Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

LABORATORY RECORD

Academic Year 2022-2023

Course Code

20CS2031

Course Name

Introduction to Data Science Lab

Register No. URK20CS2001

It is hereby certified that this is the bonafide record of work done by Mr./Ms. **RUBAN GINO SINGH A** during the odd semester of the academic year 2022-2023 and submitted for the University Practical Examination held on ____________________.

Faculty-in-charge **Program Coordinator**

Examiner

TABLE OF CONTENTS

Ex. No. 1	**Working with Python Data Structures**
Date of Exercise	

Aim:

To work with python data types of lists, tuples, dictionary, and sets.

QUESTION -1:

Create an empty dictionary and fill with some book_id and book_name as pair by user input. Then take one book_id as input from the user and traverse through dictionary to find the corresponding book_name and display the same.

ALGORITHM:

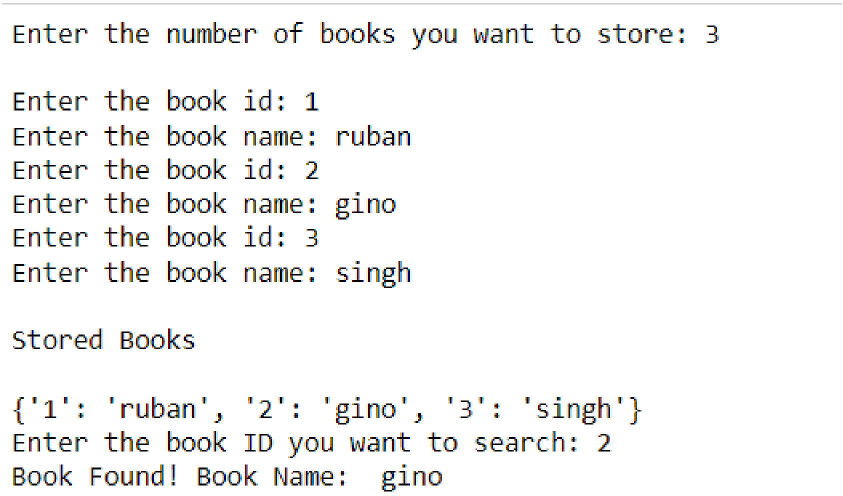
- Start the program.
- Create a Empty dictionary to store the books.
- Get the number of books as an input from the user.
- Use for loop to iterate through the book id and book name.
- Get the book id from the user to search the books.
- End the program.

PROGRAM:

```
"""1) Create an empty dictionary and fill with some book_id and book_name as pair by user
input.

Then take one book_id as input from the user and traverse through dictionary to find the
corresponding

book_name and display the same."""
```

```
books = {}


n = int(input("Enter the number of books you want to store: "))
print()


for i in range(n):
    book_id = input("Enter the book id: ")
    book_name = input("Enter the book name: ")


    books[book_id.title()] = book_name


print("\nStored Books\n")
print(books)


book_to_search = input("Enter the book ID you want to search: ")


if book_to_search in books:
    print("Book Found! Book Name: ", books[book_to_search])


else:
    print("Book NOT Found!!!")
```

OUTPUT:



```
Enter the number of books you want to store: 3

Enter the book id: 1
Enter the book name: ruban
Enter the book id: 2
Enter the book name: gino
Enter the book id: 3
Enter the book name: singh

Stored Books

{'1': 'ruban', '2': 'gino', '3': 'singh'}
Enter the book ID you want to search: 2
Book Found! Book Name:  gino
```

QUESTION -2:

Create an empty list and fill with list of strings by user input. Find the number of strings where the string length is 2 or more and the first and the last character are same.

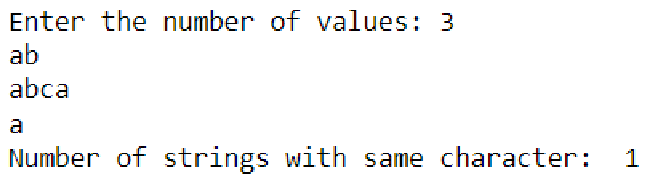
ALGORITHM:

- Start the program.
- Get the number of values from the user.
- Create a empty list to store the values.
- Use for loop to append the input to the empty created list.
- Print the results.
- End the program.

PROGRAM:

```
n = int(input("Enter the number of values: "))


list = []


for i in range(n):
    list.append(input())


count = 0
for i in list:
    if len(i)>2 and i[0]==i[-4]:
        count += 1


print("Number of strings with same character: ", count)
```

OUTPUT:



```
Enter the number of values: 3
ab
abca
a
Number of strings with same character:  1
```

QUESTION -3:

Create an empty set and fill with some values by user input. Find the maximum value in a set.

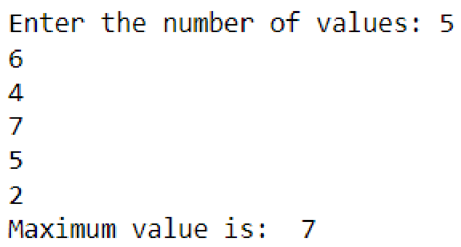
ALGORITHM:

- Start the program.
- Create a set to store the input values.
- Get the number of values as an input from the user.
- Use for loop to iterate throughout the input.
- Print the results.
- End the program.

PROGRAM:

```
set_to_store = set()


num = int(input("Enter the number of values: "))


for i in range(num):
    set_to_store.add(int(input()))
print("Maximum value is: ", max(set_to_store))
```

OUTPUT:



```
Enter the number of values: 5
6
4
7
5
2
Maximum value is:  7
```

QUESTION -4:

Create an empty tuple and fill with some values by user input. Find the sum of tuple elements.

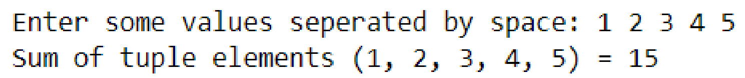
ALGORITHM:

- Start the program.
- Get some separated values from the user.
- Initialize the sum as zero.
- Use for loop to iterate throught the element.
- Print the results.
- End the program.

PROGRAM:

```
user_input = tuple(map(int, input( 'Enter some values seperated by space: ').split()))

sum = 0


for element in user_input:
   sum = sum + element


print("Sum of tuple elements", user_input, "=", sum)
```

OUTPUT:



```
Enter some values seperated by space: 1 2 3 4 5
Sum of tuple elements (1, 2, 3, 4, 5) = 15
```

QUESTION -5:

Create a 2D array and perform matrix addition using numpy.

ALGORITHM:

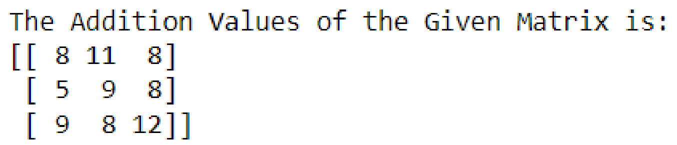
- Start the program.
- Import the NumPy library
- Create a first value as an array.
- Create a second value as an array.
- Create an addition variable and do the addition.
- Print the results.
- End the program.

PROGRAM:

```
import numpy as np


first_value = np.array([[1,2,3],[2,3,4],[3,4,5]])
```

```
second_value = np.array([[7,9,5],[3,6,4],[6,4,7]])


addition = first_value + second_value


print("The Addition Values of the Given Matrix is: ")

print(addition)
```

OUTPUT:



```
The Addition Values of the Given Matrix is:
[[ 8 11  8]
 [ 5  9  8]
 [ 9  8 12]]
```

QUESTION -6:

Read an .csv file and display the basic details.

ALGORITHM:

- Start the program.
- Import the panda’s library.
- Create a variable and read the dataset.
- Print the dataset.
- End the program.

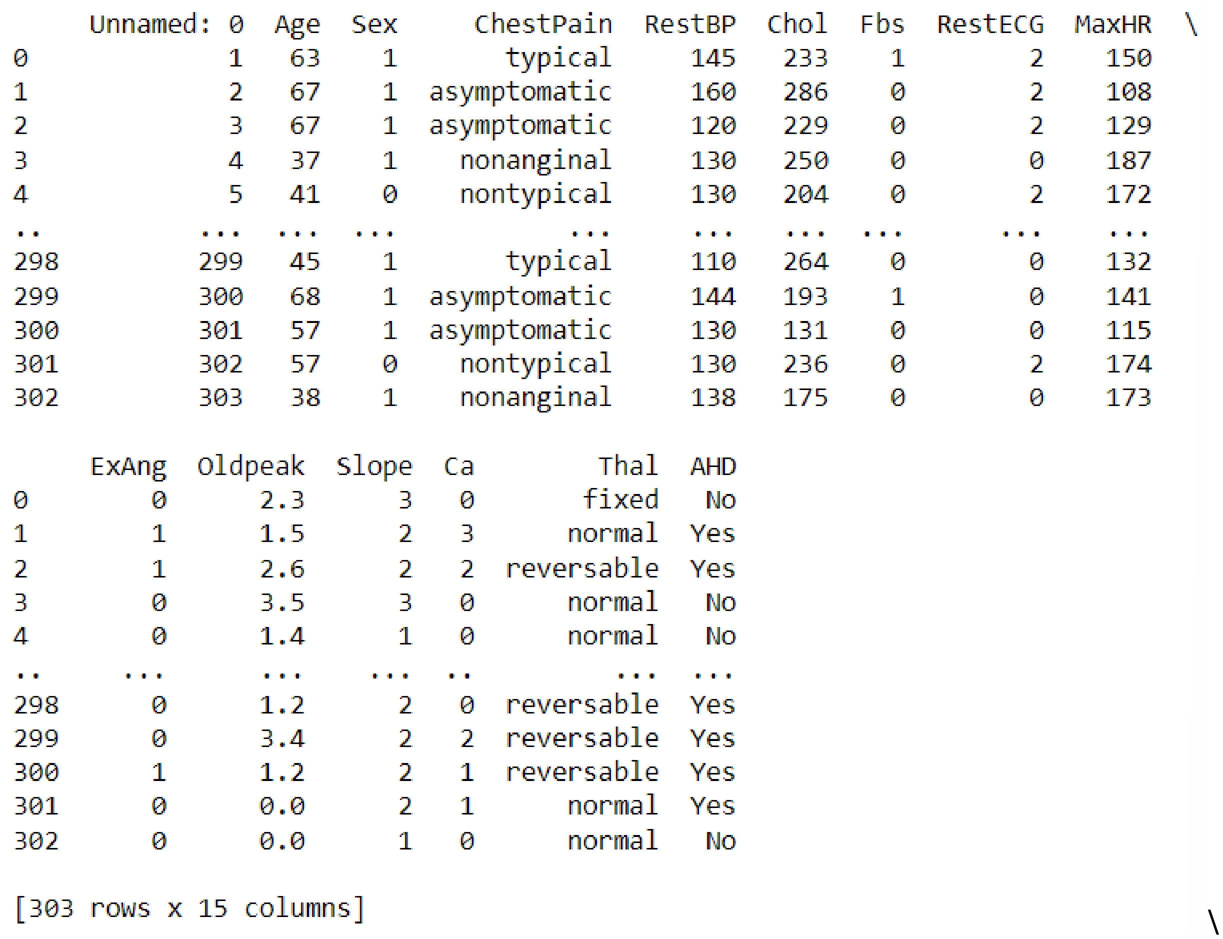
PROGRAM:

```
import pandas as pd
```

data_frame = pd.read_csv('heart.csv')

print(data_frame)

OUTPUT:



```
     Unnamed: 0  Age  Sex     ChestPain  RestBP  Chol  Fbs  RestECG  MaxHR  \
0             1   63    1       typical     145   233    1        2    150
1             2   67    1  asymptomatic     160   286    0        2    108
2             3   67    1  asymptomatic     120   229    0        2    129
3             4   37    1    nonanginal     130   250    0        0    187
4             5   41    0    nontypical     130   204    0        2    172
..          ...  ...  ...           ...     ...   ...  ...      ...    ...
298         299   45    1       typical     110   264    0        0    132
299         300   68    1  asymptomatic     144   193    1        0    141
300         301   57    1  asymptomatic     130   131    0        0    115
301         302   57    0    nontypical     130   236    0        2    174
302         303   38    1    nonanginal     138   175    0        0    173

     ExAng  Oldpeak  Slope  Ca        Thal  AHD
0        0      2.3      3   0       fixed   No
1        1      1.5      2   3      normal  Yes
2        1      2.6      2   2  reversable  Yes
3        0      3.5      3   0      normal   No
4        0      1.4      1   0      normal   No
..     ...      ...    ...  ..         ...  ...
298      0      1.2      2   0  reversable  Yes
299      0      3.4      2   2  reversable  Yes
300      1      1.2      2   1  reversable  Yes
301      0      0.0      2   1      normal  Yes
302      0      0.0      1   0      normal   No

[303 rows x 15 columns]
```

\

Result:

The program to execute the above programs are compiled and output is verified successfully.

Ex. No.2	Working with dataset using pandas
27.07.22	

Aim:

To work with dataset using pandas.

Dataset: candy.csv

AIM:

To work with dataset using Pandas.

QUESTION -1:

Import the dataset

ALGORITHM:

- Start the program.
- Run the code to import the dataset.
- End the program.

DESCRIPTION:

read_csv is used to load a CSV file as a pandas dataframe.

