Data Wrangling I
Perform the following operations using Python on any open source dataset (eg. data.csv)
1. Import all the required Python Libraries.
2. Locate an open source data from the web (eg. https://www.kaggle.com). Provide a clear description of the data and its source (i.e. URL of the web site).
3. Load the Dataset into pandas dataframe.
4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

```
#1. Import all the required Python Libraries.
import pandas as pd

#2. Locate an open source data from the web
#3. Loading the Dataset into pandas dataframe.
df = pd.read_csv('/assign_1_data.csv')

df.head()

#4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe()
#function to get some initial statistics. Provide variable descriptions. Types of variables etc.
df.describe()


df.info()

#Checking the dimensions of the data frame.
df.shape

# Dropping rows with NaN values
new_df = df.dropna()
new_df

# Filling Calories column NaN values with mean value
new_df_2 = df
```

```
mean = new_df_2['Calories'].mean()
new_df_2['Calories'] = new_df_2['Calories'].fillna(mean)
new_df_2
```

```
# Formatting wrong data to its original type
df['Date'] = pd.to_datetime(df['Date'])
df.dropna(subset=['Date'], inplace=True)
df
```

```
# Handling wrrong data
for i in df.index:
  if df.loc[i, "Duration"] > 120:
    df.drop(i, inplace=True)
df
```

```
# Removing Duplicates
df.drop_duplicates(inplace=True)
df
```

//

Q2)

Consider the “Academic performance” dataset of students (xAPI-Edu-Data.csv) and perform the following operations using Python.

Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.

Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

Reason and document your approach properly.

```
import pandas as pd



test=pd.read_csv('xAPI-Edu-Data.csv')
test


test.describe()


#number of null values in each column
missing_values=test.isnull().sum()
print(missing_values)


#null element's percentage of each column
missing_percentage=100*missing_values/len(test)
print(missing_percentage)


import numpy as np

#Dealing null values in numerical data: As we have 3 columns of numerical data

#Replacing null values in AnnouncementsView by mean
test['AnnouncementsView']=test['AnnouncementsView'].replace(np.NaN,test['AnnouncementsVi
ew'].mean())
test['AnnouncementsView'].isnull().sum()


#Replacing null values in raisedhands by median
test['raisedhands']=test['raisedhands'].replace(np.NaN,test['raisedhands'].median())
test['raisedhands'].isnull().sum()


#Replacing null values in VisITedResources by mean
x=test['VisITedResources'].mean()
test['VisITedResources']=test['VisITedResources'].replace(np.NaN,round(x))
test['VisITedResources'].isnull().sum()


#Dealing with null values in categorical data. We can see there are 5 columns of categorical
```

data having null values

```
#Replacing null values in Topic by most_frequent in that column
from sklearn.impute import SimpleImputer
impute1 = SimpleImputer(strategy='most_frequent',missing_values=np.NaN)
test['Topic'] = impute1.fit_transform(test[['Topic']])
test['Topic'].isnull().sum()

#Replacing null values in Relation by "not known"
test['Relation']=test['Relation'].replace(np.NaN,'not known')
test['Relation'].isnull().sum()

#Replacing null values in ParentschoolSatisfaction by most_frequent in that column
impute3 = SimpleImputer(strategy='most_frequent',missing_values=np.NaN)
test['ParentschoolSatisfaction'] = impute3.fit_transform(test[['ParentschoolSatisfaction']])
test['ParentschoolSatisfaction'].isnull().sum()

#Replacing null values in Class by most_frequent in that column
impute4 = SimpleImputer(strategy='most_frequent',missing_values=np.NaN)
test['Class'] = impute4.fit_transform(test[['Class']])
test['Class'].isnull().sum()


#Replacing null values in StudentAbsenceDays by "missing" word
test['StudentAbsenceDays']=test['StudentAbsenceDays'].replace(np.NaN,'missing')
test['StudentAbsenceDays'].isnull().sum()


#check if any null values left
missing_values=test.isnull().sum()
print(missing_values)


import matplotlib.pyplot as plt
import seaborn as sns

import seaborn as sns.boxplot(data=test['raisedhands'],x=test['raisedhands'])


import seaborn as sns
sns.boxplot(data=test['AnnouncementsView'],x=test['AnnouncementsView'])

import seaborn as sns
```

```
sns.boxplot(data=test['Discussion'],x=test['Discussion'])


import seaborn as sns
sns.boxplot(data=test['VisITedResources'],x=test['VisITedResources'])


numeric = list(test.select_dtypes(include=np.number).columns)
numeric

figure = plt.figure(figsize=(10, 7))
for i in range(len(numeric)):
    plt.subplot(2, 2, i+1)
    sns.histplot(test[numeric[i]], bins=10, kde=True)



from sklearn.preprocessing import PowerTransformer, QuantileTransformer
# Manual Transform
figure = plt.figure(figsize=(15, 7))
transform_data = test["AnnouncementsView"]
data = transform_data**(0.557)
plt.subplot(1, 3, 1)
sns.histplot(data, kde=True)
# using power-transformer
transformer = PowerTransformer()
data = transformer.fit_transform(np.array(transform_data).reshape(-1, 1))
plt.subplot(1, 3, 2)
sns.histplot(data, kde=True)
# using quantile-transformer
transformer = QuantileTransformer(n_quantiles=50, output_distribution='normal')
data = transformer.fit_transform(np.array(transform_data).reshape(-1, 1))
plt.subplot(1, 3, 3)
sns.histplot(data, kde=True)
```

