
nf.shape
```

```
(7787, 12)
```

```
nf.size
```

```
93444
```

```
nf.columns
```

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country',
'date_added',
       'release_year', 'rating', 'duration', 'genres', 'description'],
      dtype='object')
```

```
nf.dtypes
```

```
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
```

```
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

```
nf.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
7782    False
7783    False
7784    False
7785    False
7786    False
Length: 7787, dtype: bool
```

```
nf[nf.duplicated()]
```

```
Empty DataFrame
Columns: [show_id, type, title, director, cast, country, date_added,
release_year, rating, duration, genres, description]
Index: []
```

```
#nf.drop_duplicates(inplace=True)
```

```
nf.isnull()
```

```
      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]
```

```
nf.isnull().sum()
```

```
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
```

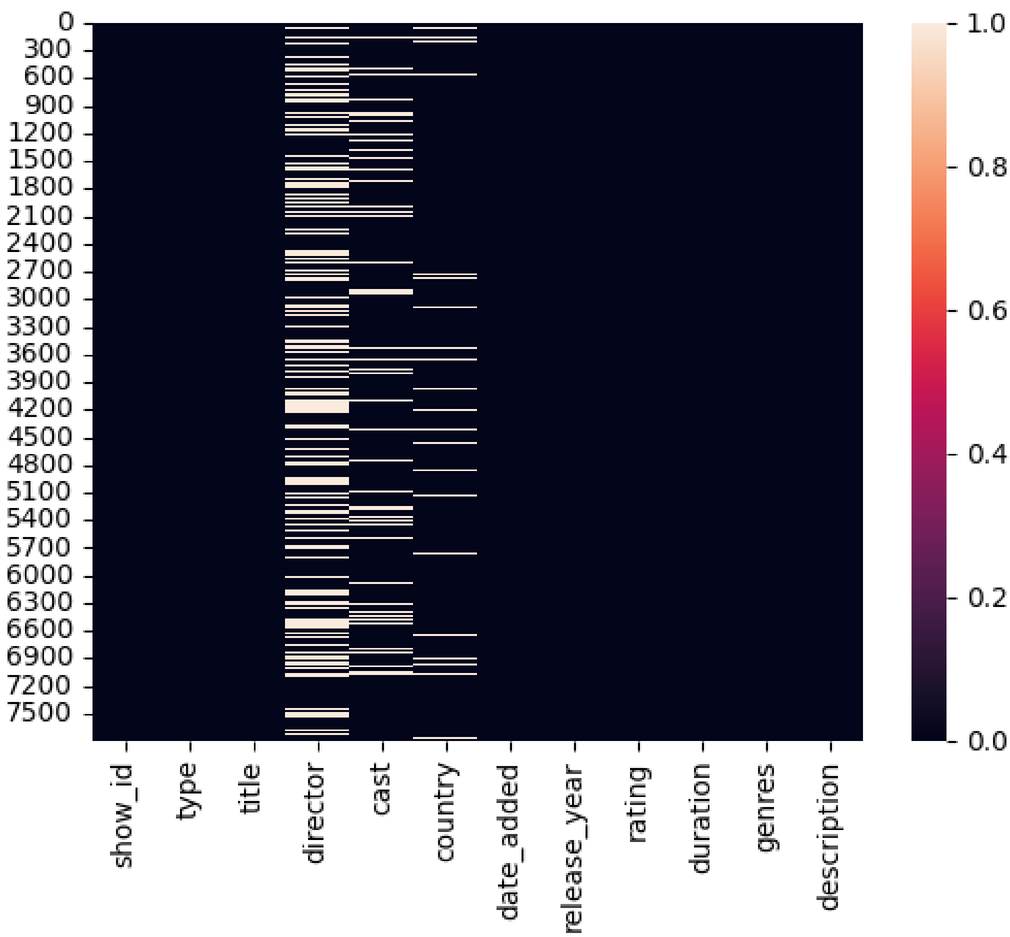
```
nf[nf.director.isnull()]
```

```
     show_id     type                                title director  \
0         s1  TV Show                                   3%      NaN
4      s1001  TV Show                       Blue Planet II      NaN
10     s1007  TV Show                                  BNA      NaN
15     s1011  TV Show                        Bo on the Go!      NaN
17     s1013  TV Show  Bob Ross: Beauty Is Everywhere      NaN
...      ...      ...                                  ...      ...
7777    s990  TV Show                           Blood Pact      NaN
7779    s992  TV Show                            Bloodline      NaN
7780    s993  TV Show                            Bloodride      NaN
7782    s995  TV Show                           Blown Away      NaN
7783    s996  TV Show                        Blue Exorcist      NaN

                                                   cast
country  \
0     João Miguel, Bianca Comparato, Michel Gomes, R...
Brazil
4                                    David Attenborough  United
Kingdom
10    Sumire Morohoshi, Yoshimasa Hosoya, Maria Naga...
Japan
15      Catherine O'Connor, Andrew Sabiston, Jim Fowler
Canada
17                                             Bob Ross
NaN
...                                                 ...               ..
.
7777  Guilherme Fontes, Ravel Cabral, Jonathan Haage...
Brazil
7779  Kyle Chandler, Ben Mendelsohn, Sissy Spacek, L...   United
States
7780  Ine Marie Wilmann, Bjørnar Teigen, Emma Spetal...
```

```
Norway
7782                                                NaN
Canada
7783  Nobuhiko Okamoto, Jun Fukuyama, Kana Hanazawa,...
Japan

     date_added  release_year rating  duration  \
0     14-Aug-20          2020  TV-MA         4
4      3-Dec-18          2017   TV-G         1
10    30-Jun-20          2020  TV-14         1
15    21-Mar-19          2007   TV-Y         1
17     1-Jun-16          1991   TV-G         1
...         ...           ...    ...       ...
7777  10-Oct-18          2018  TV-MA         1
7779  26-May-17          2017  TV-MA         3
7780  13-Mar-20          2020  TV-MA         1
7782  12-Jul-19          2019  TV-14         1
7783   1-Sep-20          2017  TV-MA         2

                                                 genres  \
0     International TV Shows, TV Dramas, TV Sci-Fi &...
4     British TV Shows, Docuseries, Science & Nature TV
10                Anime Series, International TV Shows
15                                             Kids' TV
17                                             TV Shows
...                                                 ...
7777  Crime TV Shows, International TV Shows, TV Dramas
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)

```
import seaborn as sns

sns.heatmap(nf.isnull())       #To show null values count

<Axes: >
```



Q1. For 'House of Cards', What is the Show Id and Who is the Director of this Show?

```
nf[nf['title'].isin(['House of Cards'])]

     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 ...

nf[nf['title'].str.contains('House of Cards')]

     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?

```
nf.dtypes

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

#nf['Date_N']=pd.to_datetime(nf['release_year'])

nf['Date_N'] = pd.to_datetime(nf['release_year'], format='%Y')

nf['Year'] = nf['Date_N'].dt.year

nf.head()

  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

nf.dtypes
```

```
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
```

```python
nf.drop('Date_N', axis=1, inplace=True)
```

```python
nf.dtypes
```

```
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
```

```python
nf['Year'] = pd.to_datetime(nf['Year'], format='%Y')
```

```python
nf['Year'].info()
```

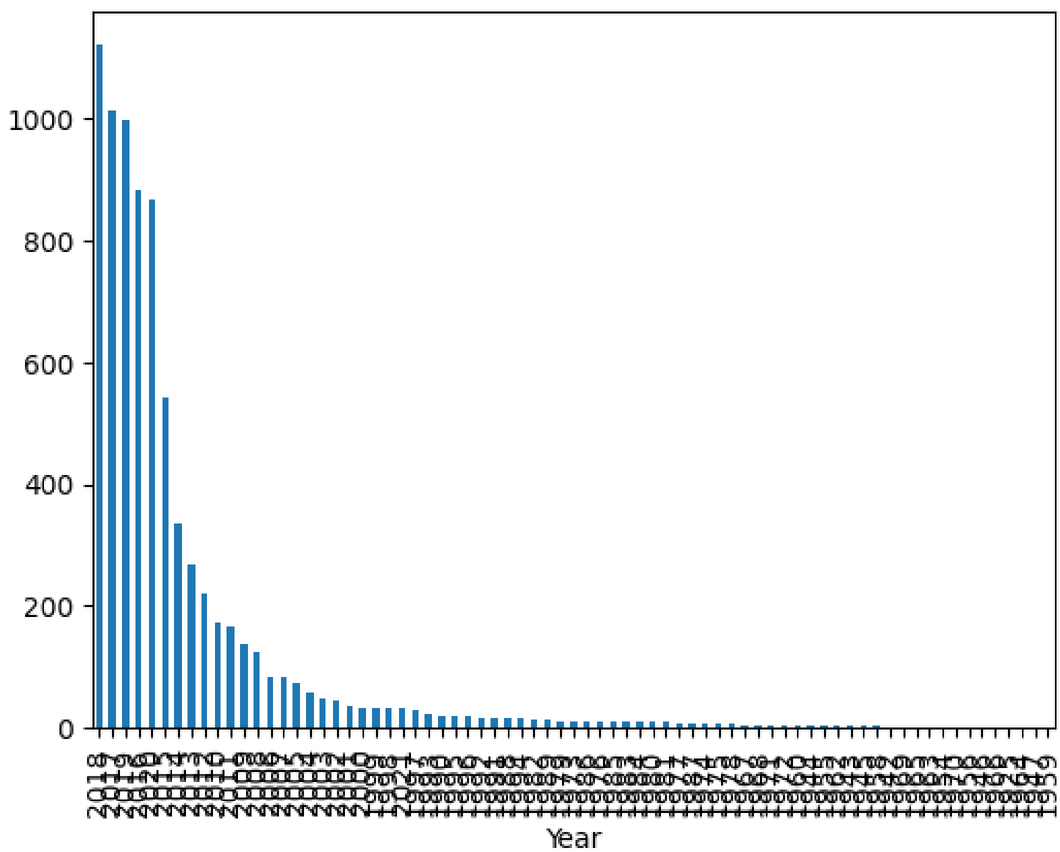
```
<class 'pandas.core.series.Series'>
RangeIndex: 7787 entries, 0 to 7786
Series name: Year
Non-Null Count  Dtype
--------------  -----
7787 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 61.0 KB
```

```python
nf['Year'].dt.year.value_counts()
```

```
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
```

```python
nf['Year'].dt.year.value_counts().plot(kind='bar')
```

```
<Axes: xlabel='Year'>
```



Q3. How many Movies and TV Shows are there in the dataset?