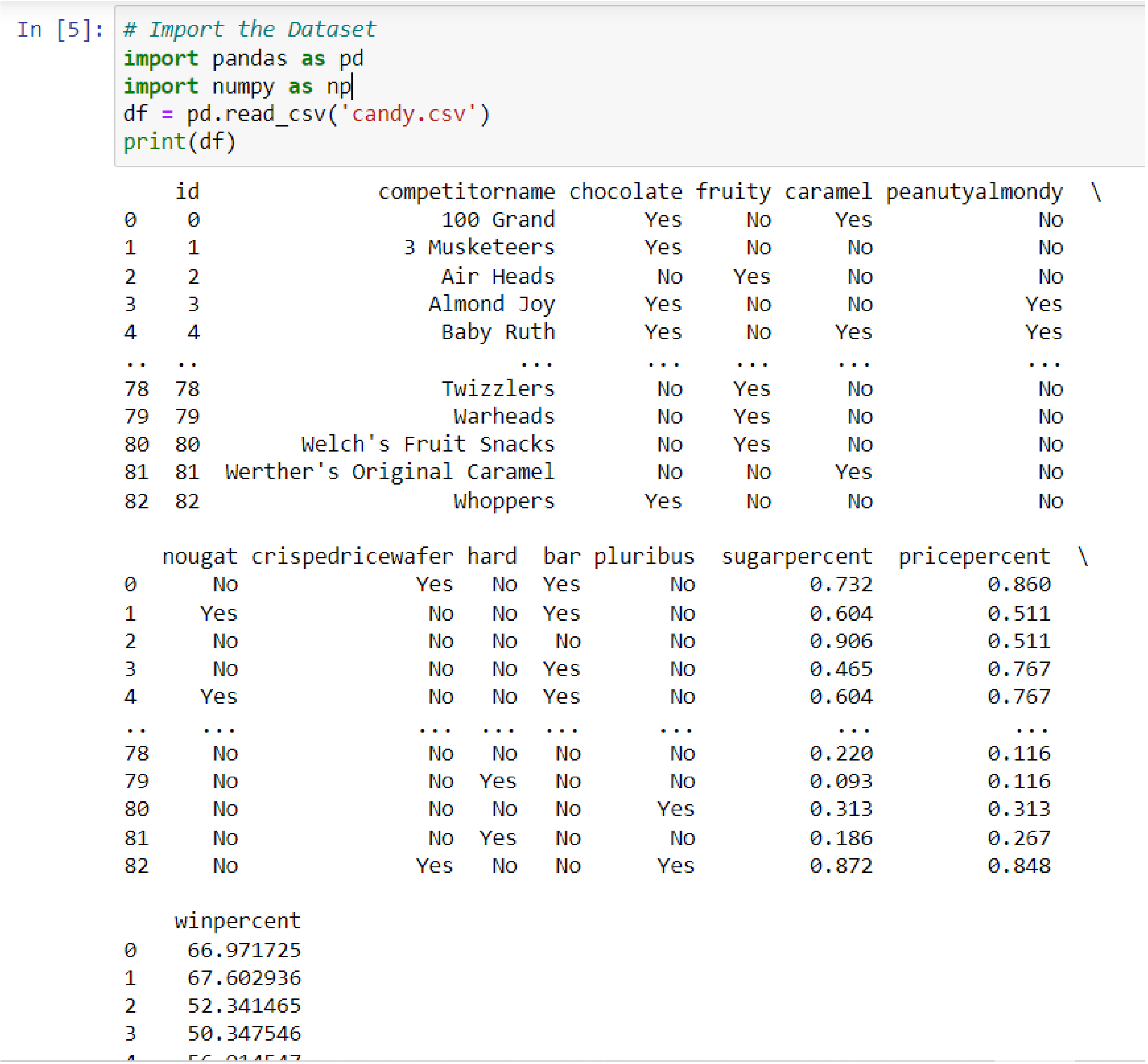
PROGRAM:

Import the Dataset

data = pd.read_csv('candy.csv')

data

OUTPUT:



```
In [5]: # Import the Dataset
        import pandas as pd
        import numpy as np
        df = pd.read_csv('candy.csv')
        print(df)
```

```
    id             competitorname chocolate fruity caramel peanutyalmondy  \
0    0                  100 Grand       Yes     No     Yes             No
1    1               3 Musketeers       Yes     No      No             No
2    2                  Air Heads        No    Yes      No             No
3    3                 Almond Joy       Yes     No      No            Yes
4    4                  Baby Ruth       Yes     No     Yes            Yes
..  ..                        ...       ...    ...     ...            ...
78  78                  Twizzlers        No    Yes      No             No
79  79                   Warheads        No    Yes      No             No
80  80        Welch's Fruit Snacks        No    Yes      No             No
81  81  Werther's Original Caramel        No     No     Yes             No
82  82                   Whoppers       Yes     No      No             No

   nougat crispedricewafer hard  bar pluribus  sugarpercent  pricepercent  \
0      No              Yes   No  Yes       No         0.732         0.860
1     Yes               No   No  Yes       No         0.604         0.511
2      No               No   No   No       No         0.906         0.511
3      No               No   No  Yes       No         0.465         0.767
4     Yes               No   No  Yes       No         0.604         0.767
..    ...              ...  ...  ...      ...           ...           ...
78     No               No   No   No       No         0.220         0.116
79     No               No  Yes   No       No         0.093         0.116
80     No               No   No   No      Yes         0.313         0.313
81     No               No  Yes   No       No         0.186         0.267
82     No              Yes   No   No      Yes         0.872         0.848

    winpercent
0    66.971725
1    67.602936
2    52.341465
3    50.347546
4    56.914547
```

QUESTION -2:

Display the head and tail of the dataset.

ALGORITHM:

- Start the program.
- Run the code to display the head and tail of the dataset.
- End the program.

DESCRIPTION:

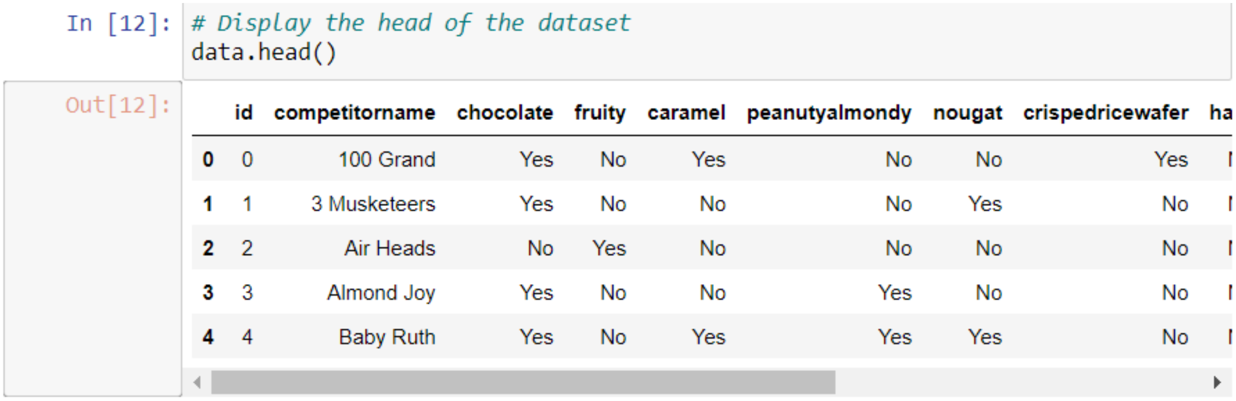
The *head()* function is used to get the first n rows.

The tail() function is used to return the last n rows.

PROGRAM:

- df.head()
- df.tail()

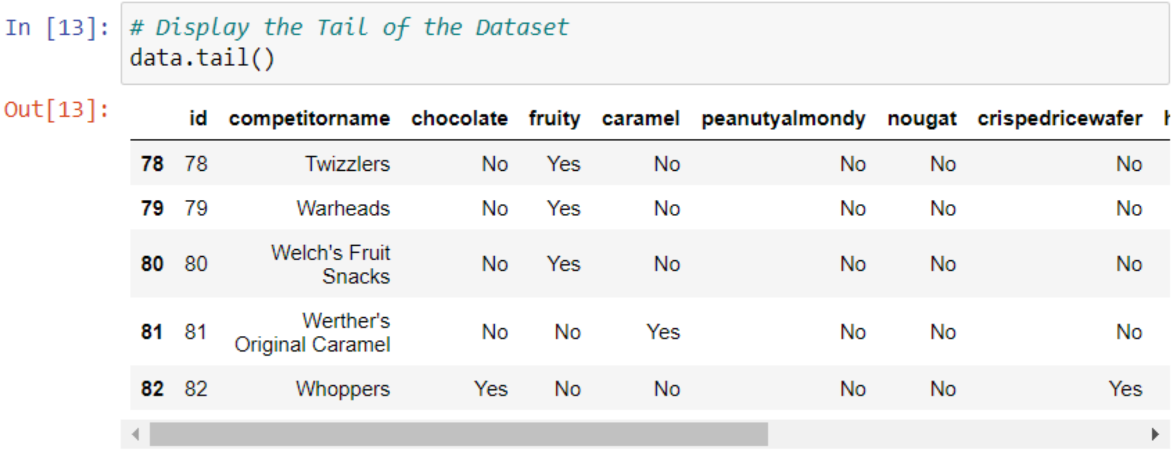
OUTPUT:



In [12]:
```
# Display the head of the dataset
data.head()
```

Out[12]:

	id	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	ha
0	0	100 Grand	Yes	No	Yes	No	No	Yes	I
1	1	3 Musketeers	Yes	No	No	No	Yes	No	I
2	2	Air Heads	No	Yes	No	No	No	No	I
3	3	Almond Joy	Yes	No	No	Yes	No	No	I
4	4	Baby Ruth	Yes	No	Yes	Yes	Yes	No	I



In [13]:
```
# Display the Tail of the Dataset
data.tail()
```

Out[13]:

	id	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	I
78	78	Twizzlers	No	Yes	No	No	No	No	
79	79	Warheads	No	Yes	No	No	No	No	
80	80	Welch's Fruit Snacks	No	Yes	No	No	No	No	
81	81	Werther's Original Caramel	No	No	Yes	No	No	No	
82	82	Whoppers	Yes	No	No	No	No	Yes	

QUESTION -3:

Display column names and datatypes of the columns.

ALGORITHM:

- Start the program.
- Run the code to display column names and datatypes
- End the program.

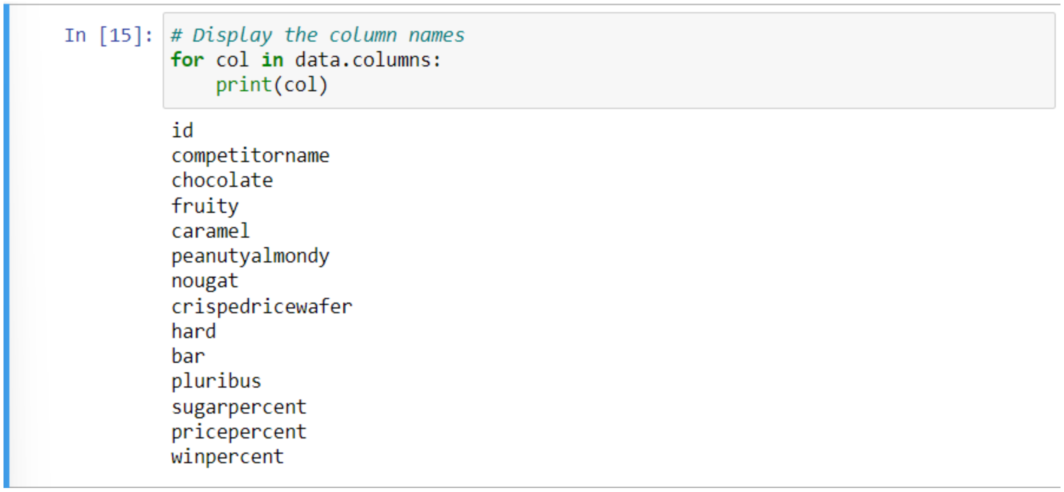
DESCRIPTION:

- **DataFrame.columns** attribute return the column labels of the given dataframe.
- DataFrame.dtypes attribute returns a series with the data type of each column.

PROGRAM:

```
# Display the column names
for col in data.columns:
    print(col)
```

OUTPUT:



```
# Display the column names
for col in data.columns:
    print(col)
```

```
id
competitorname
chocolate
fruity
caramel
peanutyalmondy
nougat
crispedricewafer
hard
bar
pluribus
sugarpercent
pricepercent
winpercent
```

QUESTION -4:

Display statistical information about suitable columns.

ALGORITHM:

- Start the program.
- Run the code to display statistical information about suitable columns.
- End the program.

DESCRIPTION:

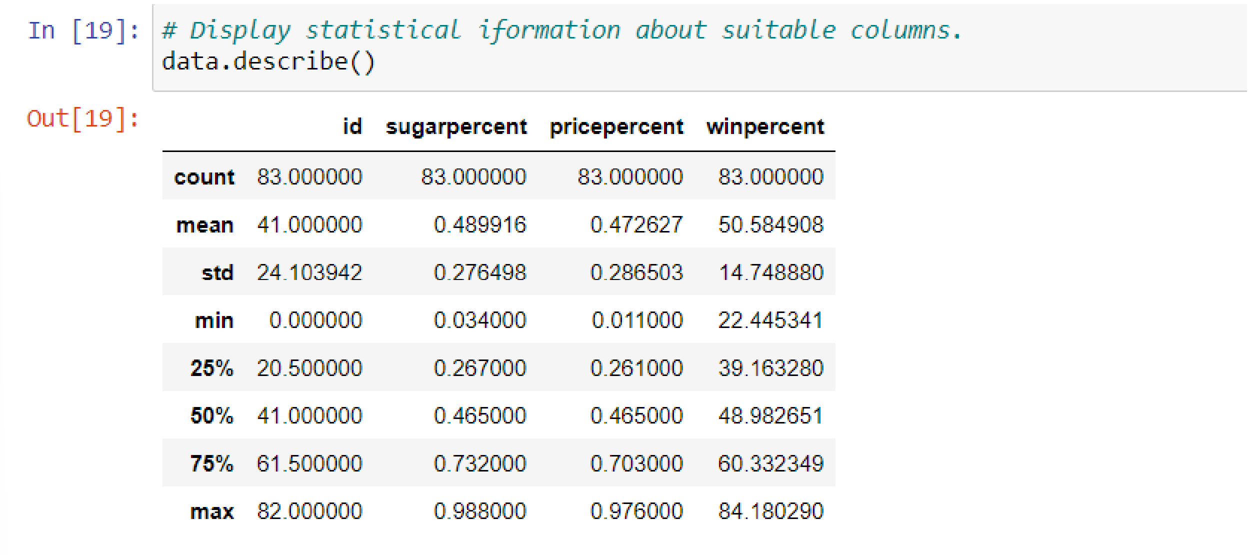
The describe() method returns description of the data in the DataFrame.

PROGRAM:

Display statistical iformation about suitable columns.

data.describe()

OUTPUT:



```
In [19]: # Display statistical iformation about suitable columns.
         data.describe()
```

Out[19]:

	id	sugarpercent	pricepercent	winpercent
count	83.000000	83.000000	83.000000	83.000000
mean	41.000000	0.489916	0.472627	50.584908
std	24.103942	0.276498	0.286503	14.748880
min	0.000000	0.034000	0.011000	22.445341
25%	20.500000	0.267000	0.261000	39.163280
50%	41.000000	0.465000	0.465000	48.982651
75%	61.500000	0.732000	0.703000	60.332349
max	82.000000	0.988000	0.976000	84.180290

QUESTION -5:

Display all the rows of the column chocolate, caramel and fruity.

ALGORITHM:

- Start the program.
- Display all the rows of chocolate, caramel and fruity columns.
- End the program.