//

Q 3. Perform the following operations on Age-Income dataset (Age- Income-Dataset.csv) Provide summary statistics (mean, median, minimum, maximum, standard deviation) for numeric variables with and without using any library functions. Provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
import csv

df1 = pd.read_excel('Age-Income-Dataset.xlsx')
df1

#Mean
# calculate mean by formula
mean_income = sum(df1['Income'])/len(df1['Income'])
print(mean_income)
# Using Pandas function
df1['Income'].mean()

#Median
# Calculate Median by formula
n = len(df1['Income'])
if n % 2:
    income_median = sorted(df1['Income'])[round(0.5*(n-1))]
else:
    x_ord, index = sorted(df1['Income']), round(0.5 * n)
    income_median = 0.5 * (x_ord[index-1] + x_ord[index])

print(income_median)
# Using Pandas function
df1['Income'].median()


#Mode
# Using Pandas function
df1['Income'].mode()[0]


#Calculating Measures of Variability or Dispersion
# Calculating Variance using Formula (without libraries)
n = len(df1['Income'])
income_mean = sum(df1['Income']) / n
income_var = sum((item - income_mean)**2 for item in df1['Income']) / (n - 1)
print(income_var)
```

```
# Finding Variance using Pandas
df1.var(ddof=1)[0]

# Calculating Variance using Formula
income_std = income_var**0.5
print(income_std)

# Finding Variance using  Pandas
df1.std(ddof=1)[0]


# Calculating Skewness using formula
income_skew = (sum((item - mean_income)**3 for item in df1['Income'])* n / ((n - 1) * (n - 2) *
income_std**3))
income_skew

# Finding Skewness using Libraries Pandas Library function
df1.skew()[0]


norm = df1
norm.plot(kind = 'density')
print('This distribution has skew', norm.skew())

#Provide summary statistics of income grouped by the age groups. Create a list that contains a
numeric value for each response to the categorical variable.
df1[['Income' , 'Age']].groupby('Age').mean()


df1.Age.replace({'Old':1 ,'Middle Age':2 ,'Young':3}, inplace=True)
df1.head()
```

//

Q 3. Write a Python program to display some basic statistical details
like percentile, mean, standard deviation etc. of the species of ‘Iris- setosa’, ‘Iris-versicolor’ and ‘Iris-virginica’ of iris.csv dataset.
Calculate the measures of variability. Calculate and provide the visualization of the Correlation among the variables.

```
print('This distribution has kurtosis', norm.kurt())


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

```
df2['Species'].value_counts()
```

```
df2.groupby("Species").describe()
```

```
#Calculating mean of given species
df2.describe()
```

####Calculating Measures of Correlation
The two statistical measures for correlation are covariance and correlation coefficient. Covariance indicates the strength and direction of a relationship between the pairs of variables.When the covariance is positive, the correlation is Positive. A higher value of covariance indicates stronger relationship.
When the covariance is negative, the correlation is Negative. A lower value of covariance indicates a stronger relationship.
When the covariance is close to zero, it indicates that correlation is weak.

```
df2.cov()
```