```python
nf.head(2)
```

```
  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
```

```
nf.groupby('type').type.count()
```

```
type
Movie      5377
TV Show    2410
Name: type, dtype: int64
```

```
nf.head(2)
```


```
  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
```

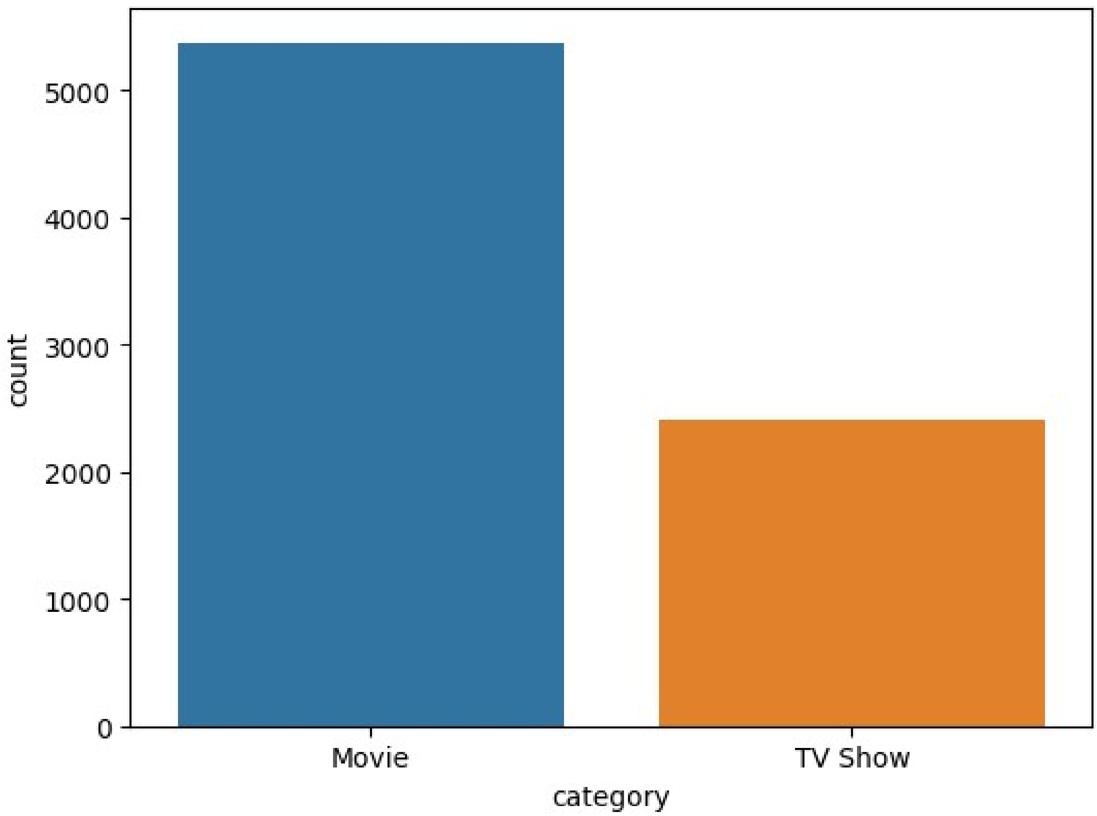
```python
nf.rename(columns={'type': 'category'}, inplace=True)
```

```python
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
```

```python
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[39], 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.

```
nf.head(2)

nf[(nf['category']=='Movie') & (nf['Year']==2000)]
```

```
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.

```
nf[(nf['category']=='TV Show') & (nf['country']=='India')]['title']
```

```
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
```

About the dataset

#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
```

```python
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
```

```python
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
```

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
```

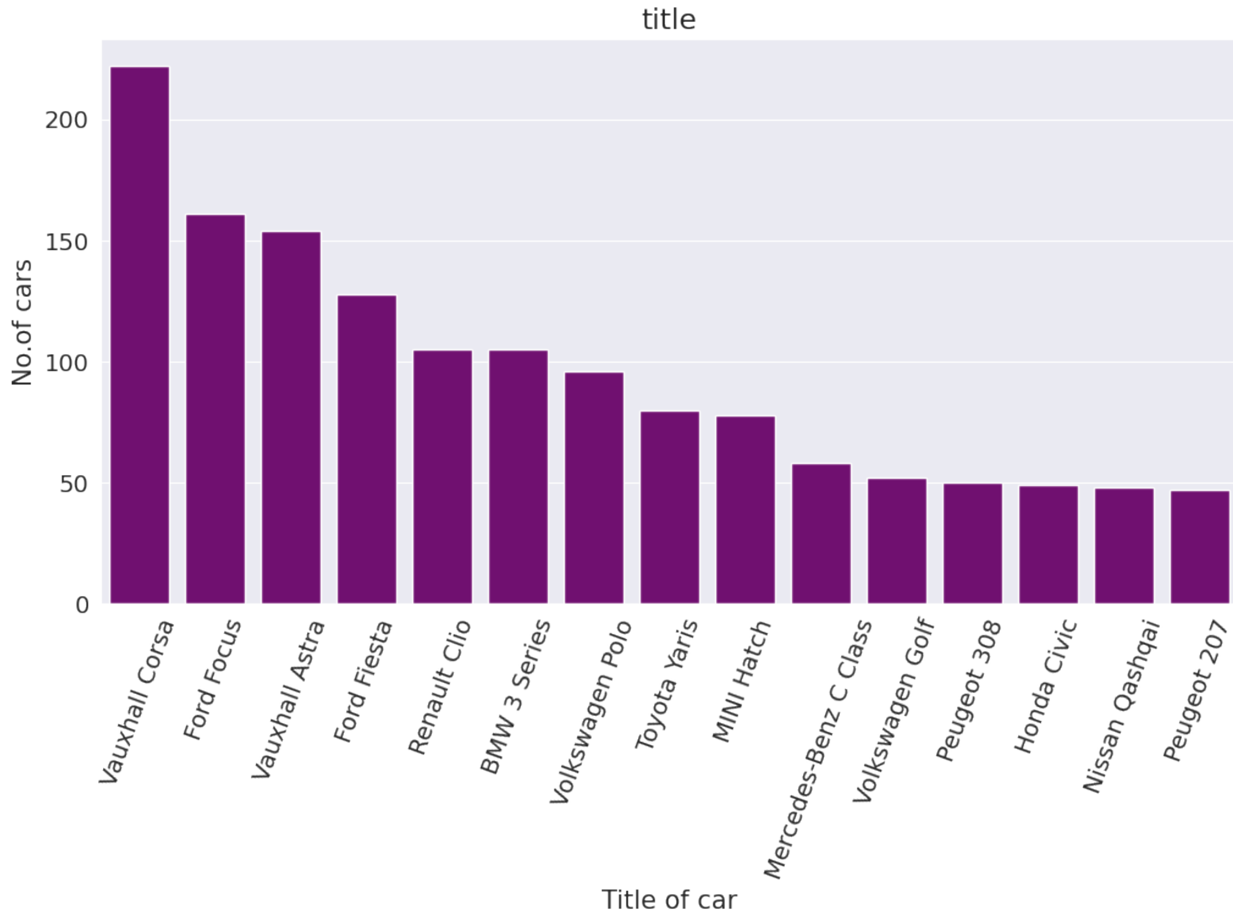
Exploratory data analysis and visualisation

list the **Top 15 cars sold** & show the graph .