DESCRIPTION:

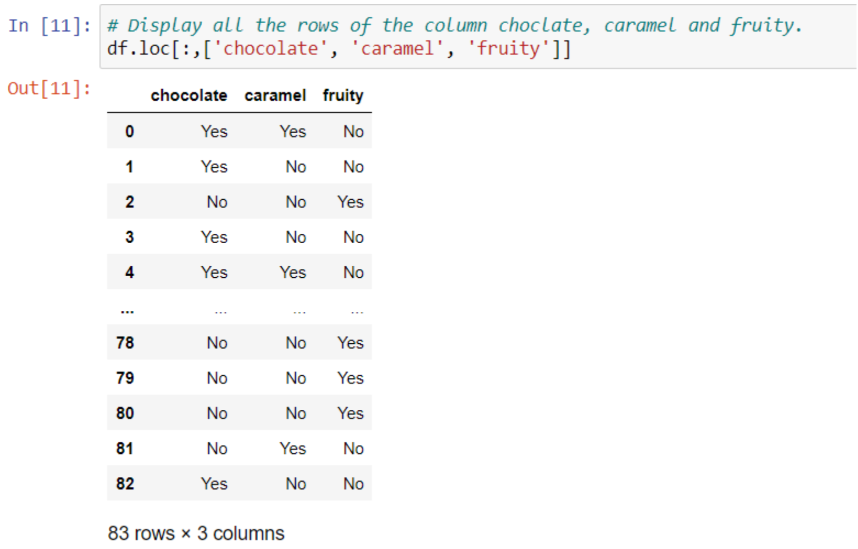
Pandas DataFrame.*loc* attribute access a group of rows and columns by label(s) or a boolean array in the given DataFrame.

PROGRAM:

Display all the rows of columns Chocolate, Caramel, Fruity

data.loc[:,['chocolate', 'caramel', 'fruity']]

OUTPUT:



```
In [11]: # Display all the rows of the column choclate, caramel and fruity.
         df.loc[:,['chocolate', 'caramel', 'fruity']]
```

Out[11]:

	chocolate	caramel	fruity
0	Yes	Yes	No
1	Yes	No	No
2	No	No	Yes
3	Yes	No	No
4	Yes	Yes	No
...	...	...	...
78	No	No	Yes
79	No	No	Yes
80	No	No	Yes
81	No	Yes	No
82	Yes	No	No

83 rows × 3 columns

QUESTION -6:

Display the total number of Competitors.

ALGORITHM:

- Start the program.
- Display total number of competitors.
- End the program.

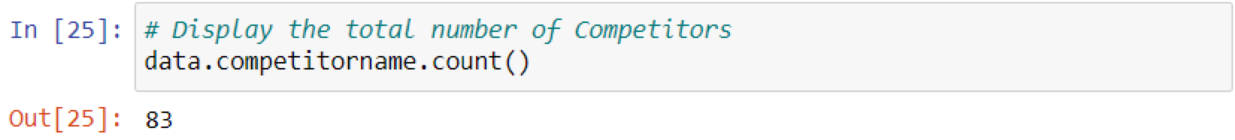
DESCRIPTION:

The count() method counts the number of not empty values for each row, or column if you specify the axis parameter as axis='columns' , and returns a Series object with the result for each row (or column).

PROGRAM:

```
# Display the total number of Competitors
data.competitorname.count()
```

OUTPUT:



```
In [25]: # Display the total number of Competitors
         data.competitorname.count()

Out[25]: 83
```

QUESTION -7:

Display by slicing the dataset using iloc and loc commands.

ALGORITHM:

- Start the program.
- Display by slicing the dataset using iloc and loc commands.
- End the program.

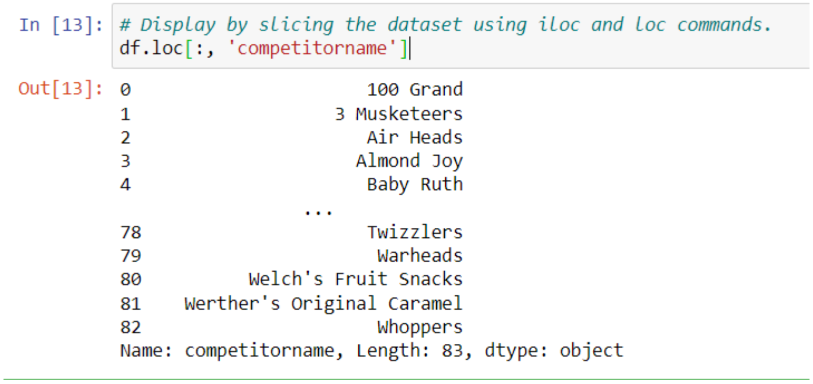
DESCRIPTION:

DataFrame.*loc* attribute access a group of rows and columns by label(s) or a boolean array in the given DataFrame.

PROGRAM:

```
# Display by slicing the dataset using iloc and loc commands.
data.loc[:, 'competitorname']
```

OUTPUT:



```
In [13]: # Display by slicing the dataset using iloc and loc commands.
         df.loc[:, 'competitorname']

Out[13]: 0                      100 Grand
         1                   3 Musketeers
         2                      Air Heads
         3                     Almond Joy
         4                      Baby Ruth
                        ...
         78                     Twizzlers
         79                      Warheads
         80           Welch's Fruit Snacks
         81     Werther's Original Caramel
         82                      Whoppers
         Name: competitorname, Length: 83, dtype: object
```

QUESTION -8:

Check the dataset for any null value and fill the null value with 0.01

ALGORITHM:

- Start the program.
- Check the dataset for any null value and fill the null value with 0.01
- End the program.

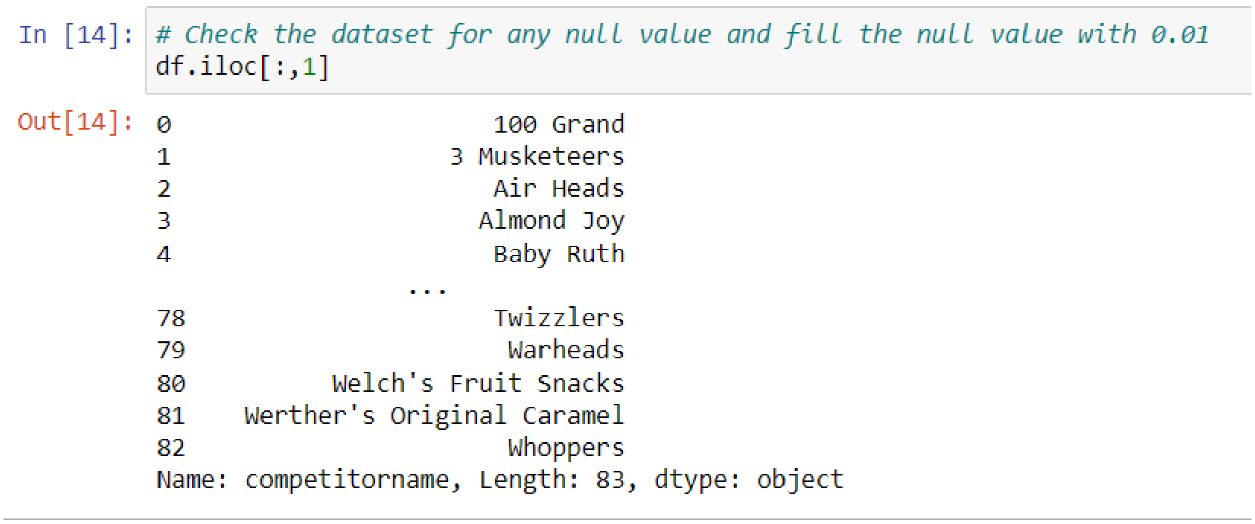
DESCRIPTION:

Dataframe.iloc[] method is used when the index label of a data frame is something other than numeric series of 0, 1, 2, 3….n or in case the user doesn’t know the index label. Rows can be extracted using an imaginary index position which isn’t visible in the data frame.

PROGRAM:

df.iloc[:,1]

OUTPUT:



```
In [14]: # Check the dataset for any null value and fill the null value with 0.01
         df.iloc[:,1]
```

```
Out[14]: 0                       100 Grand
         1                    3 Musketeers
         2                       Air Heads
         3                      Almond Joy
         4                       Baby Ruth
                          ...
         78                      Twizzlers
         79                       Warheads
         80           Welch's Fruit Snacks
         81     Werther's Original Caramel
         82                       Whoppers
         Name: competitorname, Length: 83, dtype: object
```

QUESTION -9:

Find the mean winpercent

ALGORITHM:

- Start the program.
- Find the mean of winpercent column.
- End the program.

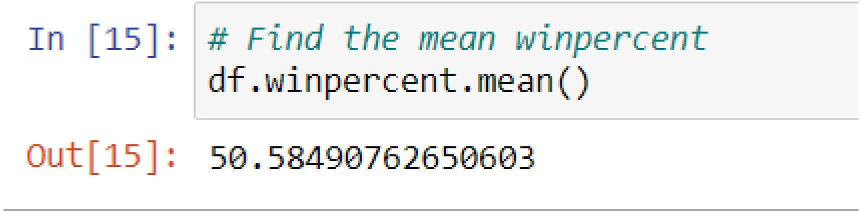
DESCRIPTION:

Pandas dataframe.*mean()* function return the mean of the values for the requested axis.

PROGRAM:

```
df.winpercent.mean()
```

OUTPUT:



```
In [15]: # Find the mean winpercent
         df.winpercent.mean()

Out[15]: 50.58490762650603
```

QUESTION -10:

Display how many competitors are both hard and bar.

ALGORITHM:

- Start the program.
- Display how many competitors are both hard and bar.
- End the program.

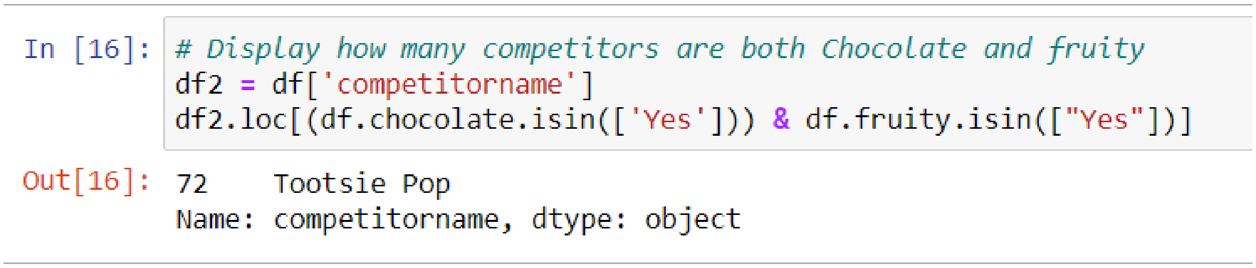
DESCRIPTION:

- DataFrame.*loc* attribute access a group of rows and columns by label(s) or a boolean array in the given DataFrame.
- The isin() method checks if the Dataframe contains the specified value(s).

PROGRAM:

```
df2 = df['competitorname']
df2.loc[(df.chocolate.isin(['Yes'])) & df.fruity.isin(["Yes"])]
```

OUTPUT:



```
In [16]: # Display how many competitors are both Chocolate and fruity
         df2 = df['competitorname']
         df2.loc[(df.chocolate.isin(['Yes'])) & df.fruity.isin(["Yes"])]

Out[16]: 72    Tootsie Pop
         Name: competitorname, dtype: object
```

QUESTION -11:

Display how many competitors are both hard and bar.

ALGORITHM:

· Start the program.

· Display how many competitors are both hard and bar.

· End the program.

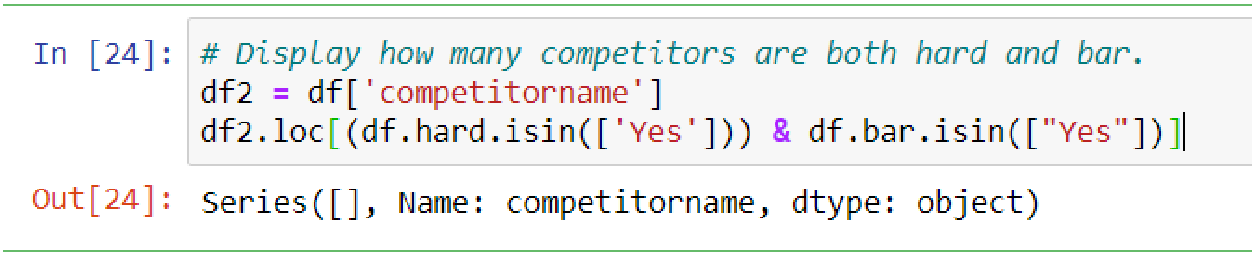
DESCRIPTION:

- DataFrame.*loc* attribute access a group of rows and columns by label(s) or a boolean array in the given DataFrame.
- The isin() method checks if the Dataframe contains the specified value(s).

PROGRAM:

```
df2 = df['competitorname']
df2.loc[(df.hard.isin(['Yes'])) & df.bar.isin(["Yes"])]
```

OUTPUT:



```
In [24]: # Display how many competitors are both hard and bar.
         df2 = df['competitorname']
         df2.loc[(df.hard.isin(['Yes'])) & df.bar.isin(["Yes"])]

Out[24]: Series([], Name: competitorname, dtype: object)
```

QUESTION -12:

Display which competitor has the higher win percent.

ALGORITHM:

- Start the program.
- Display which competitor has the higher win percent.
- End the program.

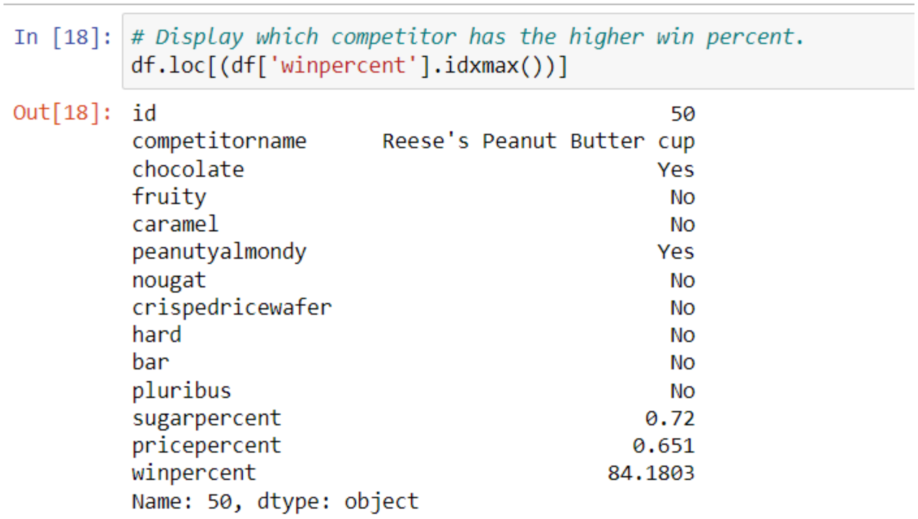
DESCRIPTION:

The idxmax() method returns a Series with the index of the maximum value for each column. By specifying the column axis (axis='columns'), the idxmax() method returns a Series with the index of the maximum value for each row.