#Correlation Coefficient also known as Pearson’s Correlation Coefficient is another measure to find out the relation between two variables.
The value $r > 0$ indicates positive correlation.
The value $r < 0$ indicates negative correlation.
The value $r = 1$ is the maximum possible value of r. It corresponds to a perfect positive linear relationship between variables.
The value $r = -1$ is the minimum possible value of r. It corresponds to a perfect negative linear relationship between variables.
The value $r \approx 0$, or when r is around zero, means that the correlation between variables is weak.

```
df2.corr()
```

```
df2.groupby("Species").corr()
```

```
#Visualization
import matplotlib.pyplot as plt
fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(df2[df2.columns[i]], df2[df2.columns[j]], color='blue')
        axes[ax1][ax2].set_xlabel(df2.columns[i])
        axes[ax1][ax2].set_ylabel(df2.columns[j])
        index = index + 1
```

#Correlation matrix to heat map
The basic idea of heatmaps is that they replace numbers with colors of varying shades, as indicated by the scale on the right. Cells that are lighter have higher values of r. This type of visualization can make it much easier to spot linear relationships between variables than a table of numbers.

```
import seaborn as sns
sns.heatmap(df2.corr());
```

```
#Calculating Measures of Variability or Dispersion

df2['PetalWidthCm'].var(ddof=1)

df2['PetalWidthCm'].std(ddof=1)

df2.skew()
```

///
/////////////

TC09 Assignment 4
Q1. Consider the Bangalore House Price Data. Perform following operations. a) Find and replace null values in the data using appropriate technique. b) Transform the ‘Size’ column to numerical values. For Example: 2 BHK to be converted as 2 c) Transform the ‘total_sqft’ column to contain numerical values on same scale. If the range is given average value of the range to

be taken. d) Calculate and add one more column as ‘Price_Per_Sqft’ e) Remove the outliers from Price_Per_Sqft and BHK Size column if any. f) Apply the Linear Regression model to the data and display the training and testing performance measures as Mean Squared Error and Accuracy

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

```
df = pd.read_csv('Banglore Housing Prices.csv')
df
```

```
df.dtypes
```

#The total_sqft column has string values so we have to convert them into integers and then extract these integers from the string. Then copy the data in a variable.

```
def extract_int(s):
    nums = []
    i = 0
    while s[i].isdigit():
        nums.append(s[i])
        i+=1
    return ''.join(nums)

def remove_sym(s):
    s = s.replace('.', '')
    return s.replace(' ', '')

sqft = df['total_sqft'].to_list()
new_sqft = []
for area in sqft:
    try:
        if '.' in area:
            num = area.split('.')
            new_sqft.append(int(num[0]))
            continue
        new_sqft.append(int(area))
    except:
```

```
        if remove_sym(area).isalnum():
            a = extract_int(area)
            new_sqft.append(int(a))
            continue
        n1, n2 = area.split(' - ')
        mean = (int(n1) + int(n2))/2
        new_sqft.append(int(mean))
df['total_sqft'] = new_sqft


#Checking for missing values
df.isnull().sum()

#Extract the string values of size column
size = df['size'].to_list()
new_sizes = []
for s in size:
    try:
        num, string = s.split()
        new_sizes.append(int(num))
    except:
        pass
new_sizes = np.array(new_sizes)
median_rooms = int(np.median(new_sizes))


#Replacing null values with median
df['size'].replace(np.nan, f'{median_rooms} BHK', inplace=True)

df['bath'].replace(np.nan, df['bath'].median(skipna=True), inplace=True)
df

#Calulating the least favourite values
least_fav = df['location'].value_counts()
least_fav

#calculating the least frequent localities
least_fav = least_fav[least_fav <= 10]
least_fav

# remove those localities that occur less than 10 times
df['location'] = df['location'].apply(lambda x: 'other' if x in least_fav else x)
```

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


# perform one-hot encoding for locations
encoded = pd.get_dummies(df['location'])
# concat attributes
df = pd.concat([df, encoded], axis = 1)
df

#Value counts of size column
size = df['size'].to_list()

new_sizes = []
for s in size:
    num, string = s.split()
    new_sizes.append(int(num))
df['size'] = new_sizes


# calculating price per sqft
df['price_per_sqft'] = (df['price']*100000)/df['total_sqft']

#Removing outiers for size
def remove_size_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('size'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                'std': np.std(bhk_df.price_per_sqft),
                'count': bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('size'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices,
bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')
df = remove_size_outliers(df)
df.shape
```

```
# drop loaction as one hot encoding is now merged with our dataframe
df.drop(columns='location', inplace=True)

df.head()