```
top_cars=df['title'].value_counts().head(15)
top_cars
```

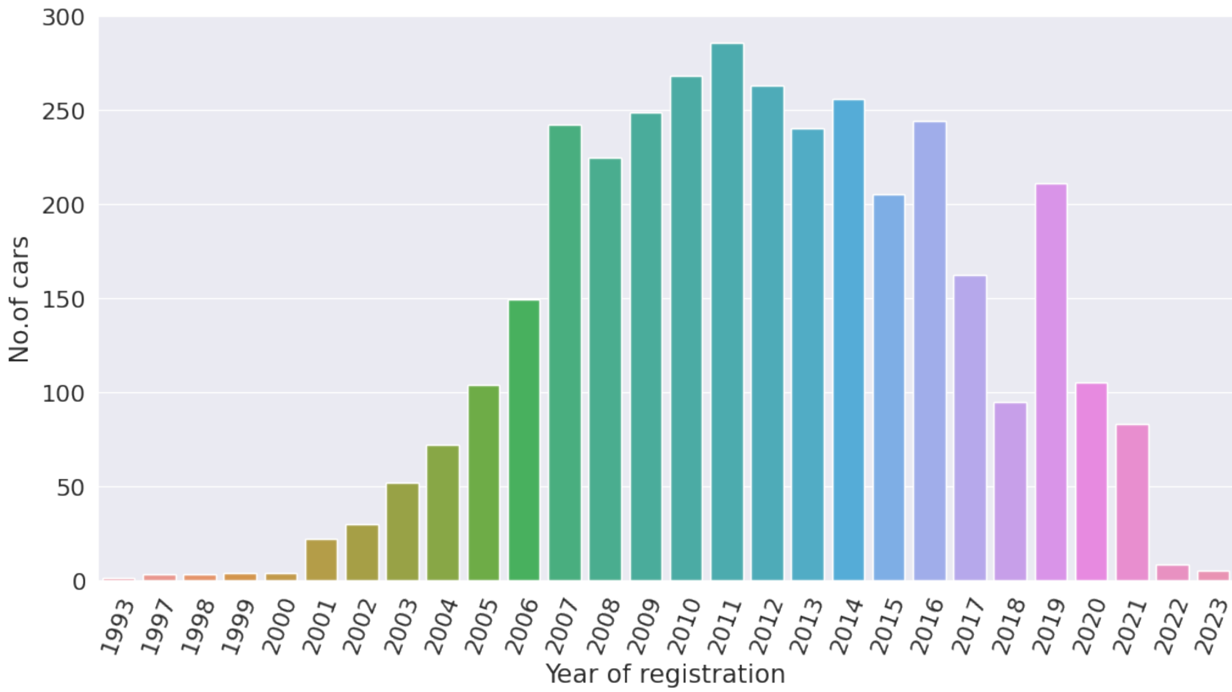
```
Vauxhall Corsa           222
Ford Focus               161
Vauxhall Astra           154
Ford Fiesta              128
Renault Clio             105
BMW 3 Series             105
Volkswagen Polo           96
Toyota Yaris              80
MINI Hatch                78
Mercedes-Benz C Class     58
Volkswagen 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});
```



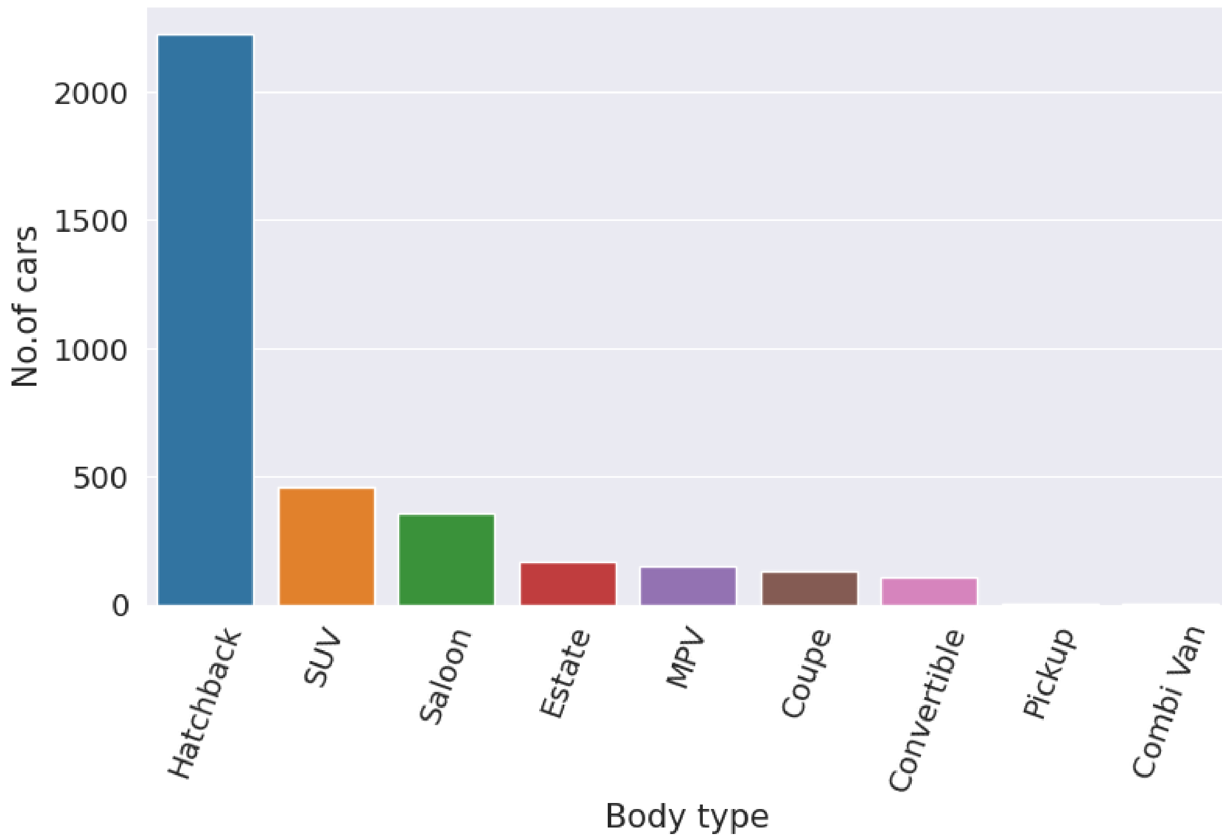
** Show the Year wise cars registered with the help of graph **

```
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});
```



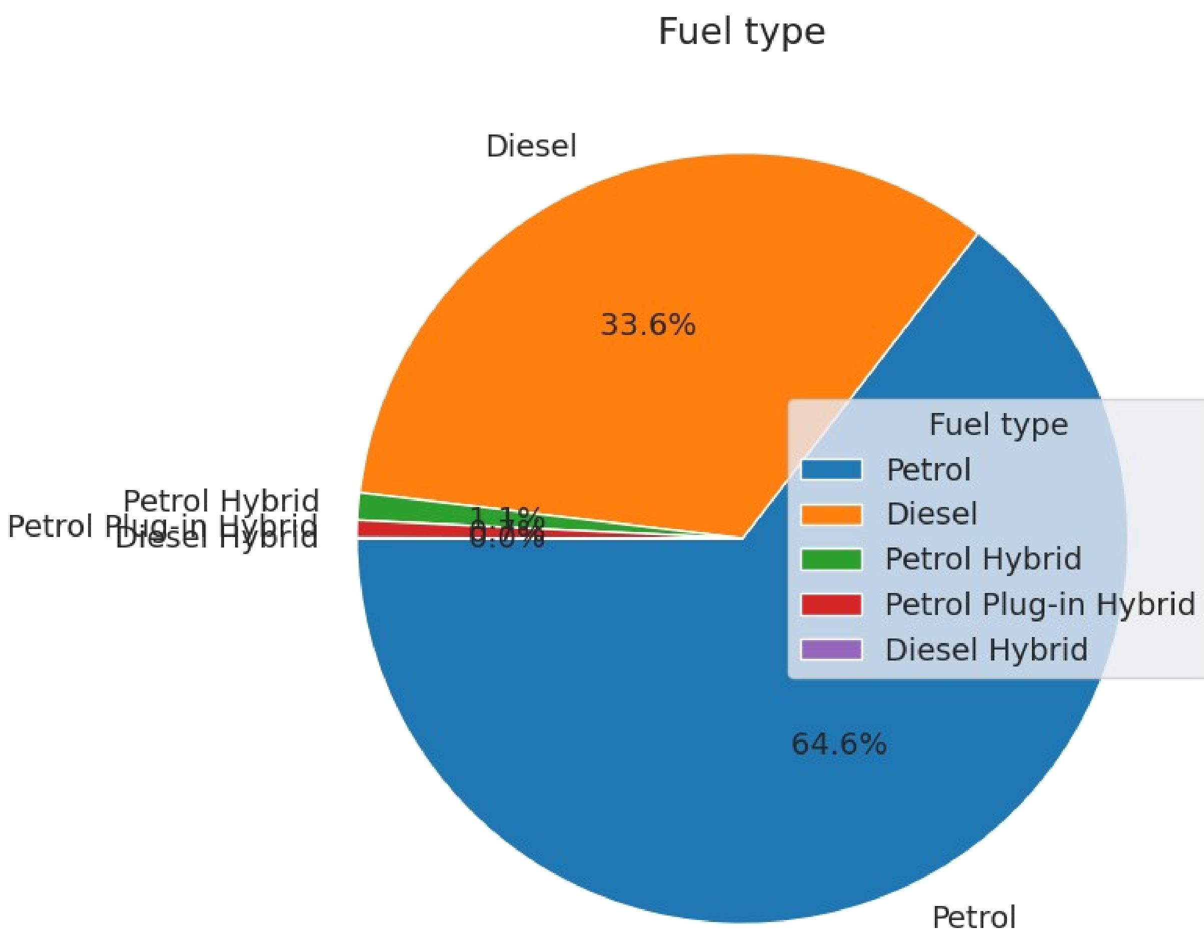
Which is the Most popular body type? also represent with the graph.

