PRACTICAL:1: DATA WRANLING-I

```
IN[1]: import pandas as pd

        import numpy as np

       import matplotlib.pyplot as plot

       import seaborn as sns

IN[2]: Data= pd.read_csv("Iris.csv")

IN[3]: Data.head(5)

IN[4]: Data.tail(5)

IN[5]: Data.isnull()

IN[6]: Data.describe()

IN[7]: Data.info()

IN[8]: Data.shape

IN[9]: Data.dtypes

IN[10]: Data.notnull()

IN[11]: Data.dropna()

IN[12]: Data.fillna(0)

IN[13]: Data.replace()

IN[14]: Data.columns

IN[15]: Data.columns IN[16]: Data.iloc[1]

IN[17]: Data.iloc[:,2]

IN[18]: Data.plot(kind='bar')
```

PRACTICAL:2:DATA WRANGLING-II

```
IN[1]: import pandas as pd
        import numpy as np
       from scipy import stats
       import matplotlib.pyplot as plt
IN[2]: df=pd.read_csv("a2.csv")
IN[3]: df.isnull()
IN[4]: df.notnull()
IN[5]: df.dropna()
IN[6]: df['VisITedResources']=df['VisITedResources'].fillna(df['VisITedResources'].mean())
IN[7]: df['VisITedResources']
IN[8]: df['VisITedResources'].fillna(value=77,inplace=True)
IN[9]: df
IN[10]: df.replace(to_replace=16, value=-99)
IN[11]: df.dropna(axis=1, how='all')
IN[12]: col=['VisITedResources','raisedhands','AnnouncementsView']
         df.boxplot(col)
IN[13]: fig, ax = plt.subplots(figsize=(18,10))
         ax.scatter(df['VisITedResources'],df['raisedhands'])
         ax.set_xlabel('VisITedResources')
         ax.set_ylabel('raisehands')
         plt.show()
IN[14]: z = np.abs(stats.zscore(df['VisITedResources']))
         print(z)
IN[15]: sorted_rscore = sorted(df['VisITedResources'])
         sorted_rscore
        q1 = np.percentile(sorted_rscore, 25)
        q3 = np.percentile(sorted_rscore, 75)
        print(q1,q3)
IN[16]: IQR = q3-q1
          lwr_bound=q1-(1.5*IQR)
         upr_bound=q3+(1.5*IQR)
         print (lwr_bound, upr_bound)
IN[17]: r_outliers = []
          lwr_bound = 20.0
          upr_bound = 84.0
          for i in sorted_rscore:
               if i < lwr_bound or i > upr_bound:
                    r_outliers.append(i)
            print(r_outliers)
IN[18]: import numpy as np
        df_stud = df
        ninetieth_percentile=np.percentile(df_stud['VisITedResources'],90)
       b=np.where(df_stud['VisITedResources'] > ninetieth_percentile,
                        ninetieth_percentile, df_stud['VisITedResources'])
      print("New array:", b)
IN[19]: import numpy as np
         import pandas as pd
         median = np.median(sorted_rscore)
         col = ['VisITedResources']
         df.boxplot(column=col)
        refined_df = df.copy()
        refined_df['VisITedResources'] = np.where(refined_df['VisITedResources'] < upr_bound, m
         refined_df['VisITedResources'] = np.where(refined_df['VisITedResources'] > lwr_bound, m
IN{20]: df['VisITedResources'].plot(kind='hist', bins=20)
         plt.title('Histogram of VisITedResources')
         plt.xlabel('VisITedResources')
         plt.ylabel('Frequency')
         plt.show()
         df['log_math'] = np.log10(df['VisITedResources'])
```

PRACTICAL:3: Describe statistic means user or central tendency and Variability

```
IN[1]: import numpy as np
       import pandas as pd
       import seaborn as sns
       import matplotlib.pyplot as plt
IN[2]: df=pd.read_csv("Iris.csv")
IN[3]: df.head(5)
IN[4]: df.info()
IN[5]: df.shape
IN[6]: df.describe()
IN[7]: df.isna()
IN[8]: df.count()
IN[9]: df.sum()
IN[10]: df.mean(numeric_only="True")
IN[11]: numeric_df = df.select_dtypes(include=['number'])
        result = numeric_df.median(axis=1)[0:4]
IN[12]: df.mode()
IN[13]: df.min()
IN[14]: df.max()
IN[15]: df['SepalLengthCm'].mean()
IN[16]: df.describe(include='object')
IN[17]: groupby_sum = df.groupby(['Species']).sum()
        groupby_sum
IN[18]: groupby_std = df.groupby(['Species']).std()
        groupby_std
IN[19]: groupby_median = df.groupby(['Species']).median()
         groupby_median
IN[20]: groupby_min = df.groupby(['Species']).min()
         groupby_min
IN[21]: groupby_max = df.groupby(['Species']).max()   groupby_max
IN[22]: groupby_quantile = df.groupby(['Species']).quantile()    groupby_quantile
```