# drop price_per_sqft
df.drop(columns='price_per_sqft', inplace=True)


#remove outliers for bathrooms
df = df[df.bath < df.size + 2]
df.shape


sns.boxplot(data=df['price'])


from sklearn.preprocessing import PowerTransformer
# using power transformer for removing outliers
transformer = PowerTransformer()
df['price'] = transformer.fit_transform(np.array(df['price']).reshape(-1,1))


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
reg = LinearRegression()
reg.fit(x_train, y_train)
score = reg.score(x_test, y_test)
y_pred = reg.predict(x_test)
# r2_score
print('Test R2 Score', score)
print('Train R2 Score', reg.score(x_train, y_train))


# mean_squared_error
from sklearn.metrics import mean_squared_error
error = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', error)
```

//
Data Analytics III
1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
II. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

#Imported the required libraries and loaded the iris dataset into the frame.The dataset was present in sklearn module

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
iris=load_iris()

dir(iris)
```

#Loaded the dataset in the df variable with their column_names as feature_names

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df
```

#Added the target column as which flower it is

```
[ ]
df['target']=iris.target
df
```

#Displayed the correlation between the variables

```
[ ]
df.corr()


[ ]
x=df.drop(['target'],axis='columns')
y=df.target
[ ]
x

[ ]
y
```

#Imported train_test_split and splitted the dataset in trainning and testing set

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

[ ]
len(x_test)

#imported naive_byes and fitted the model in gaussianNB

[ ]
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
model.score(x_test,y_test)

[ ]
model.predict([[5.1,3.5,1.4,0.2]])

#Imported confusion matrix and displayed it and discovered that there are no errors in the predicted result

[ ]
from sklearn.metrics import confusion_matrix
cm=confusion_matrix
y_pred=model.predict(x_test)
cm(y_test,y_pred)

[ ]
from sklearn.metrics import accuracy_score,precision_score,recall_score
A_s=accuracy_score(y_test,y_pred)
A_s

[ ]
P_s=precision_score(y_test,y_pred,pos_label='positive',average='micro')
P_s

[ ]
R_s=recall_score(y_test,y_pred,pos_label='positive',average='micro')

R_s

//
/

Assingment 7 Data visulization 1

#Importing the required libraries and loading the dataset into the frame the dataset is imported from seaborn library

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

import matplotlib.pyplot as mat
import seaborn as sns

df = sns.load_dataset("titanic")
df

[ ]
df.head(10)


[ ]
df.shape

[ ]
df.isna().sum()



[ ]
df.drop("deck",axis=1,inplace = True)
df


[ ]
df = df.dropna()
df
```

[ ]
```
df.isna().sum()
```

[ ]
```
sns.histplot(data=df)
```

[ ]
```
sns.displot(df,x="fare")
```

[ ]
```
sns.displot(df,x="fare",binwidth=50)
```

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

[ ]
```
sns.displot(df,x="pclass")
```

[ ]
```
sns.displot(df,x="class")
```

[ ]
```
df.shape
```

[ ]
```
sns.displot(df,x="fare",hue="pclass",multiple="stack",binwidth=40)
```

[ ]
```
sns.displot(df,x="sex",hue="survived",multiple="stack",binwidth=40)
```

[ ]
```
df.dtypes
```

[ ]

```
sns.displot(df,x="sex",hue="alone",multiple="stack",binwidth=40)
```

[ ]

```
sns.displot(df,x="class",hue="survived",multiple="stack",binwidth=40)
```

[ ]

```
sns.displot(df,x="pclass",hue="survived",multiple="stack",binwidth=40)
```

[ ]

```
sns.displot(df,x="fare",col="sex",binwidth=40)
```

[ ]

```
sns.barplot(x="sex",y="age",data=df)
```

[ ]

```
sns.countplot(x="sex",data=df)
```

#displayed no of survived or non-survived frome ach embark town pclass wise

[ ]

```
sns.catplot(x="embark_town",hue="survived",kind='count',col="pclass",data=df)
```

#plotted the boxplot and detected the outliers there are 3 outliers in the male column of age