```
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});
```

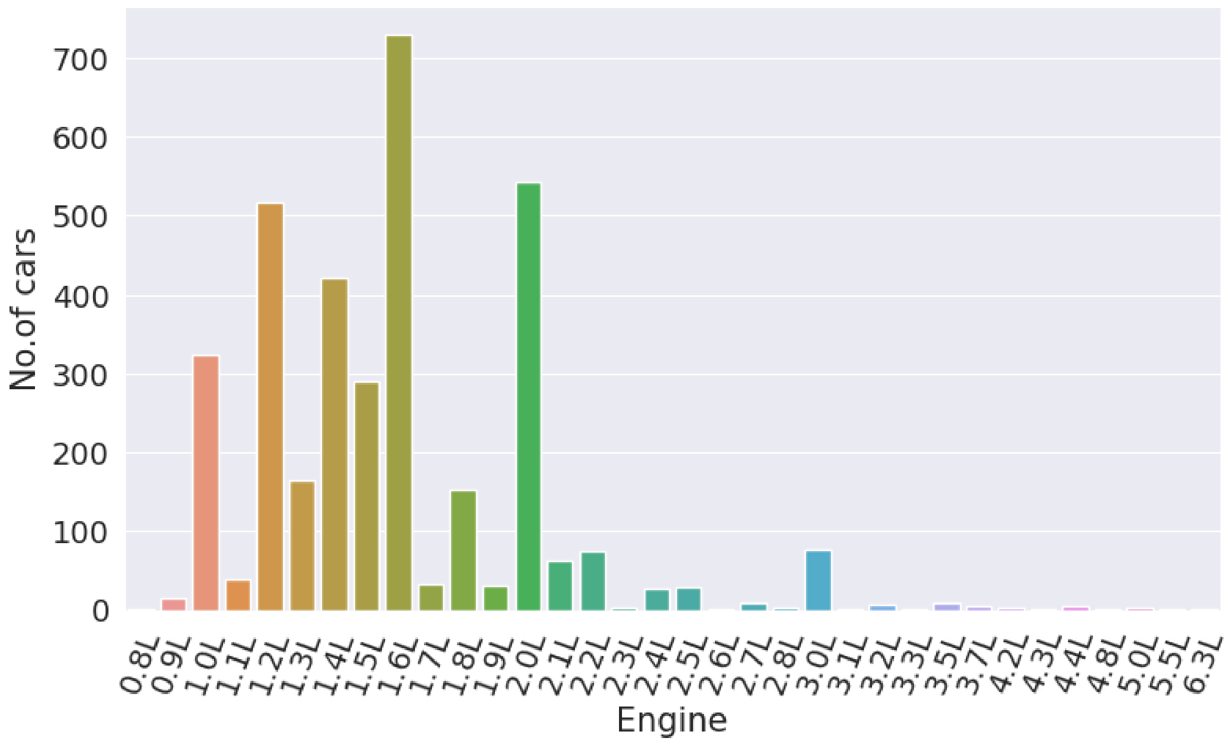


** What is the Most popular fuel type ? Represent with the pie chart & Graph. **

```
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)>
```

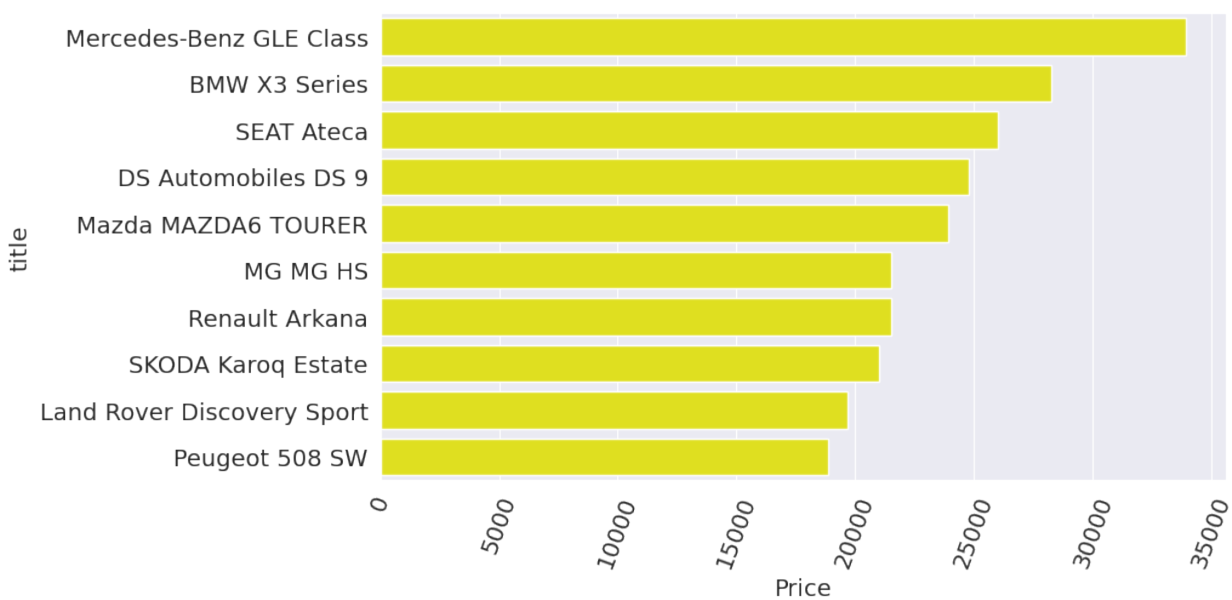



```
plt.xticks(rotation=70)
chart=sns.barplot(x=df['Engine'].value_counts().sort_index().index,
y=df['Engine'].value_counts().sort_index())
chart.set_ylabel('No.of cars', fontdict={'size': 15})
chart.set_xlabel('Engine', fontdict={'size': 15});
```



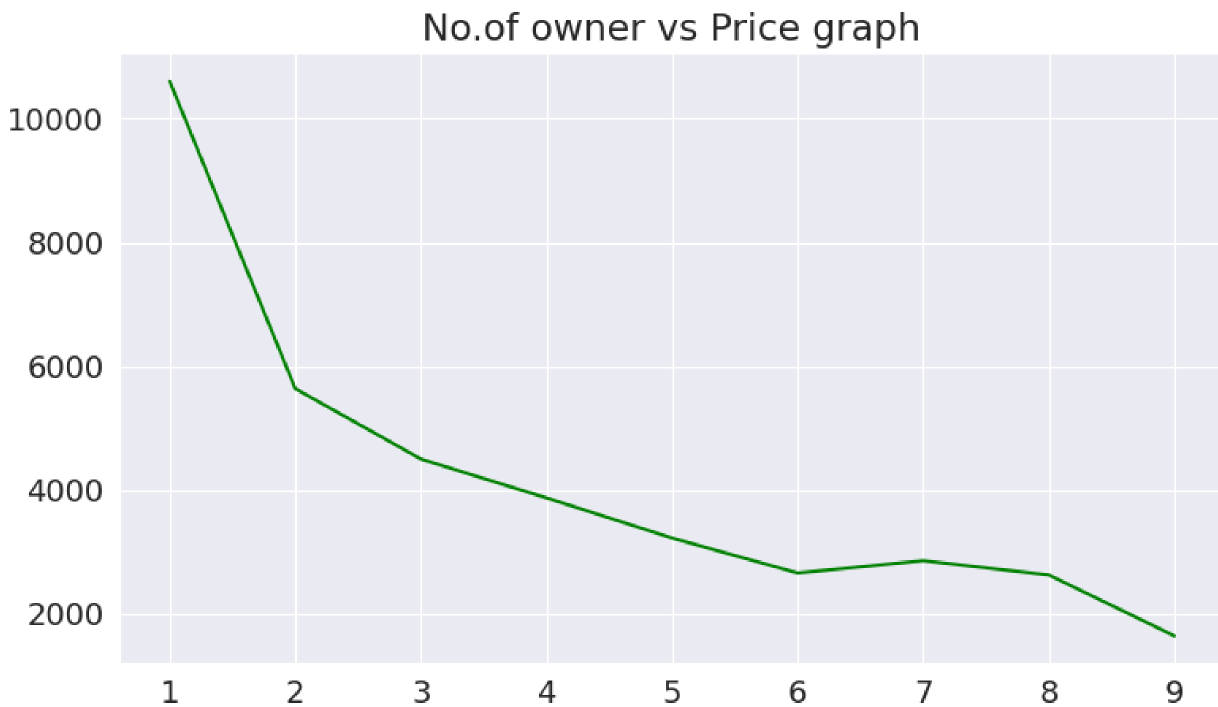
Which Cars are with highest average price? Represent with graph.