PROGRAM:

df.loc[(df['winpercent'].idxmax())]

OUTPUT:



```
In [18]: # Display which competitor has the higher win percent.
         df.loc[(df['winpercent'].idxmax())]

Out[18]: id                                       50
         competitorname      Reese's Peanut Butter cup
         chocolate                               Yes
         fruity                                   No
         caramel                                  No
         peanutyalmondy                          Yes
         nougat                                   No
         crispedricewafer                         No
         hard                                     No
         bar                                      No
         pluribus                                 No
         sugarpercent                           0.72
         pricepercent                          0.651
         winpercent                          84.1803
         Name: 50, dtype: object
```

QUESTION -13:

Sort the Competitors by winpercent

ALGORITHM:

- Start the program.
- Sort the Competitors by winpercent
- End the program.

DESCRIPTION:

Pandas sort_values() function sorts a data frame in Ascending or Descending order of passed Column.

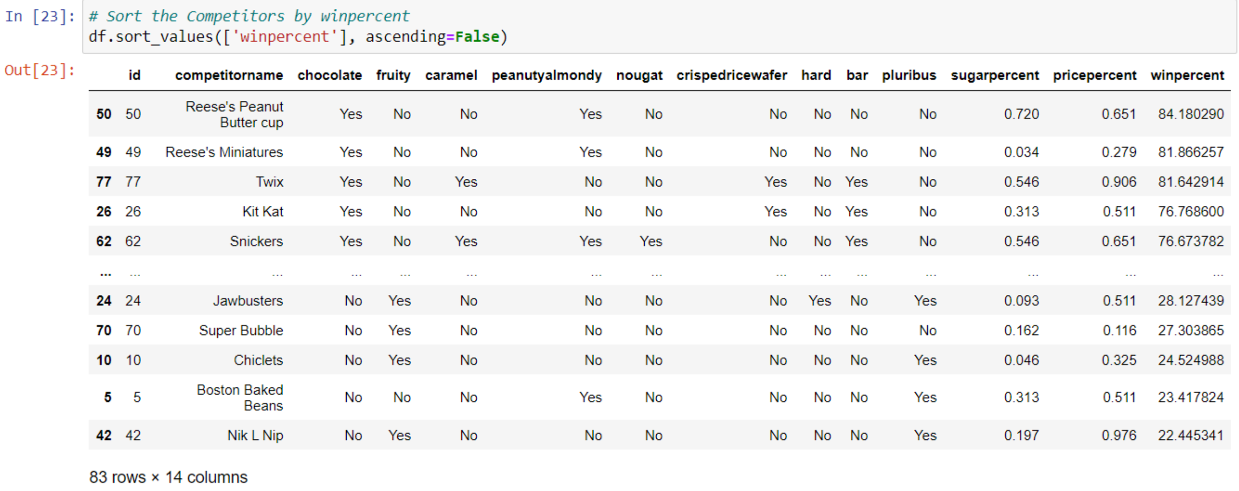
Syntax:

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na_position=’last’)

PROGRAM:

df.sort_values(['winpercent'], ascending=False)

OUTPUT:



```
In [23]: # Sort the Competitors by winpercent
         df.sort_values(['winpercent'], ascending=False)
```

Out[23]:

	id	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard	bar	pluribus	sugarpercent	pricepercent	winpercent
50	50	Reese's Peanut Butter cup	Yes	No	No	Yes	No	No	No	No	No	0.720	0.651	84.180290
49	49	Reese's Miniatures	Yes	No	No	Yes	No	No	No	No	No	0.034	0.279	81.866257
77	77	Twix	Yes	No	Yes	No	No	Yes	No	Yes	No	0.546	0.906	81.642914
26	26	Kit Kat	Yes	No	No	No	No	Yes	No	Yes	No	0.313	0.511	76.768600
62	62	Snickers	Yes	No	Yes	Yes	Yes	No	No	Yes	No	0.546	0.651	76.673782
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
24	24	Jawbusters	No	Yes	No	No	No	No	Yes	No	Yes	0.093	0.511	28.127439
70	70	Super Bubble	No	Yes	No	No	No	No	No	No	No	0.162	0.116	27.303865
10	10	Chiclets	No	Yes	No	No	No	No	No	No	Yes	0.046	0.325	24.524988
5	5	Boston Baked Beans	No	No	No	Yes	No	No	No	No	Yes	0.313	0.511	23.417824
42	42	Nik L Nip	No	Yes	No	No	No	No	No	No	Yes	0.197	0.976	22.445341

83 rows × 14 columns

Result:

The program to execute the above programs are compiled and output is verified successfully.

Ex. No. 3	DATA VISUALIZATION
03-08-2022	

Question-1:

Aim:

Draw a bar chart with Team and its count (use different colors for each team).

Description:

value_counts() method returns the count of unique values in the given column. plt() function displays the mentioned kind of plot in it. Bar charts are the tools for presenting the relating proportions of categorical variables. xticks() method rotates the x-axis labels according to thegiven value in it.

Algorithm:

Step-1: Create a variable called data and pass the count of unique values in the team columnin it and print it.

Step-2: Use the plt.xticks() method to rotate the labels on x-axis.

Step-3: Now, use plt.bar() method to plot the bar chart and pass the data values, colors in it.

Code:

```
import matplotlib.pyplot as plt

data=df['Team'].value_counts()
print(data);

plt.bar(data.index,data.values,color={'green', 'orange',
'violet', 'blue', 'red', 'yellow', 'pink'});
```

Sample Input & Output:

Result:

The code is executed and expected output is printed on the screen.

Question-2:

Aim:

Draw a comparative bar chart for Salary and New_Salary against each person (first 15 persons).

Description:

NumPy is a Python library used for working with arrays. np.arange() method returns evenly spaced values within a given interval. Comparative bar charts are used to identify the minimum and maximum values in a series, and whether a trend is increasing or decreasing.

Algorithm:

Step-1: Import the numpy array o work with the arrays. Take the first 15 rows and use

np.arange() method that returns evenly spaced values within a given interval and save it in the ind variable.

Step-2: Give the width of the bars. Pass the x-axis values of the salary column of the first 15 persons in the xvals method.

Step-3: In bar1 variable pass the ind, width, xvals and color of the bar.

Step-4: Pass the y-axis values of the new_salary column of the first 15 persons in the y vals method.

Step-5: In bar2 variable pass the ind, width, yvals and color of the bar.

Step-6: plt.xlabel(), plt.ylabel() and plt.title() methods are used to display the names of x-axis,y-axis and the title of the bar chart.

Step-7: Pass ind+width(returns the spacing between grouped bar plot), persons names on x-axis labels and rotate the persons on x-axis to plt.xticks method().

Step-8: plt.legend() function is used to place a legend on the axes.

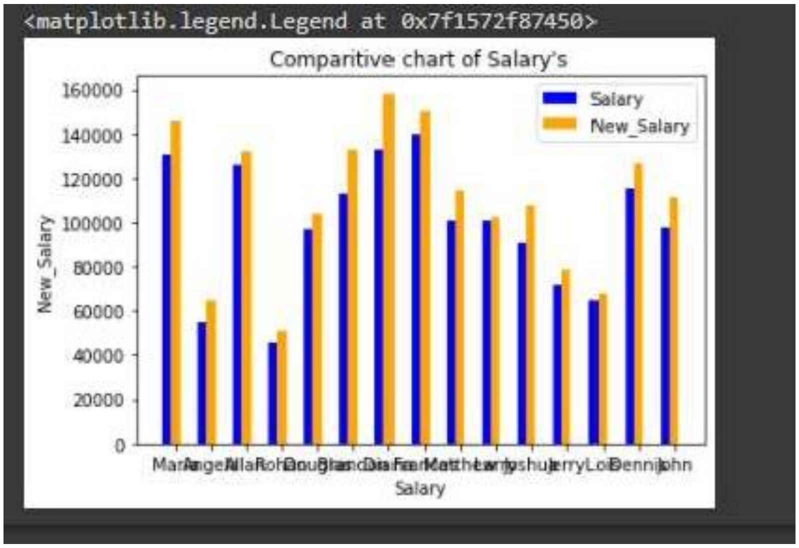
Code:

```
import numpy as np
N = 15

ind = np.arange(N)
width = 0.25


xvals = df['Salary'].head(15)

bar1 = plt.bar(ind, xvals, width, color = 'blue')


yvals = df['New_Salary'].head(15)

bar2 = plt.bar(ind+width, yvals, width, color='orange')


plt.xlabel("Salary")
plt.ylabel('New_Salary')
plt.title("Comparitive chart of
Salary's")


plt.xticks(ind+width,df['First  Name'].head(15))

plt.legend( (bar1,
```

```
bar2),
```

Sample Input & Output:

```
('Salary','New_Salary')
)
```



```
<matplotlib.legend.Legend at 0x7f1572f87450>
```

Comparitive chart of Salary's
Salary
New_Salary
New_Salary
160000
140000
120000
100000
80000
60000
40000
20000
0
MariAngelAllaRohDougBlanDiaFraMattheLarryJoshuJerryLoiDennisJohn
Salary

Result:

The code is executed and expected output is printed on the screen.

Question-3:

Aim:

Draw a horizontal bar chart for Team and Salary.

Description:

Bar charts are the tools for presenting the relating proportions of categorical variables. plt.barh() method displays the horizontal bar chart of the passed columns in it.

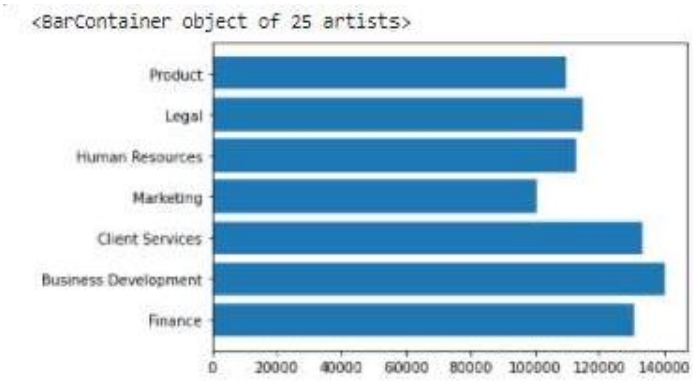
Algorithm:

Step-1: Use plt.barh() method to display a horizontal bar chart. Step-2: Now, pass the team and salary columns of the dataframe to it.

Code:

```
plt.barh(df['Team'],df['Salary'])
```

Sample Input & Output:



```
<BarContainer object of 25 artists>
```

Result:

The code is executed and expected output is printed on the screen.

Question-4:

Aim:

Draw a stacked bar chart for Salary and New_Salary against the person (first 10 persons).

Description:

Partitioning each bar into pieces yields the stacked bar chart. head() method returns the mentioned number of top rows. xticks() method rotates the x-axis labels according to the given value in it.

Algorithm:

Step-1: Use plt.bar() function to display the bar charts of the salary and new_salary columnsof the first 10 persons.

Step-2: Give the person's name on the x-axis labels of the first 10 persons. Step-3: Use head() method to display the first 10 persons.

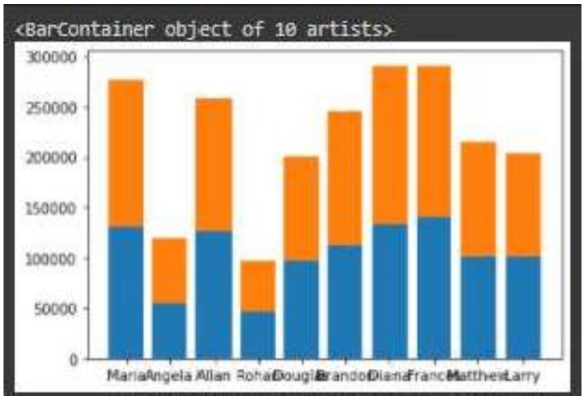
Step-4: Give the column name salary to the bottom function to display the salary columns values on the bottom of the new_salary column.

Step-5: Use the plt.xticks() method to rotate the labels on the x-axis.

Code:

```
plt.bar(df['First Name'].head(10),df['Salary'].head(10))
plt.bar(df['First
Name'].head(10),df['New_Salary'].head(10),bottom=df['
Salary'].head(10))
```

Sample Input & Output:



```
<BarContainer object of 10 artists>
```

Result:

The code is executed and expected output is printed on the screen.

Question-5:

Aim:

Draw a pie chart with Gender and its count.

Description:

value_counts() method returns the count of unique values in the given column. Pie charts are the tools for presenting the relating proportions of categorical variables. plt.pie() function displays a pie chart.

Algorithm:

Step-1: Create a variable called data and pass the count of unique values in the gendercolumn in it and print it.

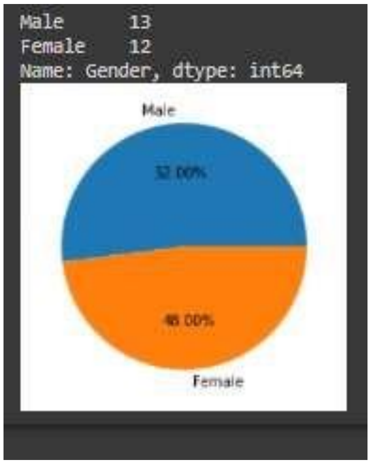
Step-2: Now, use plt.pie() method to plot the bar chart and pass the data values in it.

Code:

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

```
print(data)
plt.pie(data.values,labels=data.index,autopct='%1.2f%%');
```

Sample Input & Output:



```
Male      13
Female    12
Name: Gender, dtype: int64
```

Result:

The code is executed and expected output is printed on the screen.

Question-6:

Aim:

Draw the dot plot between person and experience (first 15 persons).

Description:

Dot plots provide the visual representation of a function(y=f(x)) defined by a set of points and they just show the data points. head() method returns the mentioned number of top rows.xticks() method rotates the x-axis labels according to the given value in it.

Algorithm:

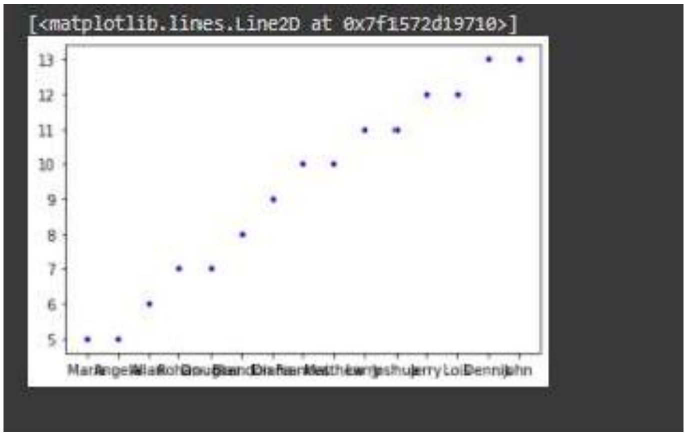
Step-1: Pass the names of columns of the person's name and their experience of the first 15 persons to the plt.plot() method.