[ ]

```
sns.boxplot(x="sex",y="age",data=df)
```

[ ]

```
sns.boxplot(x="sex",y="age",data=df,hue="survived")
```

[ ]

```
sns.violinplot(x="sex",y="age",data=df)
```

[ ]

```
sns.violinplot(x="sex",y="age",data=df,hue="survived",split=True)
```

[ ]

```
sns.stripplot(x="sex",y="age",data=df)
```

[ ]
```
sns.stripplot(x="sex",y="age",data=df,hue="survived",split=True)
```

[ ]
```
sns.swarmplot(x="sex",y="age",data=df)
```

[ ]
```
sns.swarmplot(x="sex",y="age",data=df,hue="survived")
```

[ ]
```
sns.boxplot(x="age",y="sex",data=df)
```

[ ]
```
sns.boxplot(x="sex",y="age",data=df,hue="survived")
```

[ ]
```
sns.scatterplot(data=df)
```

#These were some of the other plots use for visulization

//
///////////

Assignment 8 Data Viualization 2

Importing the required libraries and loading the dataset into the frame the dataset is imported from seaborn library

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

import matplotlib.pyplot as mat
import seaborn as sns

df = sns.load_dataset("titanic")
df
```

[ ]
```
df.head(10)
```

[ ]
```
df.shape
```

#check for the number of null values

[ ]
```
df.isna().sum()
```

#dropping the deck column as it has large number of null values

[ ]
```
df.drop("deck",axis=1,inplace = True)
df
```

#dropping the other nan values

[ ]
```
df = df.dropna()
df
```

[ ]
```
df.isna().sum()
```

#plotting the histogram

[ ]
```
sns.histplot(data=df)
```

#plotting with 'fare' attribute and it is concluded that there many passengers having fare between 0-50 and few with greater than 100

[ ]
```
sns.displot(df,x="fare")
```

[ ]
```
sns.displot(df,x="fare",binwidth=50)
```

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

#most number of passengers are from pclass3 then from pclass1 and then from pclass2

[ ]
sns.displot(df,x="pclass")

#Most of the passengers were travelling from third class.

[ ]
sns.displot(df,x="class")

[ ]
df.shape

#Pclass wise distribution of fare is shown here

[ ]
sns.displot(df,x="fare",hue="pclass",multiple="stack",binwidth=40)

#here we have plotted the graph of sex vs the survived and discovered that there are more female survivors than male survivors

[ ]
sns.displot(df,x="sex",hue="survived",multiple="stack",binwidth=40)

[ ]
df.dtypes

#Male travellers which were travelling alone are more than female travelleres which were travellng alone

[ ]
sns.displot(df,x="sex",hue="alone",multiple="stack",binwidth=40)

#Most number of people who unable to survive were from third class

[ ]
sns.displot(df,x="class",hue="survived",multiple="stack",binwidth=40)

[ ]
sns.displot(df,x="pclass",hue="survived",multiple="stack",binwidth=40)

[ ]
sns.displot(df,x="fare",col="sex",binwidth=40)

[ ]
```
sns.barplot(x="sex",y="age",data=df)
```

[ ]
```
sns.countplot(x="sex",data=df)
```

#displayed no of survived or non-survived frome ach embark town pclass wise

[ ]
```
sns.catplot(x="embark_town",hue="survived",kind='count',col="pclass",data=df)
```

#plotted the boxplot and detected the outliers there are 3 outliers in the male column of age

[ ]
```
sns.boxplot(x="sex",y="age",data=df)
```

[ ]
```
sns.boxplot(x="sex",y="age",data=df,hue="survived")
```

[ ]
```
sns.violinplot(x="sex",y="age",data=df)
```

[ ]
```
sns.violinplot(x="sex",y="age",data=df,hue="survived",split=True)
```

[ ]
```
sns.stripplot(x="sex",y="age",data=df)
```

[ ]
```
sns.stripplot(x="sex",y="age",data=df,hue="survived",split=True)
```

[ ]
```
sns.swarmplot(x="sex",y="age",data=df)
```

[ ]
```
sns.swarmplot(x="sex",y="age",data=df,hue="survived")
```

[ ]
```
sns.boxplot(x="age",y="sex",data=df)
```

[ ]
```
sns.boxplot(x="sex",y="age",data=df,hue="survived")
```

[ ]
sns.scatterplot(data=df)

#These were some of the other plots use for visulization

///
///////////
_Assignment_9_DataViualization 3.ipynb

#Imorted the required libraries

[ ]
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as mat
#loaded the dataset which was in the seaborn module in the dataframe variable df

#There are 4 numeric columns and 1object column in the dataset

[ ]
df = sns.load_dataset("iris")
df

[ ]
df.shape

[ ]
df.describe()

[ ]
df.isnull().sum()

[ ]
df.groupby('species').size()
species

#Plotting distribution for each feature

[ ]
sns.displot(df,x="sepal_length")

[ ]
```
sns.displot(df,x="sepal_width")
```

[ ]
```
sns.displot(df,x="petal_length")
```

[ ]
```
sns.displot(df,x="petal_width")
```

[ ]
```
df.dtypes
```

[ ]
```
sns.histplot(df)
```

#creating histograms for each feature

[ ]
```
sns.histplot(df,x="sepal_length")
```

[ ]
```
sns.histplot(df,x="sepal_width")
```

[ ]
```
sns.histplot(df,x="petal_length")
```

[ ]
```
sns.histplot(df,x="petal_width")
```

[ ]
```
sns.boxplot(data=df)
```

#Creating box plot for each feature

[ ]
```
sns.boxplot(data=df,x="sepal_length")
```

[ ]
```
sns.boxplot(data=df,x="sepal_width")
```

[ ]
```
sns.boxplot(data=df,x="petal_length")
```