```
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');
```



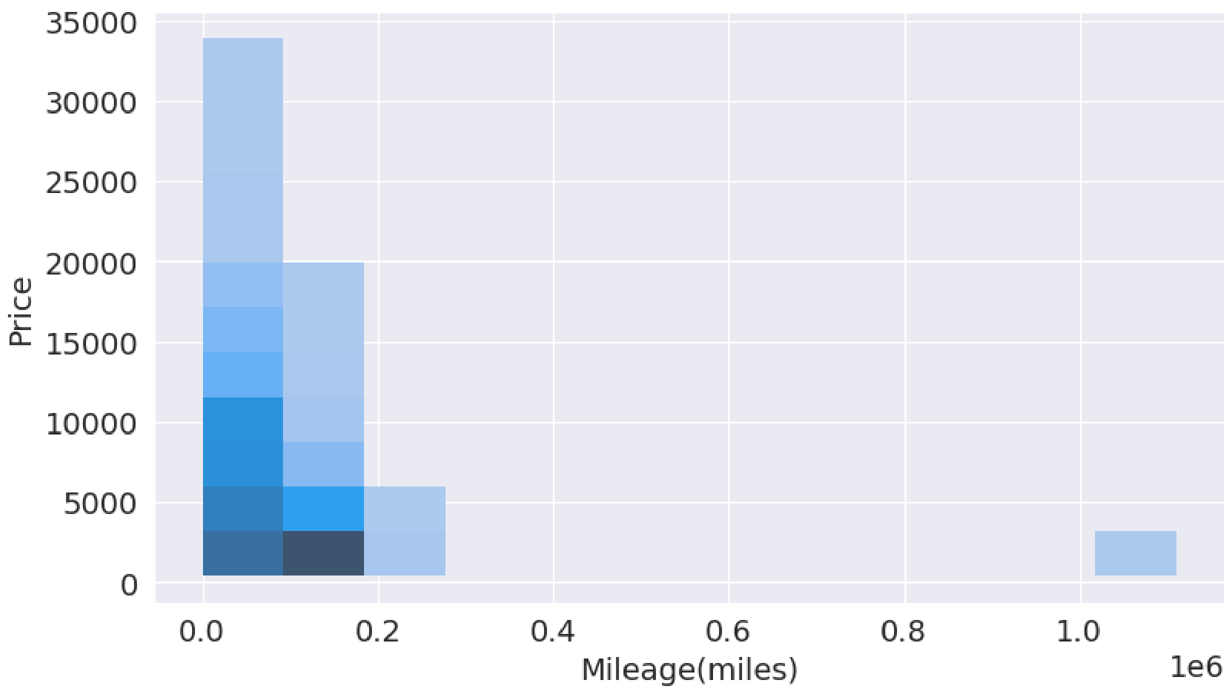
Show How is the Effect on price with the no. of owners?

```
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)

```python
# 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
```

```python
import pandas as pd
import numpy as np

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

```
                                      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]
```

```
df.head()
```

```
                           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
```

```
df.tail()
```

```
                                         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
```

```
df.shape
```

```
(14850, 5)
```

```
df.columns
```

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

```
rows,colums=df.shape
rows
```

```
14850
```

```
rows,columns=df.shape
columns
```

```
5
```

```
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
```

```
df.describe()
```

```
              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
```

```
df.isnull().sum()
```

```
Anime_name    0
episode       0
duration      0
members       0
Score         0
dtype: int64
```

```
df.isnull()
```

```
       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]
```

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

```
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)
```

```python
df['members'].unique
```

```
<bound method Series.unique of 0        3,263,142 members
1        2,505,884 members
2          614,907 members
```

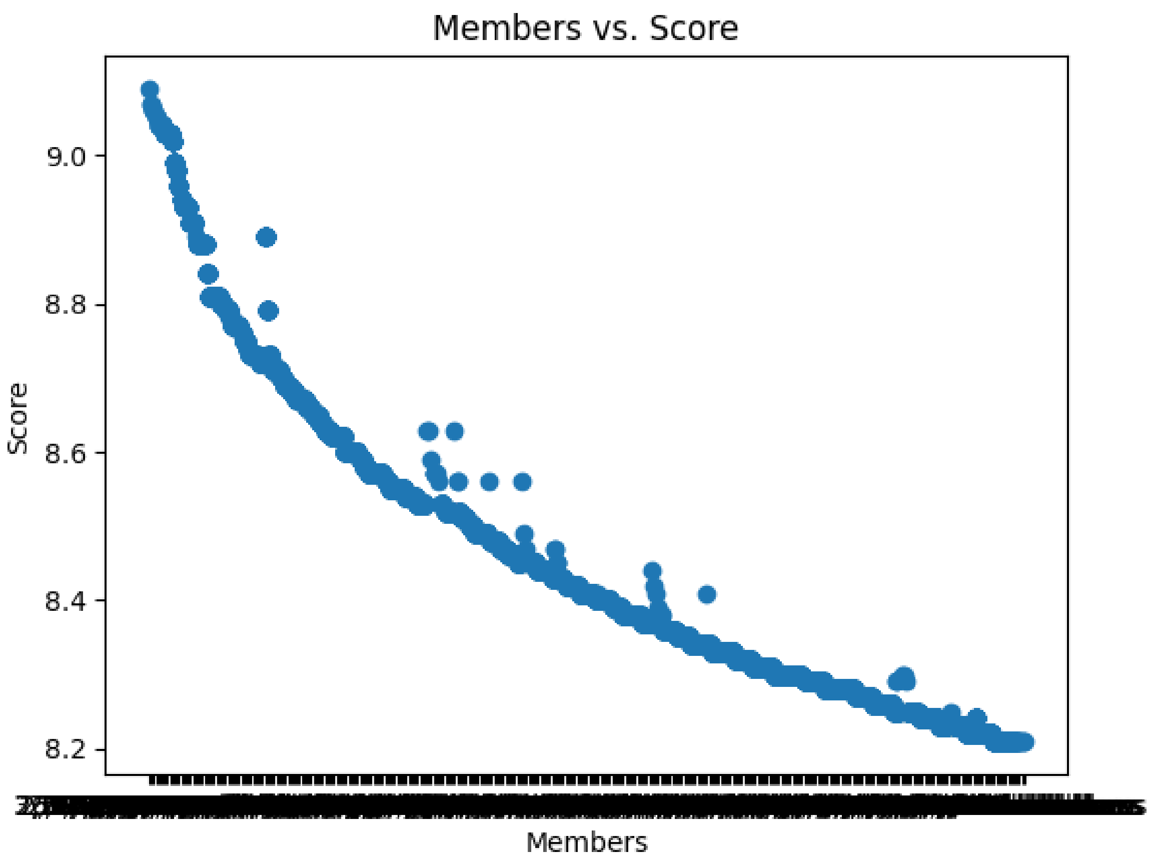
```
3         2,195,508 members
4           501,080 members
                ...
14845        58,615 members
14846       182,431 members
14847       214,440 members
14848       333,432 members
14849       612,395 members
Name: members, Length: 14850, dtype: object>
```

```python
df['members'].isnull().sum()
```

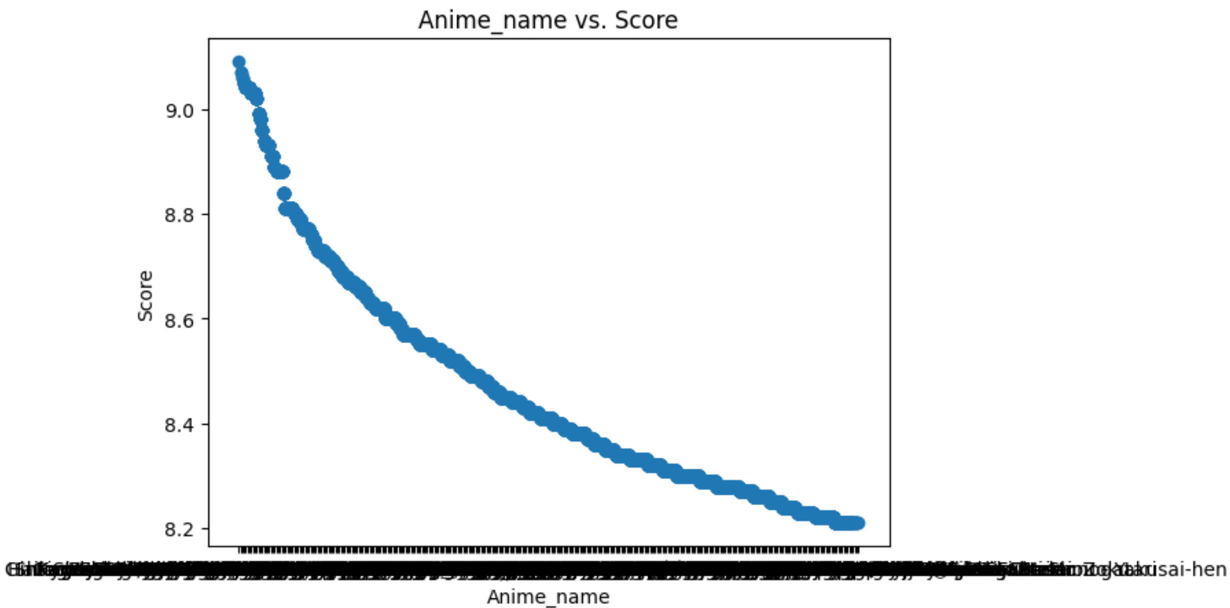
```
0
```