Step-2: Give the linewidth as zero coz we are not connecting the dots. Step-3: Give the marker to be used on the plot and color of the dots. Step-4: Use the plt.xticks() method to rotate the labels on the x-axis.

Code:

```
plt.plot(df['FirstName'].head(15),df['Experience'].head(15),linewidth=0,marker='.',color='blue ”)
```

Sample Input & Output:



```
[<matplotlib.lines.Line2D at 0x7f1572d19710>]
```

Result:

The code is executed and expected output is printed on the screen.

Question-7:

Aim:

Draw the line plot between age and experience. Observe the trend line.

Description:

Line plots also provide the visual representation of a function(y=f(x)) defined by a set ofpoints and they connect the data points. head() method returns the mentioned number of toprows. xticks() method rotates the x-axis labels according to the given value in it. sort_values() function sorts the data frame in ascending or descending order of the passed column.

Algorithm:

Step-1: Use the sort_values() function to sort the values of the age column in ascending order and sign it to a variable.

Step-2: Pass the names of columns of the age and experience of the dataframe by the variableto the plt.plot() method.

Step-3: Give the linewidth as one coz we are connecting the dots with the line for line plot.Step-4: Give the marker to be used on the plot and color of the dots.

Step-5: Use the plt.xticks() method to rotate the labels on the x-axis.

Step-6: plt.xlabel(), plt.ylabel() and plt.title() methods are used to display the names of x-axis,y-axis and the title of the line plot.

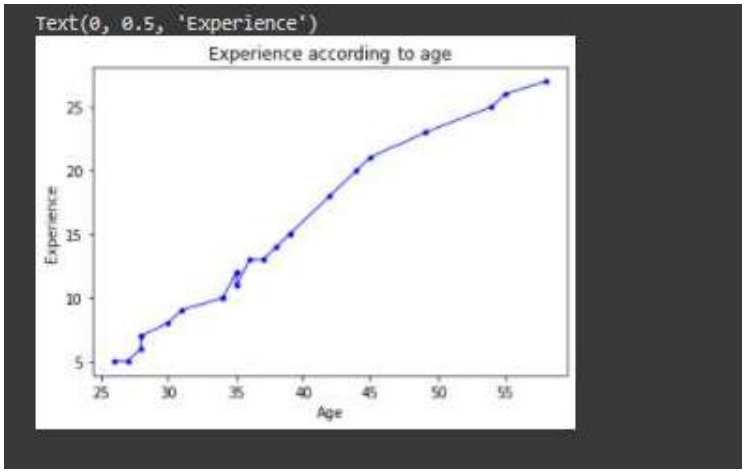
Code:

```
t=df.sort_values(by='Age',ascending=True)

plt.plot(t['Age'],t['Experience'],linewidth=1,marker='.',color='blue')

plt.title("Experience according to age")
```

plt.xlabel("Age")

plt.ylabel("Experience")

Sample Input & Output:



```
Text(0, 0.5, 'Experience')
```

Result:

The code is executed and expected output is printed on the screen.

Question-8:

Aim:

Draw the scatter plot between Salary and New_Salary. Observe the correlation.

Description:

Scatter plots are used to convey the relationship between two numerical variables and theircorrelation. plt.scatter() method returns the scatter plot. When the y variable tends to increase as the x variable increases, we say there is a positive correlation between the variables and it is known as positive correlation**.**

Algorithm:

Step-1: Assign the columns salary and new_salary to the x and y variables.

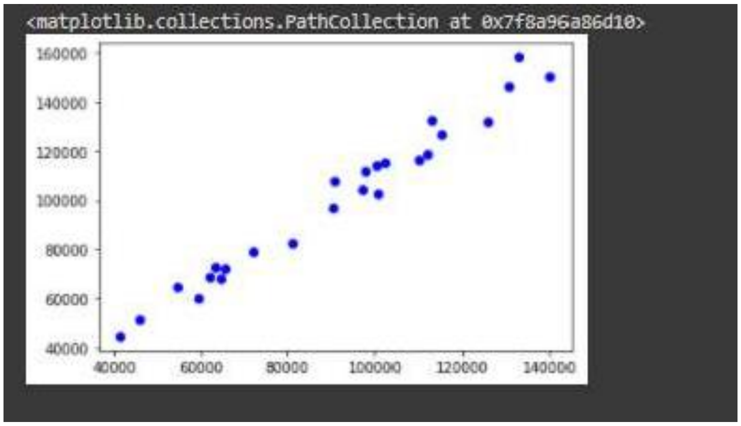
Step-2: Now, pass the x,y variables into the plt.scatter() function and give the color.

Step-3: Observe whether it is positive or negative or no correlation. Displayed scatter plot is positive correlation.

Code:

```
x=df['Salary']
y=df['New_Salary']
plt.scatter(x,y,c='blue')
```

Sample Input & Output:



```
<matplotlib.collections.PathCollection at 0x7f8a96a86d10>
```

Result:

The code is executed and expected output is printed on the screen.

Question-9:

Aim:

Draw the scatter plot between Age and Incentive. Observe the correlation.

Description:

Scatter plots are used to convey the relationship between two numerical variables and their correlation. plt.scatter() method returns the scatter plot. When the y variable tends to decrease as the x variable increases, we say there is negative correlation between the variables and it is known as negative correlation.

Algorithm:

Step-1: Assign the columns age and incentive to the x and y variables.

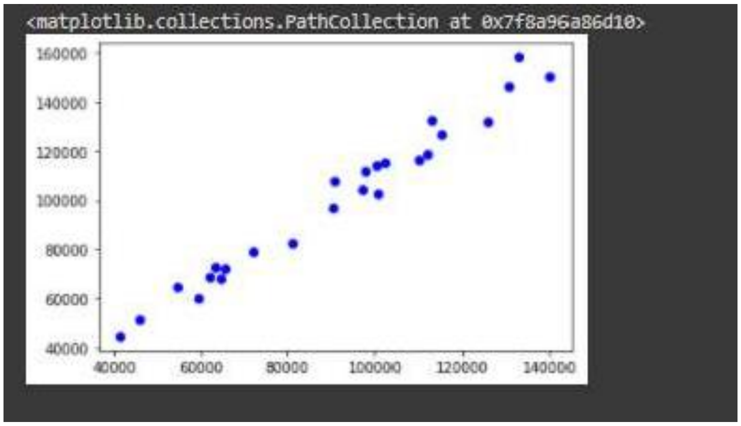
Step-2: Now, pass the x,y variables into the plt.scatter() function and give the color.

Step-3: Observe whether it is positive or negative or no correlation. Displayed scatter plot is negative correlation.

Code:

```
x=df['Age']
y=df['Incentive']
plt.scatter(x,y,c='blue')
```

Sample Input & Output:



```
<matplotlib.collections.PathCollection at 0x7f8a96a86d10>
```

Result:

The code is executed and expected output is printed on the screen.

Question-10:

Aim:

Draw the box plot to show the statistical summary of the Age column and verify with describe().

Description:

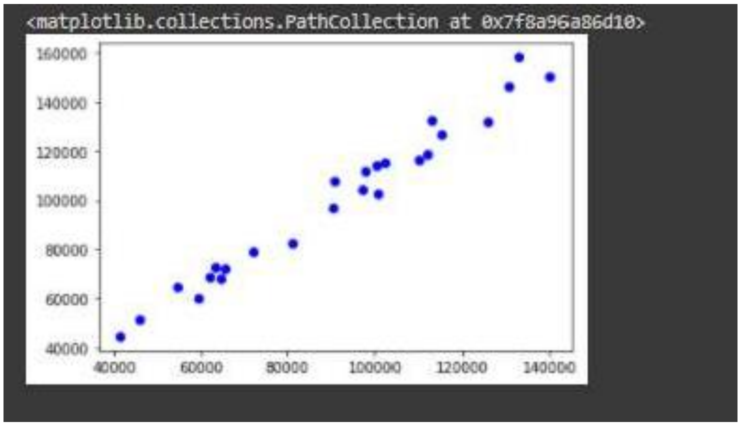
Box plot means summarizing the set of data measured on an interval scale. plt.boxplot() function displays the statistical summary of the column passes in it. describe() method returnsthe statistical summary of all the columns in the dataframe.

Algorithm:

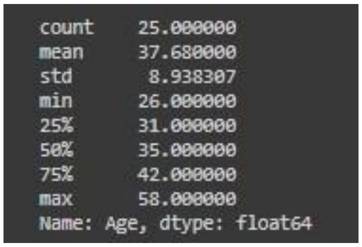
Step-1: Assign a variable to plt.boxplot() method and pass the age column of the dataframe. Step-2: Using describe() method display the statistical summary of the age column.

Code: df['Age'].describe()

Sample Input & Output:



```
<matplotlib.collections.PathCollection at 0x7f8a96a86d10>
```

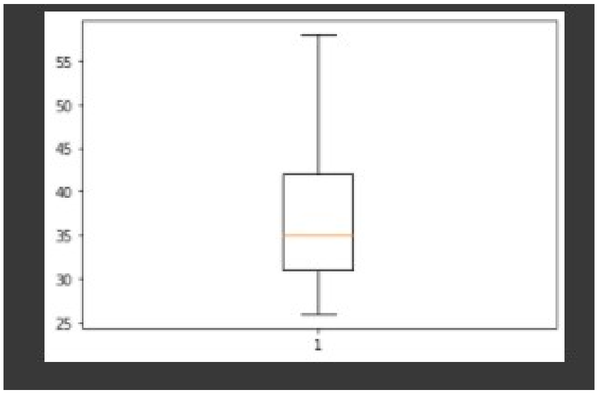


```
count    25.000000
mean     37.680000
std       8.938307
min      26.000000
25%      31.000000
50%      35.000000
75%      42.000000
max      58.000000
Name: Age, dtype: float64
```

Code:

```
t=plt.boxplot(df['Age'])
```

Output:



Result:

The code is executed and expected output is printed on the screen.

Question-11:

Aim:

Draw the histogram plot for the Experience column.

Description:

Histograms are an accurate representation of frequency distribution of numerical data. plt.hist() method displays the histogram plot for the passed column of the dataframe.

Algorithm:

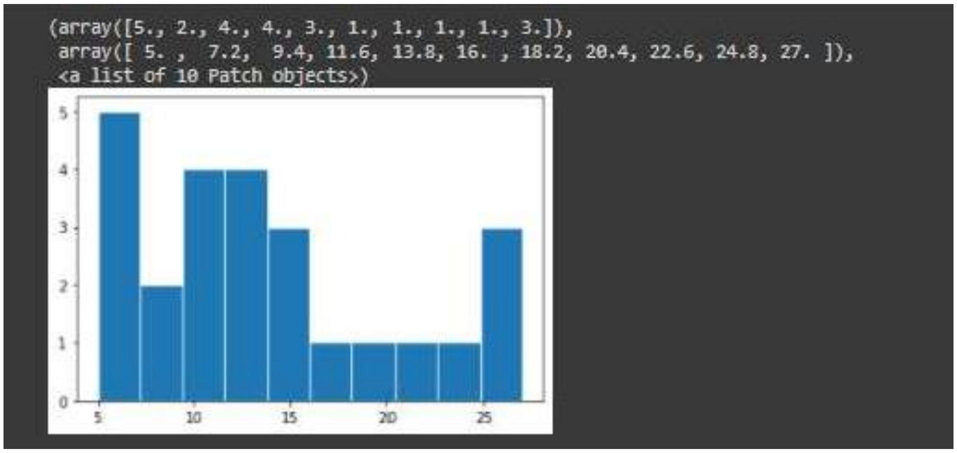
Step-1: Use plt.hist() method to display the histogram of the given column.

Step-2: Pass the experience column of the dataframe in the plt.hist() method.

Code:

```
plt.hist(df['Experience'],edgecolor='white')
```

Sample Input & Output:



```
(array([5., 2., 4., 4., 3., 1., 1., 1., 1., 3.]),
 array([ 5. ,  7.2,  9.4, 11.6, 13.8, 16. , 18.2, 20.4, 22.6, 24.8, 27. ]),
 <a list of 10 Patch objects>)
```

Result:

The code is executed and expected output is printed on the screen.

Question-12:

Aim:

Draw the histogram plot for Experience column with bin value and PDF.

Description:

Histograms are an accurate representation of frequency distribution of numerical data. Barsof a histogram are called bins and the height of each bin shows how many values from that data fall into that range. A pdf is a function that describes the probability that a random variable will take a certain value in histograms.

Algorithm:

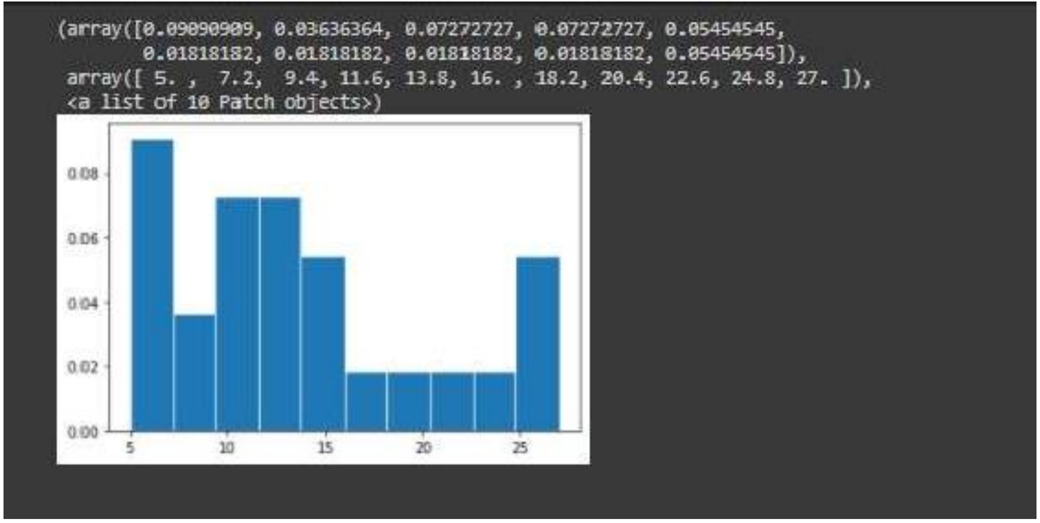
Step-1: Use plt.hist() method to display the histogram of the given column.

Step-2: Pass the experience column of the dataframe, bins value, edge color and density inthe plt.hist() method .