```
[ ]
sns.boxplot(data=df,x="petal_width")
```

#Double-click (or enter) to edit

```
[ ]
data_to_plot = [df["sepal_length"],df["sepal_width"],df["petal_length"],df["petal_width"]]
sns.set_style("whitegrid")
# Creating a figure instance
fig = mat.figure(1, figsize=(12,8))
# Creating an axes instance
ax = fig.add_subplot(111)
# Creating the boxplot
bp = ax.boxplot(data_to_plot);
```

#If we observe closely, for the box 2, interquartile distance is roughly around 0.75 hence the values lying beyond this range of (third quartile + interquartile distance) i.e. roughly around 4.05 will be considered as outliers. Similarly outliers with other boxplots can be found.

//

#Word count program

```
var map = sc.textFile("/opt/spark/bin/sample.txt").flatMap(line => line.split(" ")).map(word =>
(word,1));
var counts = map.reduceByKey(_ + _);
counts.saveAsTextFile("/opt/spark/bin/WD_fri")
```

Sort

```
var map = sc.textFile("/opt/spark/bin/sample.txt").flatMap(line => line.split(" ")).map(word =>
(word,1));
var counts = map.reduceByKey(_ + _);
val keysRdd = counts.keys
val sorted = keysRdd.sortBy(x => x, true)
sorted.collect
sorted.saveAsTextFile("/opt/spark/bin/sort_fri")
```

Log file

```
val lines = sc.textFile("/opt/spark/bin/weblog.txt") // read text file
```

```
val errorLines = lines filter { l => l.contains("200")}
val warningLines = lines filter { l => l.contains("302")}
val errorCount = errorLines.count
val warningCount = warningLines.count
errorLines.saveAsTextFile("/opt/spark/bin/error_cnt1")
warningLines.saveAsTextFile("/opt/spark/bin/warn_cnt1")
```

#Max80

```
val lines = sc.textFile("/opt/spark/bin/sample.txt")
val longLines = lines filter { l => l.length > 80}
longLines.saveAsTextFile("/opt/spark/bin/max80")
```

Write a code in JAVA for a simple WordCount application
that counts the number of occurrences of each word in a
given input set using the Hadoop Map-Reduce framework on a
local standalone set-up.

Wc_mapper.java

```
package com.javatpoint;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(LongWritable key, Text
value,OutputCollector<Text,IntWritable> output,
```

```
Reporter reporter) throws IOException{
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()){
word.set(tokenizer.nextToken());
output.collect(word, one);
}
}
}
```

Wc_reducer.java

```
package com.javatpoint;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WC_Reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterator<IntWritable>
values,OutputCollector<Text,IntWritable> output,
Reporter reporter) throws IOException {
int sum=0;
while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}
```

Wc_runner.java

```
package com.javatpoint;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
```

```java
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
public static void main(String[] args) throws IOException{
JobConf conf = new JobConf(WC_Runner.class);

conf.setJobName("WordCount");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(WC_Mapper.class);
conf.setCombinerClass(WC_Reducer.class);
conf.setReducerClass(WC_Reducer.class);
conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(conf,new Path(args[0]));
FileOutputFormat.setOutputPath(conf,new Path(args[1]));
JobClient.runJob(conf);
}
}

////////////////////////////////////////////////////////////////////////////////
```