```python
import matplotlib.pyplot as plt
```

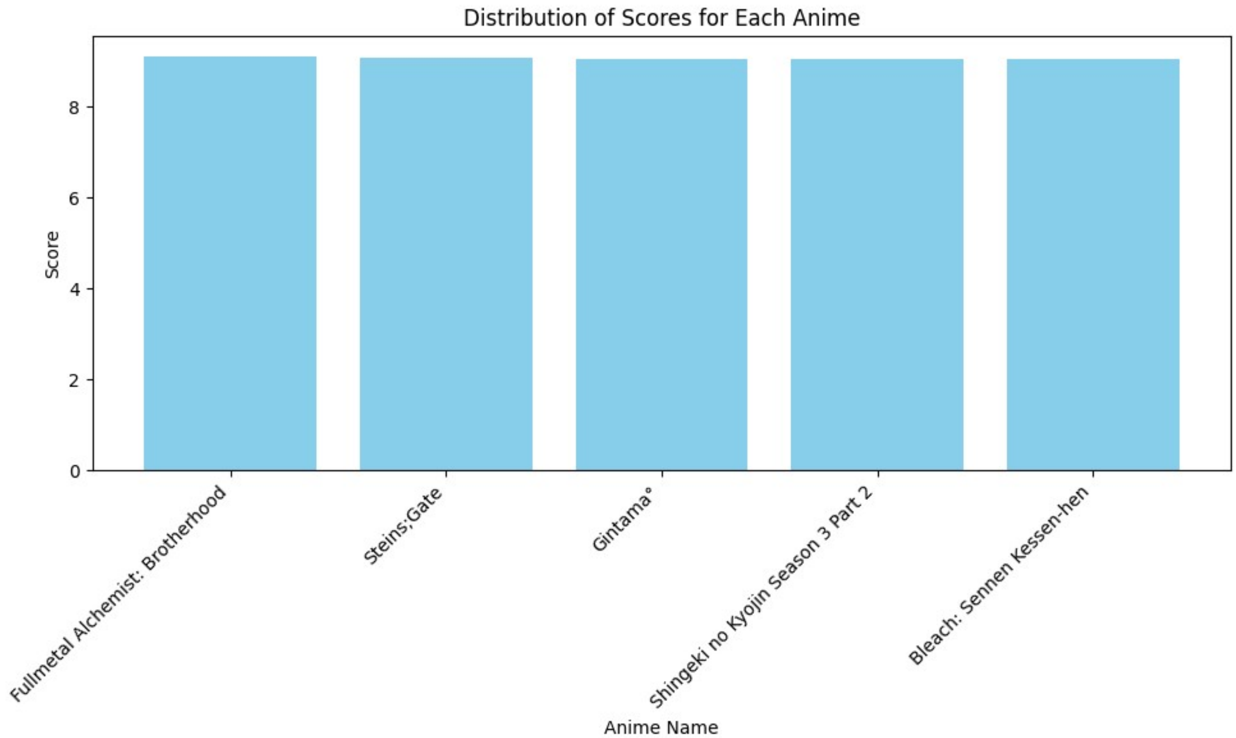
```python
plt.scatter(df['members'], df['Score'])
plt.title('Members vs. Score')
plt.xlabel('Members')
plt.ylabel('Score')
plt.show()
```



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




```
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?**

```
row,columns=df.shape
rows

14850

row,columns=df.shape
columns

5
```

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

```
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
```

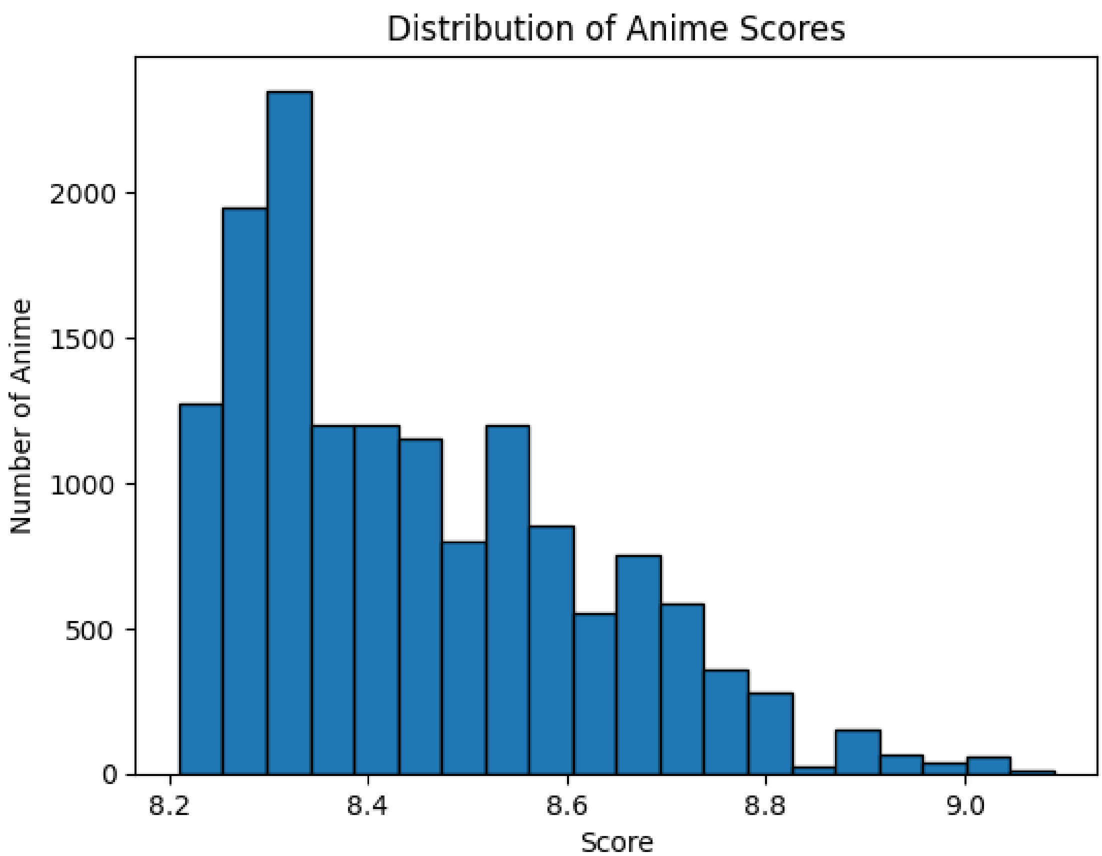
Question 3: What is the maximum and minimum score among the anime in the dataset?

```
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?

```python
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?

```python
# 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.
```

titanic-que-1

December 11, 2023

[1]:
```python
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  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
```

```
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)
```

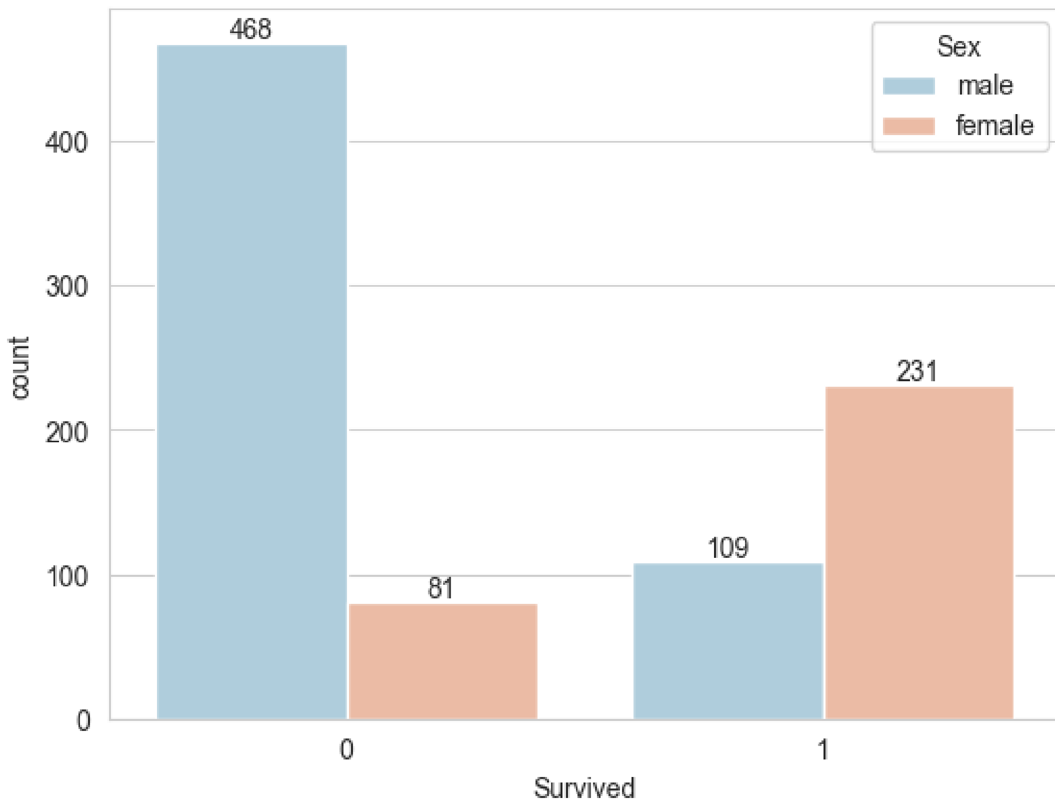
1 1. Find the maximum age and corresponding name