Code:

```
plt.hist(df['Experience'],edgecolor='white',bins=10,density=True)
```

Sample Input & Output:



```
(array([0.09090909, 0.03636364, 0.07272727, 0.07272727, 0.05454545,
        0.01818182, 0.01818182, 0.01818182, 0.01818182, 0.05454545]),
 array([ 5. ,  7.2,  9.4, 11.6, 13.8, 16. , 18.2, 20.4, 22.6, 24.8, 27. ]),
 <a list of 10 Patch objects>)
```

Result:

The code is executed and expected output is printed on the scre

Ex. No.4	Exploratory Data Analysis
10.08.22	

Dataset: Emp_EDA.csv

Aim:

To work with dataset using pandas and scipy to perform Exploratory Data Analysis.

Algorithm:

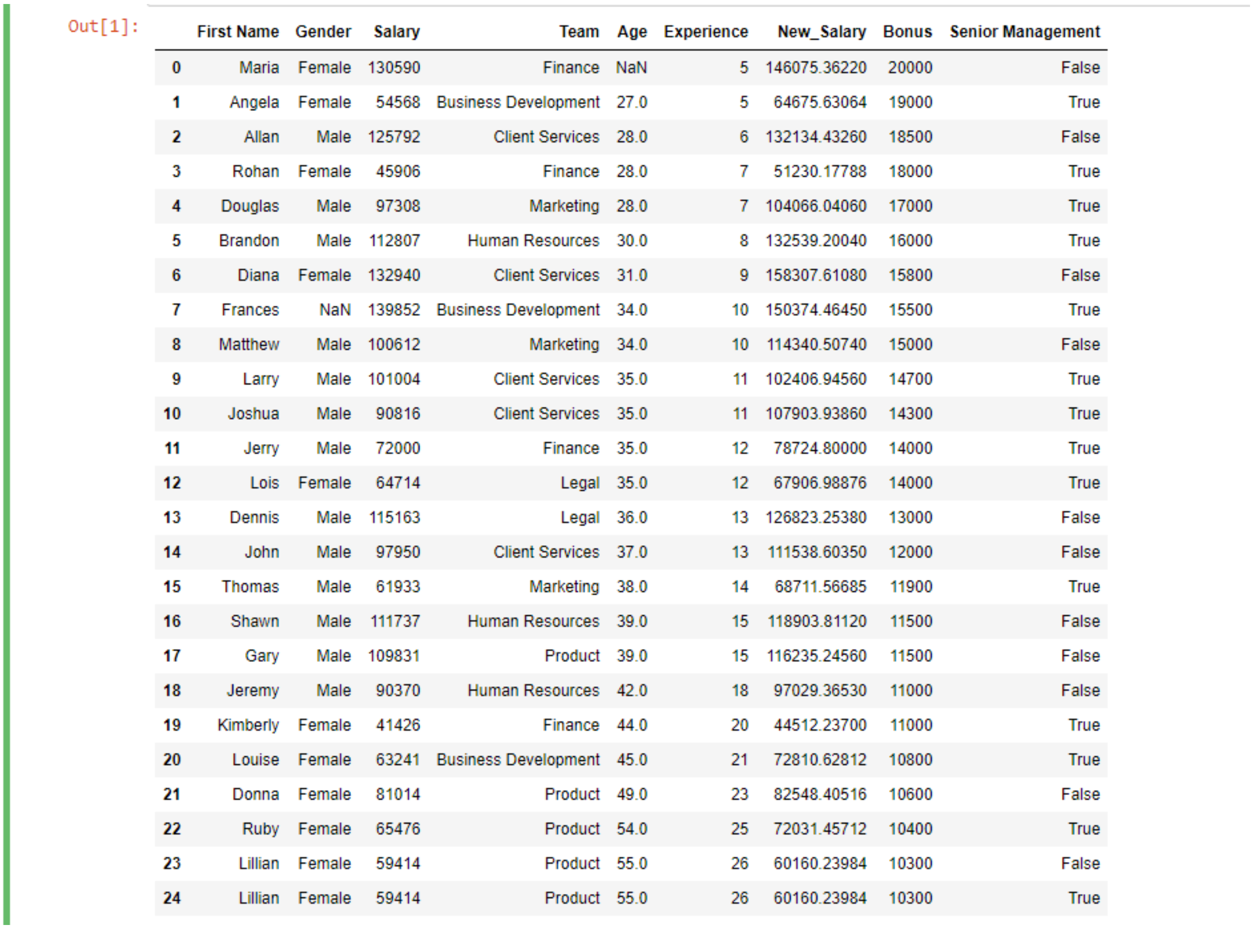
Step 1: Importing the pandas and scipy libraries.

Step 2: Creating a variable to store the read_csv file as a Data Frame. Print the dataframe to check the data is printed.

Program:

```
import pandas as pd

import scipy as stats


data = pd.read_csv('Emp_EDA.csv')

data
```

Output:



Out[1]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600	False
22	Ruby	Female	65476	Product	54.0	25	72031.45712	10400	True
23	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	False
24	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	True

QUESTION -1:

Remove the irrelevant column ‘Senior Management’ (inplace=True)

DESCRIPTION:

The drop() method removes the specified row or column.

By specifying the column axis (axis='columns'), the drop() method removes the specified column.

By specifying the row axis (axis='index'), the drop() method removes the specified row.

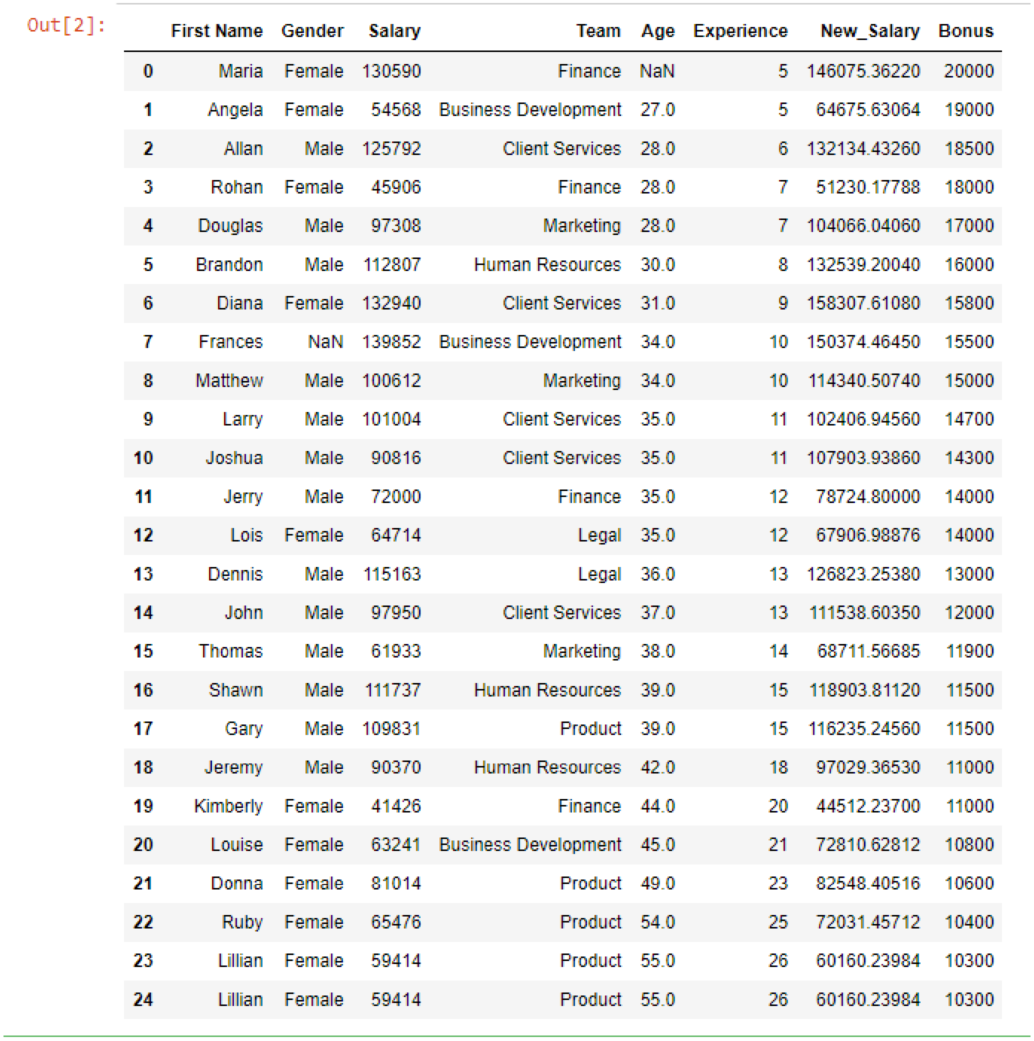
ALGORITHM:

Step 1: Removing the column using the drop function in pandas with a parameter of columns and inplace

PROGRAM:

```
data.drop(columns = ['Senior Management'], inplace=True)

data
```

OUTPUT:



Out[2]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600
22	Ruby	Female	65476	Product	54.0	25	72031.45712	10400
23	Lillian	Female	59414	Product	55.0	26	60160.23984	10300
24	Lillian	Female	59414	Product	55.0	26	60160.23984	10300

QUESTION-2

Remove the duplicate rows and analyze

DESCRIPTION:

The drop_duplicates() method removes duplicate rows.

Use the subset parameter if only some specified columns should be considered when looking for duplicates.

ALGORITHM:

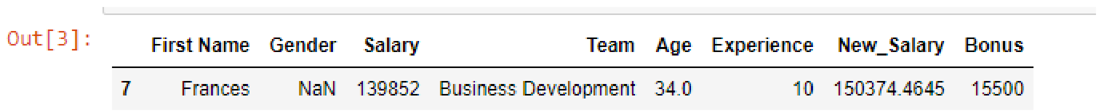
Step 1: Removing the duplicate rows and columns using the drop_duplicates() function with a parameter of subset, keep and inplace true.

Step 2: The subset will check the whole data frame to remove the duplicate rows and columns.

PROGRAM:

```
data.drop_duplicates(subset = "Gender", keep=False, inplace=True)
data
```

OUTPUT:



Out[3]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus
7	Frances	NaN	139852	Business Development	34.0	10	150374.4645	15500

QUESTION-3

Rename the column bonus to Incentive

DESCRIPTION:

The rename() method allows you to change the row indexes, and the columns labels.

The index, columns, axis, copy, inplace, level, errors parameters are keyword arguments.

ALGORITHM:

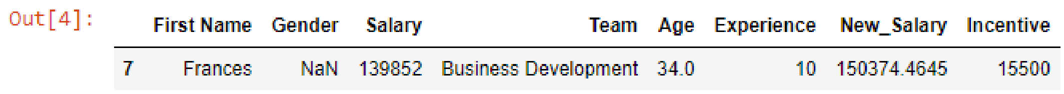
Step 1: The rename function in pandas is used to rename the specific columns.

Step 2: The parameter of columns will change the column ‘Bonus’ to ‘Incentive’ using the functions.

PROGRAM:

```
data.rename(columns={'Bonus':'Incentive'}, inplace=True)

data
```

OUTPUT:



Out[4]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Incentive
7	Frances	NaN	139852	Business Development	34.0	10	150374.4645	15500

QUESTION-4

Calculate the central tendency measures for ‘Experience’

DESCRIPTION:

The mean() method returns a Series with the mean value of each column.

The median() method returns a Series with the median value of each column.

The mode of a set of values is the value that appears most often.

ALGORITHM:

Step 1: Creating another data frame as data1 to perform the central tendancy measures of mean, median and mode.

Step 2: Applying the function of mean, median and mode to calculate the central tendancy measures for the Column ‘Experience’

Step 3: Calculating the mean, median and mode for the ‘Experience’ column and printing the results at the end.

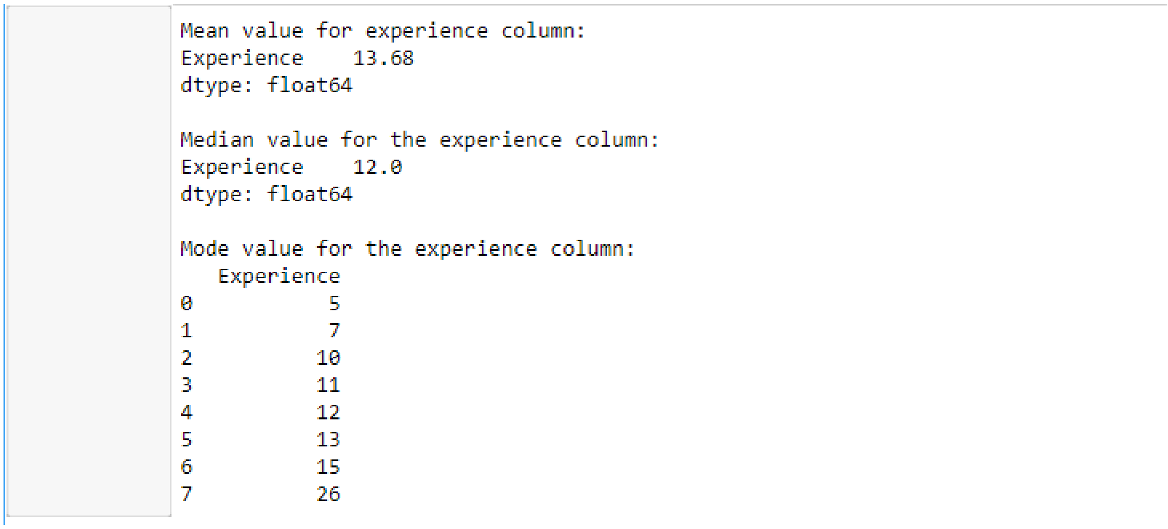
PROGRAM:

```
data1 = pd.read_csv('Emp_EDA.csv')
print("Mean value for experience column:")
mean = data1[["Experience"]].mean()
print(mean)

print()

print("Median value for the experience column:")
median = data1[["Experience"]].median()
print(median)

print()

print("Mode value for the experience column:")
```

mode = data1[["Experience"]].mode()

print(mode)

OUTPUT:



```
Mean value for experience column:
Experience    13.68
dtype: float64

Median value for the experience column:
Experience    12.0
dtype: float64

Mode value for the experience column:
   Experience
0           5
1           7
2          10
3          11
4          12
5          13
6          15
7          26
```

QUESTION-5

Calculate the variability measures for ‘Experience’

DESCRIPTION:

The min() method returns a Series with the minimum value of each column. By specifying the column axis (axis='columns'), the max() method searches column-wise and returns the minimum value for each row.

The var() method calculates the standard deviation for each column. By specifying the column axis (axis='columns'), the var() method searches column-wise and returns the standard deviation for each row.

The Pandas std() is defined as a function for calculating the standard deviation of the given set of numbers, DataFrame, column, and rows. In respect to calculate the standard deviation, we need to import the package named "statistics" for the calculation of median.