PRACTICAL:7: TEXT ANALYSIS

```
IN[1]: import nltk
       nltk.download('punkt')
IN[2]: from nltk import word_tokenize, sent_tokenize
       sent = "Sachin is considered to be one of the greatest cricket players. Virat
    is the captain of the Indian cricket team"
    print(word_tokenize(sent))
    print(sent_tokenize(sent))
IN[3]: from nltk.corpus import stopwords
       import nltk
       nltk.download('stopwords')
       stop_words = stopwords.words('english')
       print(stop_words)
IN[4]: token = word_tokenize(sent)
       cleaned_token = []
       for word in token:
          if word not in stop_words:
          cleaned_token.append(word)
          print("This is the unclean version : ",token) print("This is the cleaned version : ",cleaned_token)
IN[5]: words = [cleaned_token.lower() for cleaned_token in cleaned_token
       If  cleaned_token.isalpha()]
       print(words)
IN[6]: from nltk.stem import PorterStemmer
       stemmer = PorterStemmer()
      port_stemmer_output = [stemmer.stem(words) for words in words]
      print(port_stemmer_output)
IN[7]: from nltk.stem import WordNetLemmatizer
        nltk.download('wordnet')
        lemmatizer = WordNetLemmatizer() lemmatizer_output = [lemmatizer.lemmatize(words) for
words in words]
print(lemmatizer_output)
```

```
IN[8]: from nltk import pos_tag
       import nltk
       nltk.download('averaged_perceptron_tagger')
      token = word_tokenize(sent)
     cleaned_token = []
      for word in token:
          if word not in stop_words:
         cleaned_token.append(word)
         tagged = pos_tag(cleaned_token)
         print(tagged)
IN[9]: from sklearn.feature_extraction.text import TfidfVectorizer
       from sklearn.metrics.pairwise import cosine_similarity
       import pandas as pd
IN[10]: docs = [ "Sachin is considered to be one of the greatest cricket players", "Federer is considered
one of the greatest tennis players", "Nadal is considered one of the greatest tennis players", "Virat is
the captain of the Indian cricket team"]
IN[10]: vectorizer = TfidfVectorizer(analyzer = "word", norm = None , use_idf = True,
        smooth_idf=True)
        Mat = vectorizer.fit(docs)
       print(Mat.vocabulary_)
IN[11]: tfidfMat = vectorizer.fit_transform(docs)
         print(tfidfMat)
IN[12]: features_names = vectorizer.get_feature_names_out()
         print(features_names)
IN[13]: dense = tfidfMat.todense()
        denselist = dense.tolist()
       df = pd.DataFrame(denselist , columns = features_names)
      df
IN[14]: features_names = vectorizer.get_feature_names_out()
         print(features_names)
IN[15]: dense = tfidfMat.todense()
         denselist = dense.tolist()
      df = pd.DataFrame(denselist , columns = features_names)
      df
IN[16]: docList = ['Doc 1','Doc 2','Doc 3','Doc 4']
      skDocsIfIdfdf=pd.DataFrame(tfidfMat.todense(),index=sorted(docList), columns=features_names)
      print(skDocsIfIdfdf)
IN[17]: csim = cosine_similarity(tfidfMat,tfidfMat)
         csimDf =pd.DataFrame(csim,index=sorted(docList),columns=sorted(docList))
          print(csimDf)
```

PRACTICAL:8: DATA VISUALIZATION-I

```
IN[1]: import seaborn as sns
         import pandas as pd
         titanic = sns.load_dataset("titanic")
        titanic
IN[2]: titanic.info()
IN[3]: x=titanic["fare"]
IN[4]: titanic.describe()
IN[5]: titanic.info()
IN[5]:titanic_cleaned=titanic.drop(['pclass','embarked','deck','embark_town'],axis=1)
titanic_cleaned.head(15)
IN[6]: titanic_cleaned.info()
IN[7]: titanic_cleaned.isnull().sum()
IN[8]: titanic_cleaned.corr(method='pearson')
IN[9]: sns.histplot(data=titanic,x="fare",bins=8)
IN[10]: sns.histplot(data=titanic,x="fare",binwidth=10)
IN[11]: sns.histplot(data=titanic,x="fare",bins=20,binwidth=10)
IN[12]: sns.histplot(data=titanic,x="fare",binwidth=20)
IN[13]: sns.histplot(data=titanic,x="fare",binwidth=1)
IN[14]: sns.histplot(data=titanic,x="fare",bins=20,binwidth=50)
```