[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

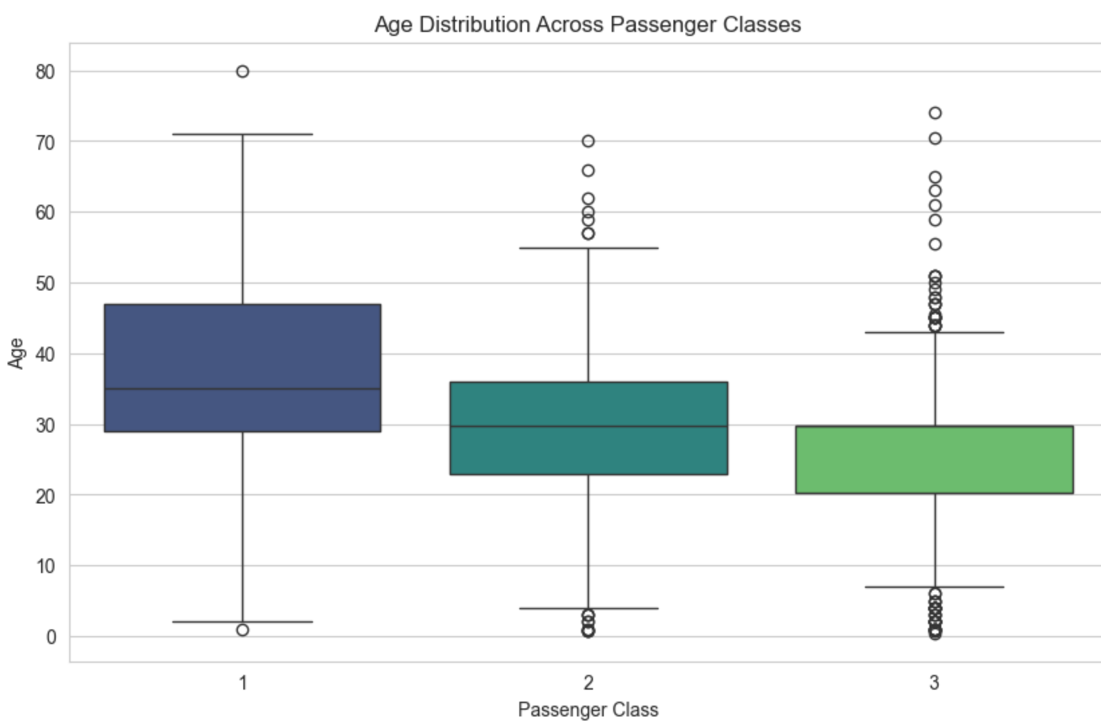
[11]:
```
sns.set_style('whitegrid')
dx = sns.countplot(x='Survived',hue='Sex',data=d,palette='RdBu_r')
for bars in dx.containers:
    dx.bar_label(bars)
```



3 3. The age distribution vary across different passenger classes

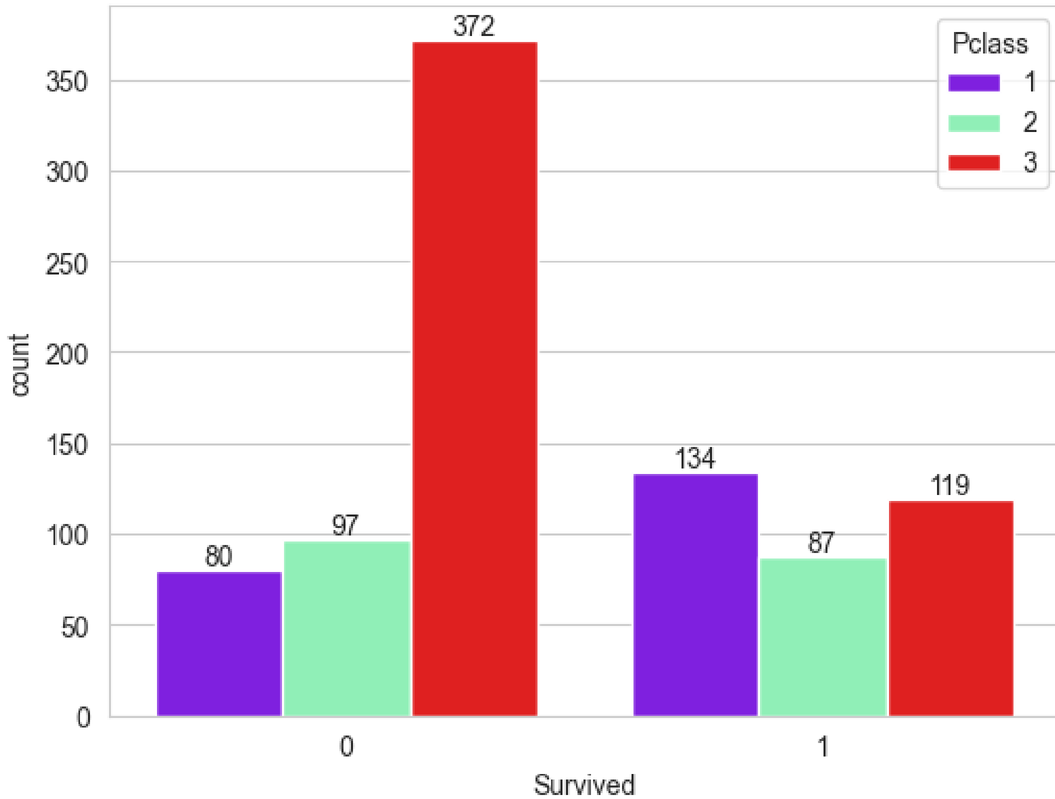
[12]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Pclass', y='Age', data=d, palette='viridis')
plt.title('Age Distribution Across Passenger Classes')
plt.xlabel('Passenger Class')
plt.ylabel('Age')
plt.show()
```

```
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')
```



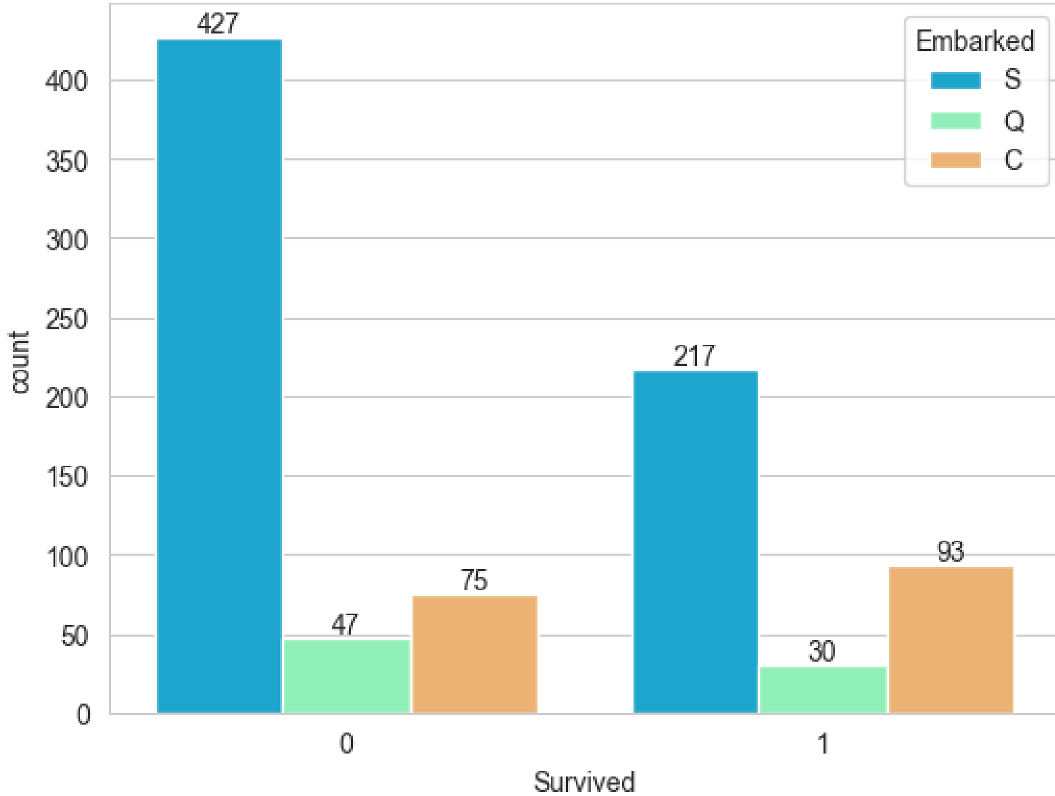
4 4. You visualize the distribution of survivors and non-survivors across different classes

[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)
```