ALGORITHM:

Step 1: Using the min() and max() function in pandas to get the Range value of the variablility measures.

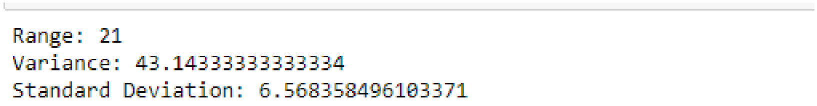
Step 2: Using the var() function in pandas to get the Variance for the column Experience.

Step 3: Using the std() method in pandas to get the Standard Deviation for the column Experience.

PROGRAM:

```
print('Range:',data1['Experience'].max() - data1['Experience'].min())

print('Variance:',data1['Experience'].var())

print('Standard Deviation:',data1['Experience'].std())
```

OUTPUT:



```
Range: 21
Variance: 43.14333333333334
Standard Deviation: 6.568358496103371
```

QUESTION-6

Calculate the IQR using quantile for ‘Experience’

DESCRIPTION:

The quantile() method calculates the quantile of the values in a given axis. Default axis is row. By specifying the column axis (axis='columns'), the quantile() method calculates the quantile column-wise and returns the mean value for each row.

ALGORITHM:

Step 1: To find the IQR using quantile, first to get the q1 to get the quantile using 0.25.

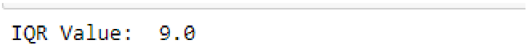
Step 2: To get the quantile(0.75) for the column Experience. After storing the two quantile values, Subtract the q3 value to q1 value.

Step 3: Print the IQR Results to get the IQR for the Experience columnn.

PROGRAM:

```
q1 = data1['Experience'].quantile(0.25)

q3 = data1['Experience'].quantile(0.75)

IQR = q3 - q1

print("IQR Value: ", IQR)
```

OUTPUT:



```
IQR Value:  9.0
```

QUESTION-7

Calculate the z-score for ‘Experience’

DESCRIPTION:

Simply put, a z-score (also called a standard score) gives you an idea of how far from the mean a data point is. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is. A z-score can be placed on a normal distribution curve.

scipy.stats.zscore(arr, axis=0, ddof=0) function computes the relative Z-score of the input data, relative to the sample mean and standard deviation.

ALGORITHM:

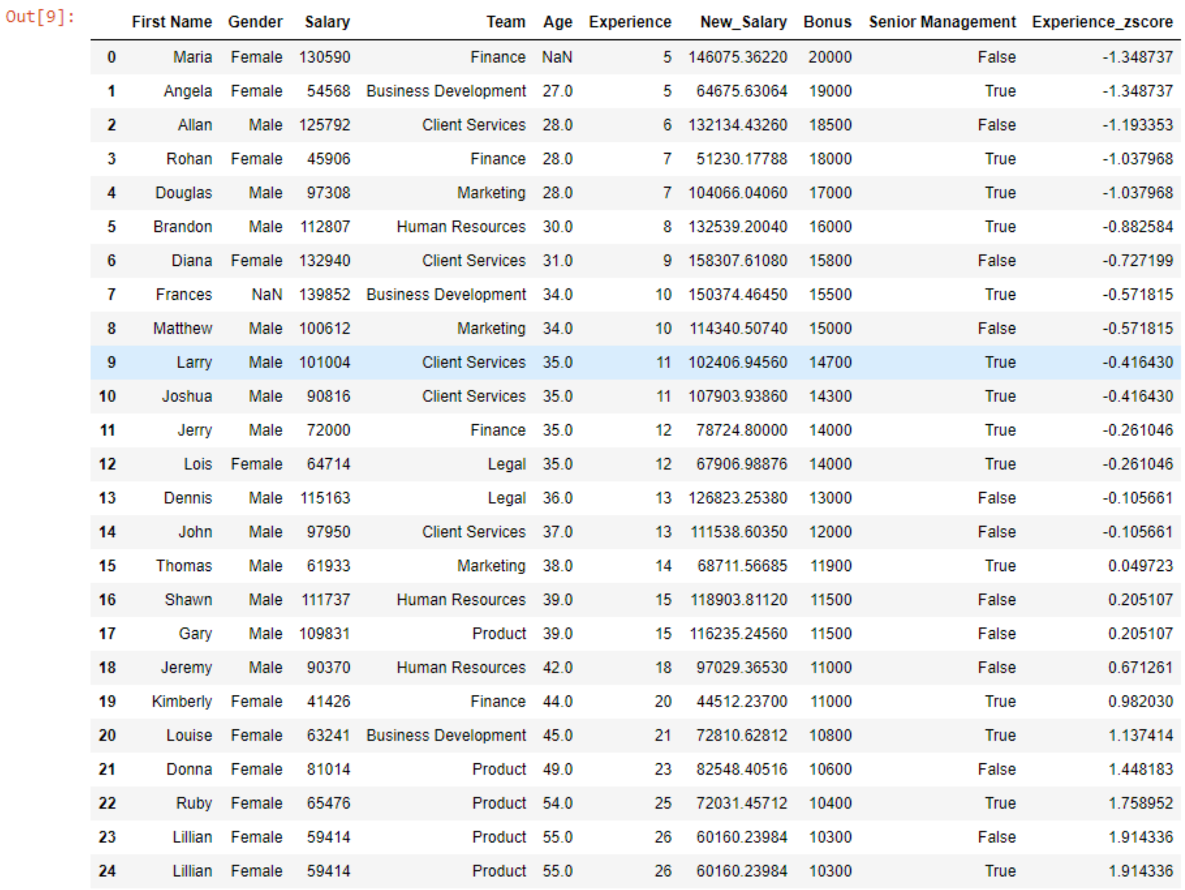
Step 1: To get the zscore for the Experience column, have to import the Scipy library.

Step 2: using the data1 frame and fetching the experience column to fill the columns using zero, using the stats.zscore function to calculating the zscore for the experience column.

PROGRAM:

```
import scipy

from scipy import stats

data1['Experience'].fillna(0, inplace=True)

data1['Experience_zscore']=stats.zscore(data1['Experience'])

data1
```

OUTPUT:



Out[9]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management	Experience_zscore
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False	-1.348737
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True	-1.348737
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False	-1.193353
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True	-1.037968
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True	-1.037968
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True	-0.882584
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False	-0.727199
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True	-0.571815
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False	-0.571815
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True	-0.416430
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True	-0.416430
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True	-0.261046
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True	-0.261046
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False	-0.105661
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False	-0.105661
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True	0.049723
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False	0.205107
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False	0.205107
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False	0.671261
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True	0.982030
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True	1.137414
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600	False	1.448183
22	Ruby	Female	65476	Product	54.0	25	72031.45712	10400	True	1.758952
23	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	False	1.914336
24	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	True	1.914336

QUESTION-8

Add two rows at the end of the dataframe with the given values.

DESCRIPTION:

The append() method appends a DataFrame-like object at the end of the current DataFrame.

The append() method returns a new DataFrame object, no changes are done with the original DataFrame.

ALGORITHM:

Step 1: Using the append function creating the datas as dictionary values and append it to the data frame.

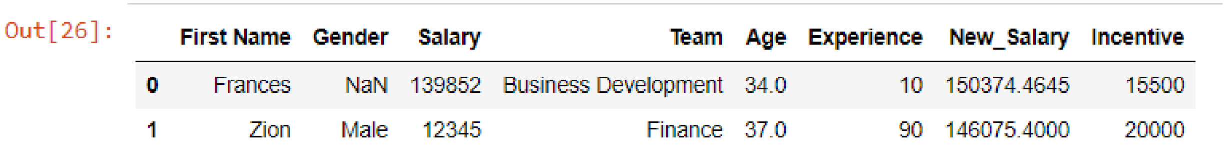
Step 2: After appending the two rows in the data frame, the results are printed the new table values.

PROGRAM:

```
data.append({'First Name':'Zion', 'Gender':'Male', 'Salary':'12345', 'Team':'Finance', 'Age':37, 'Experience':90, 'New_Salary':146075.4, 'Incentive':20000}, ignore_index=True)
```

```
data.append({'First Name':'Frances', 'Gender':'Male', 'Salary':'13952', 'Team':'Business Development', 'Age':39, 'Experience':95, 'New_Salary':150374.5, 'Incentive':15500}, ignore_index=True)
```

OUPUT:



Out[26]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Incentive
0	Frances	NaN	139852	Business Development	34.0	10	150374.4645	15500
1	Zion	Male	12345	Finance	37.0	90	146075.4000	20000

QUESTION-9

Replace the nan value with give value (Salary=130590)

DESCRIPTION:

The fillna() method replaces the NULL values with a specified value. The fillna() method returns a new DataFrame object unless the inplace parameter is set to True , in that case the fillna() method does the replacing in the original DataFrame instead.

ALGORITHM:

Step 1: Create a new dataframe to replace the nan values with the given value (salary = 130590)

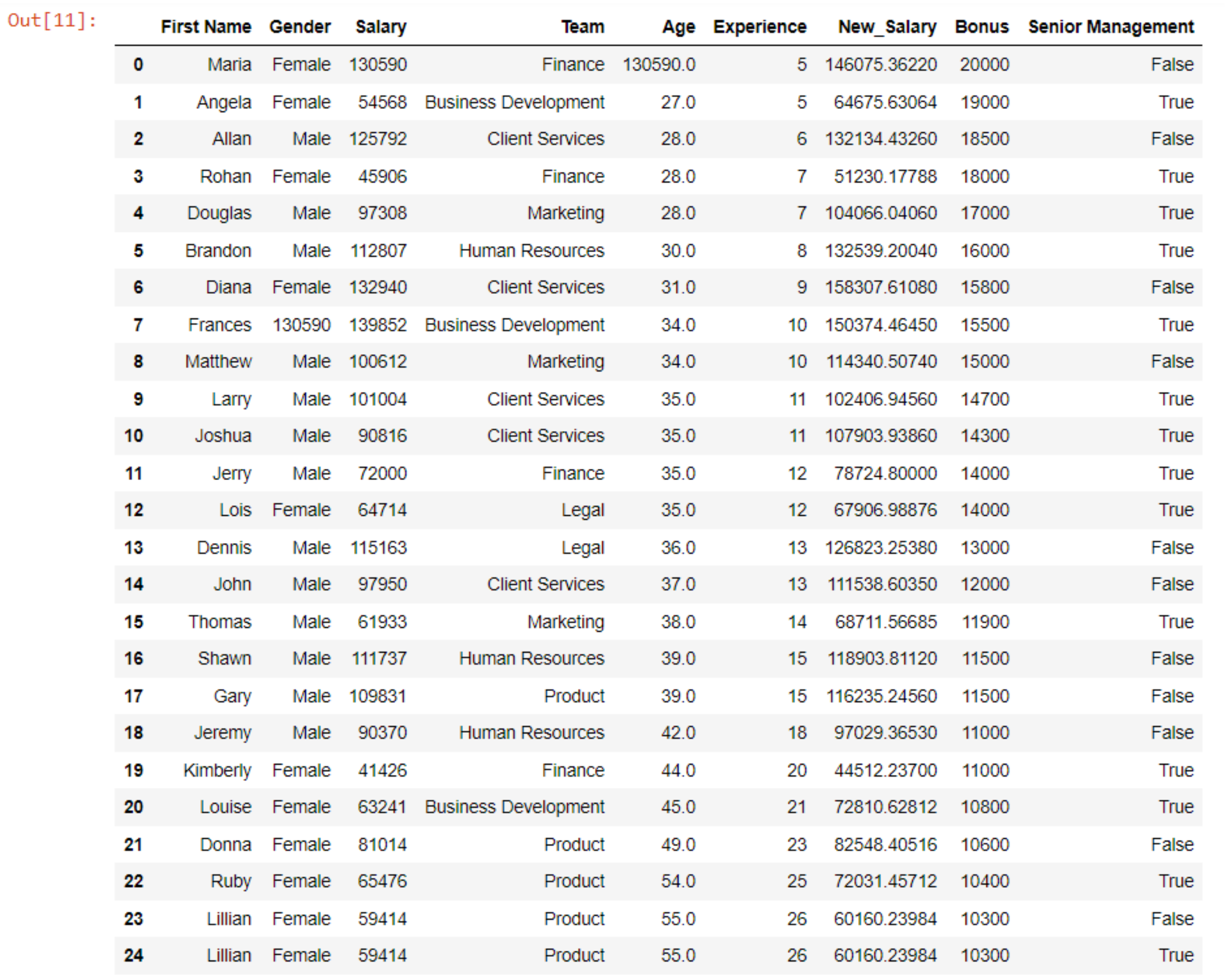
Step 2: Use data.fillna(130590) to fill the nan values in the dataframe.

Step 3: Printing out the results getting the nan values in the dataframe.

PROGRAM:

```
data2 = pd.read_csv('Emp_EDA.csv')

data2.fillna(130590)
```

OUTPUT:



Out[11]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	130590.0	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	130590	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600	False
22	Ruby	Female	65476	Product	54.0	25	72031.45712	10400	True
23	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	False
24	Lillian	Female	59414	Product	55.0	26	60160.23984	10300	True

QUESTION-10

Replace the nan value in salary column with the previous value, next value, linear nterpolation and the central tendancey measures.

DESCRIPTION:

The fillna() method replaces the NULL values with a specified value.

The fillna() method returns a new DataFrame object unless the inplace parameter is set to True, in that case the fillna() method does the replacing in the original DataFrame instead.

Method - Optional, default None'. Specifies the method to use when replacing

ALGORITHM:

Step 1: Read the Emp_EDA.csv to perform the following operations.

Step 2: To replace the nan values in the salary column, use the fillna function with the respected parameters.

Step 3: The fillna parameter has the four different types of methods, they are.

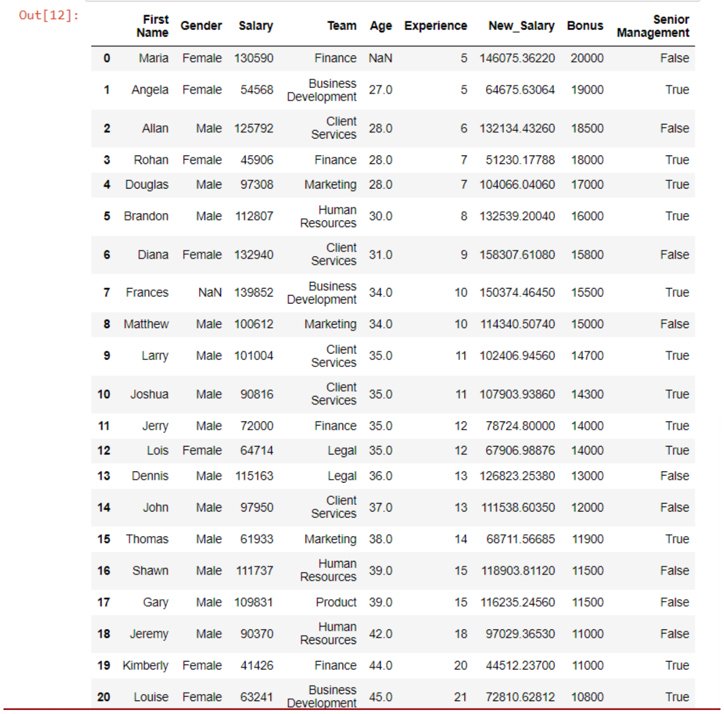
Step 4: Print the results accordingly to see the replaced methods.