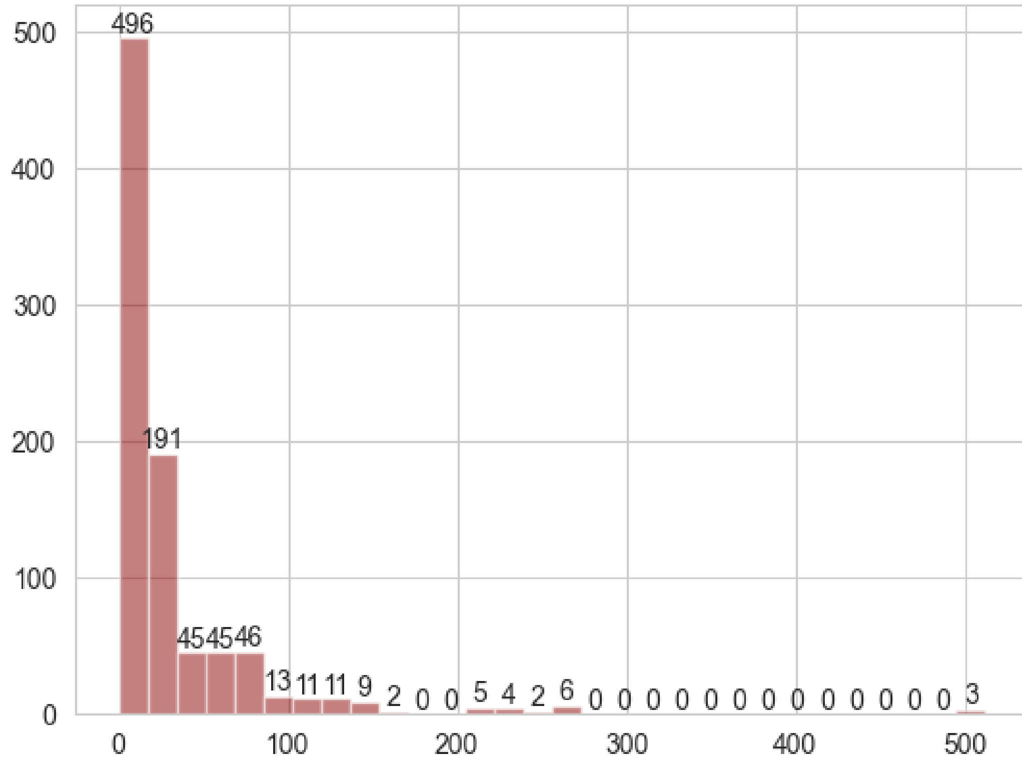
5 5. Correlation between the port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton) and survival

[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
0
100
200
300
400
500
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

ICC World Cup 2023

Import libraries

```
# 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

# 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")

# Display the first few rows of the World_cup DataFrame.
World_cup
```

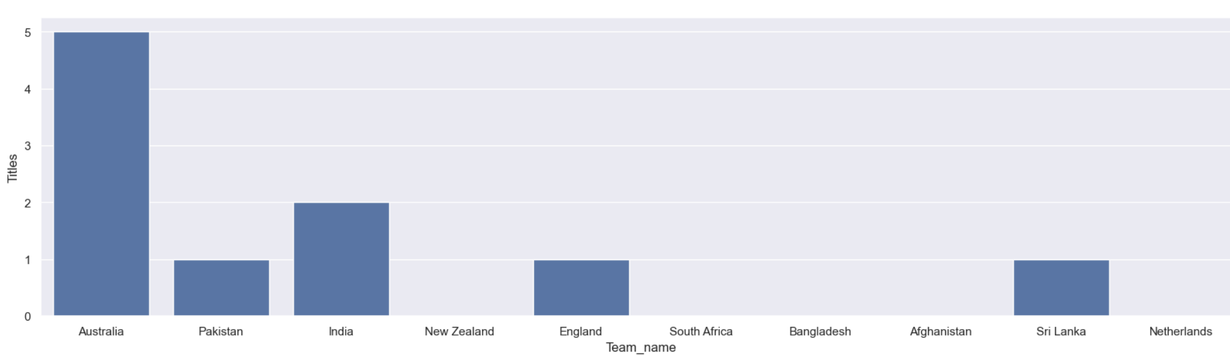
```
      Team_name  Team_ranking  Titles  Win_percentage_ODI  WC_matches
\
0     Australia             1       5               60.73          94

1      Pakistan             2       1               52.78          79

2         India             3       2               52.38          84

3   New Zealand             4       0               45.89          89

4       England             5       1               50.32          83

5  South Africa             6       0               61.00          64

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


   WC_match_won  Win_percent_WC  WC_match_loss  Loss_percent_WC  Tied
\
0            69           73.40             23            24.46     1

1            45           56.96             32            40.50     0
```

2	53	63.09	29	34.52	1
3	54	60.67	33	37.07	1
4	48	57.83	32	38.55	2
5	38	59.37	23	35.93	2
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

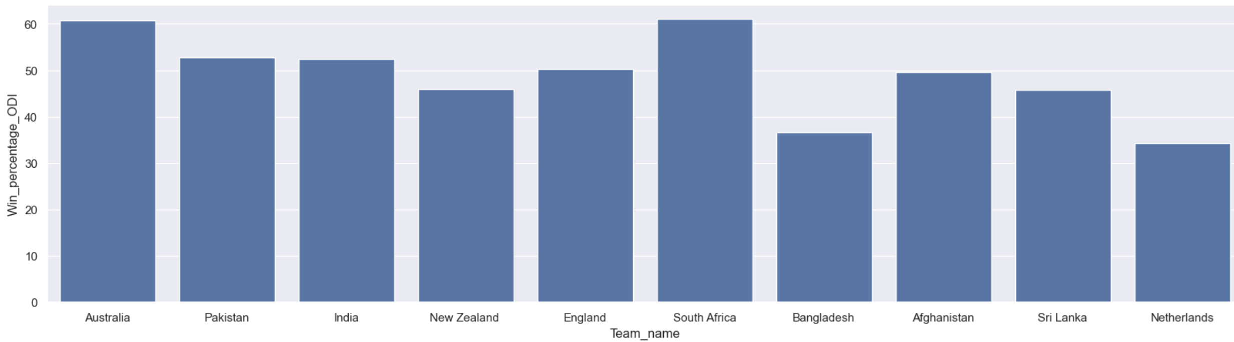
Q(1) No.of titles won by each teams

```
# 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()
```



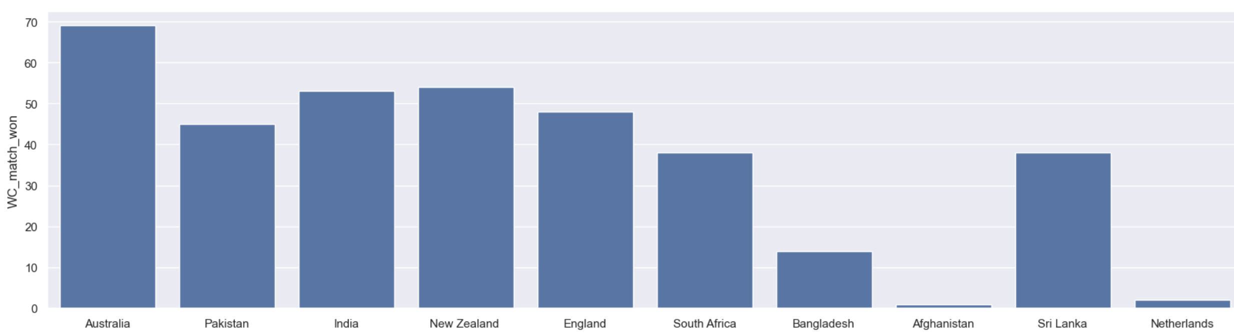
Q(2) Win percentage in ODI by each team

```
# 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()
```



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

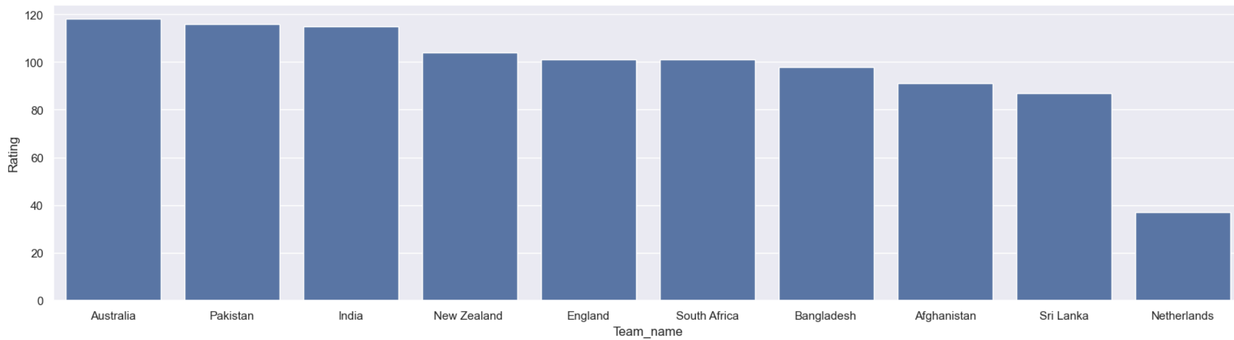
```
# Set the figure size for the bar plot using Seaborn
sns.set(rc={'figure.figsize':(20, 5)})

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

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



Q(4) Recent ICC ODI rating

```
# 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()
```




```
# Displaying the first few rows of the results DataFrame.
#results.head()

# 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)
```

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

```
# 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
```

```python
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()
```