PROGRAM:

```
data3 = pd.read_csv('Emp_EDA.csv')

data3['Salary'].fillna(method='pad', inplace=True)

data3


data4 = pd.read_csv('Emp_EDA.csv')

data4['Salary'].fillna(method='bfill', inplace=True)

data4


data5 = pd.read_csv('Emp_EDA.csv')

data5['Salary'].interpolate(method='linear', limit_direction='forward', inplace=True)

data5


data6 = pd.read_csv('Emp_EDA.csv')

data6['Salary'].fillna(data['Salary'].mean(), inplace=True)

data6
```

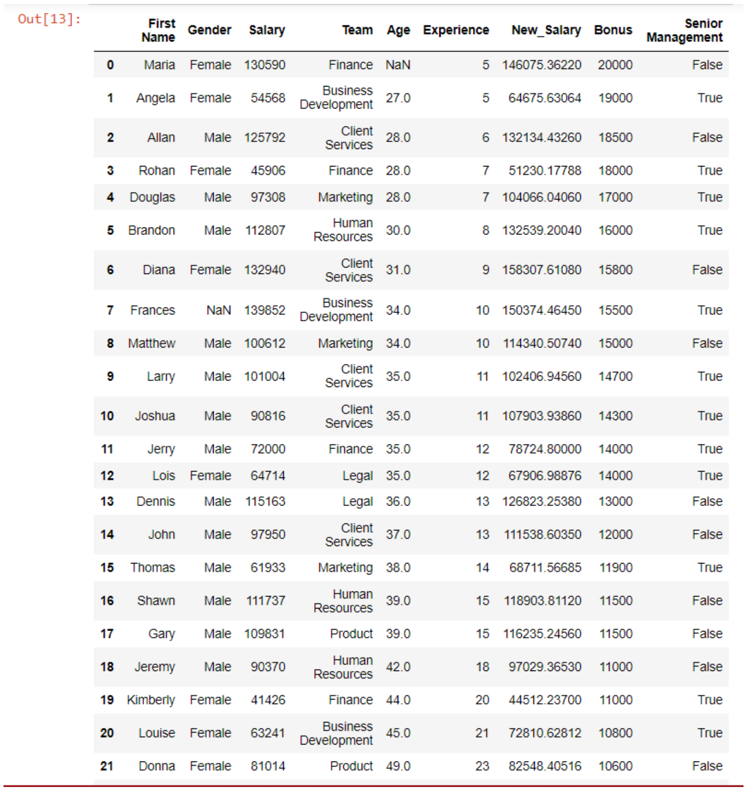
```
data7 = pd.read_csv('Emp_EDA.csv')

data7['Salary'].fillna(data['Salary'].median(), inplace=True)

data7
```

OUTPUT:



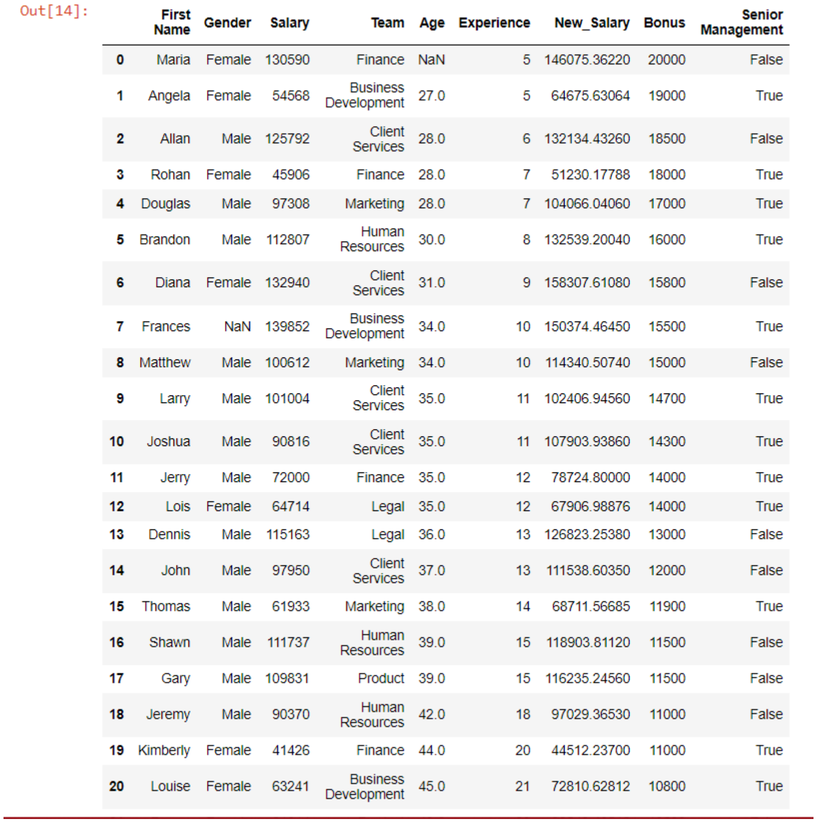
Out[12]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True



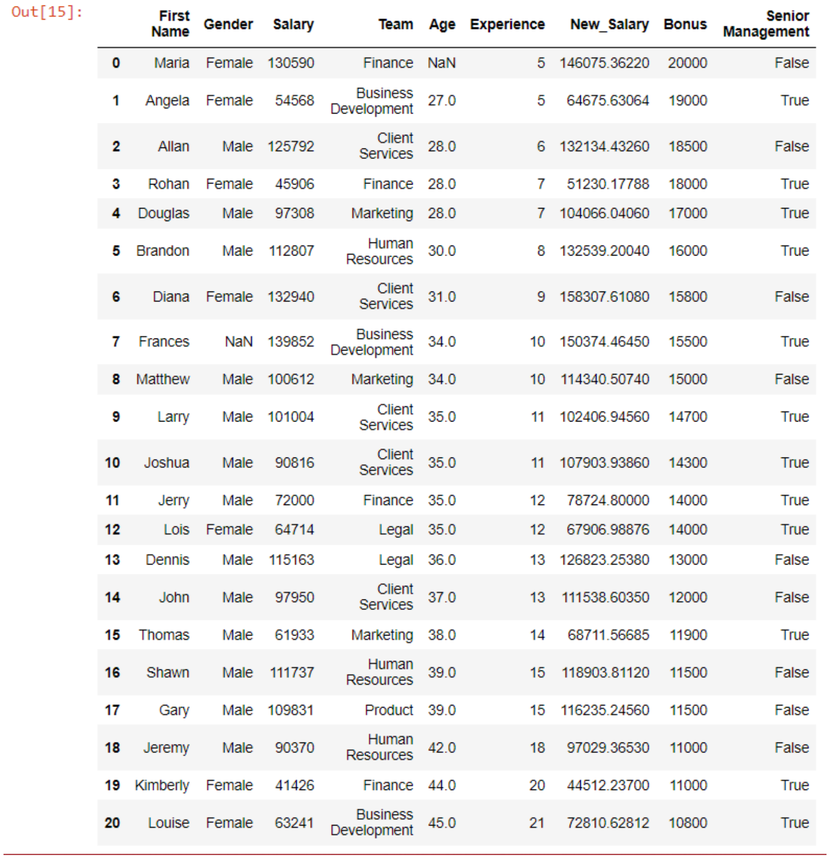
Out[13]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600	False



Out[14]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True



Out[15]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True

QUESTION-11

Detect the outliers in updated ‘Experience’ with boxplot, scatter plot and histogram

DESCRIPTION:

A box plot which is also known as a whisker plot displays a summary of a set of data containing the minimum, first quartile, median, third quartile, and maximum. In a box plot, we draw a box from the first quartile to the third quartile. A vertical line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum.

A Scatter plot is a diagram where each value in the data set is represented by a dot. The Matplotlib module has a method for drawing scatter plots, it needs two arrays of the same length, one of the values of the x-axis, and one for the values of the y-axis.

A histogram is a graphical display of data using bars of different heights. In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range. A histogram displays the shape and spread of continuous sample data.

ALGORITHM:

Step 1: Import the necessary libraries to plot the graphical data visualization.

Step 2: Create and import the data frame to read the csv.

Step 3: Use plt.boxplot to provide the ‘Experience’ Column inside the box plot.

Step 4: Use plt.show function to show the box plot.

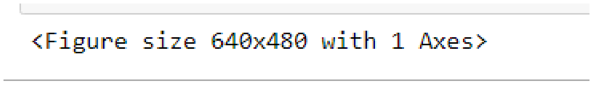
Step 5: For Scatter plot use linewidth, markers and color as a parameters inside the plot function.

Step 6: For the Histogram plot, use edgecolor and color with the histogram function.

PROGRAM:

```
import matplotlib.pyplot as plt

data8 = pd.read_csv('Emp_EDA.csv')

plt.boxplot(data8['Experience'])

plt.show()
```

OUTPUT:



```
<Figure size 640x480 with 1 Axes>
```

QUESTION-12

Remove the outliers using IQR by recalculating the IQR in the updated ‘Experience’ and analyze with box plot.

DESCRIPTION:

The IQR describes the middle 50% of values when ordered from lowest to highest. To find the interquartile range (IQR), first find the median (middle value) of the lower and upper half of the data. These values are quartile 1 (Q1) and quartile 3 (Q3). The IQR is the difference between Q3 and Q1.

ALGORITHM:

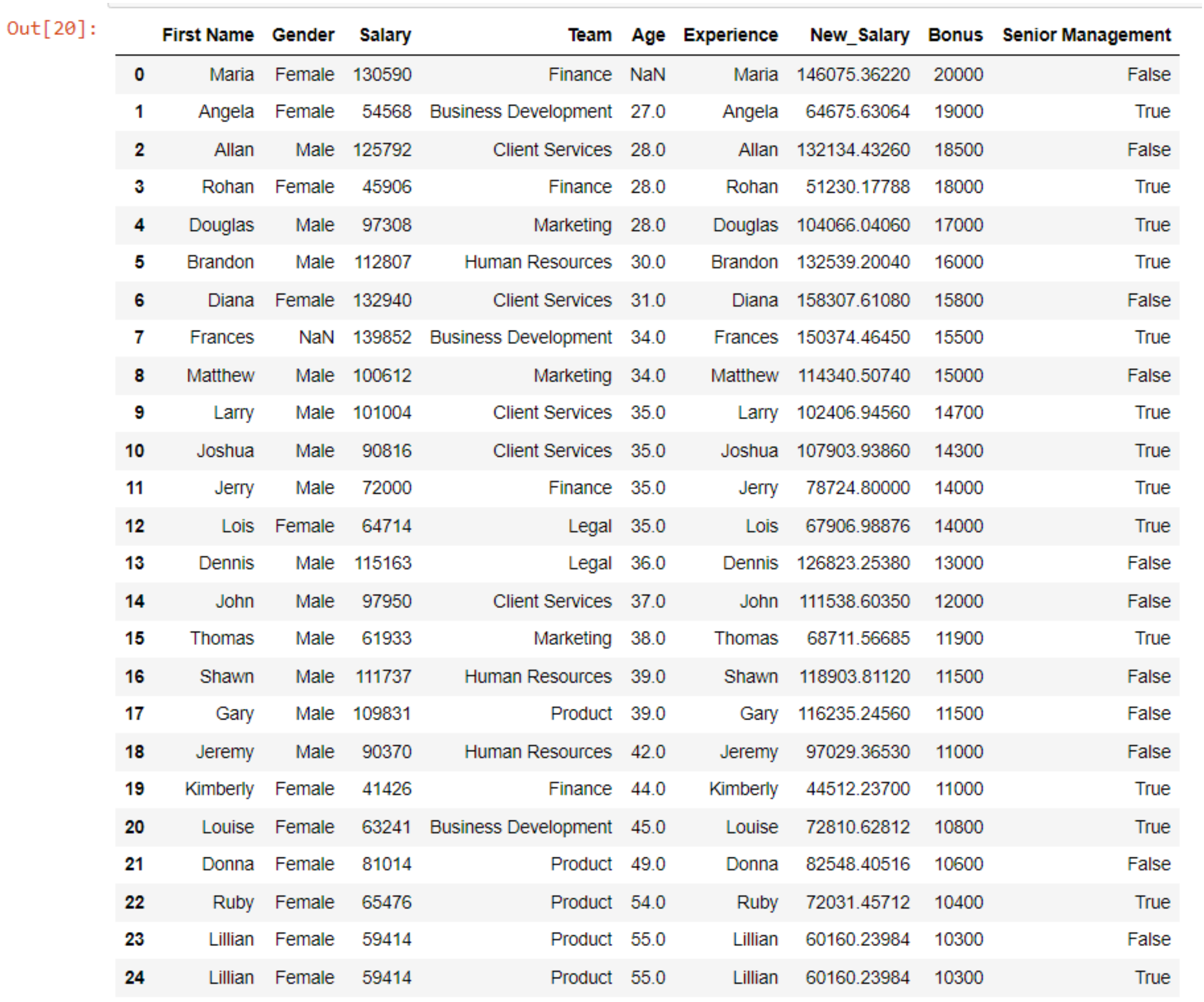
Step 1: To find the Quartile, Create two variables containing the Q3 which is subtracted from Q1 for finding the Quartile Range.

Step2: To create a data, use the parameter inside the dataframe section to solve the Experience column.

PROGRAM:

```
Q1c=data7['Experience'].quantile(0.25)

Q3c=data7['Experience'].quantile(0.75)

IQRc = Q3c-Q1c

l=Q1c-1.5*IQRc

h=Q3c+1.5*IQRc

data7['Experience']=data7[(data7['Experience']>l) | (data7['Experience']< h)]

data7
```

OUTPUT:



Out[20]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	Maria	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	Angela	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	Allan	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	Rohan	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	Douglas	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	Brandon	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	Diana	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	Frances	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	Matthew	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	Larry	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	Joshua	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	Jerry	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	Lois	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	Dennis	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	John	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	Thomas	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	Shawn	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	Gary	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	Jeremy	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	Kimberly	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	Louise	72810.62812	10800	True
21	Donna	Female	81014	Product	49.0	Donna	82548.40516	10600	False
22	Ruby	Female	65476	Product	54.0	Ruby	72031.45712	10400	True
23	Lillian	Female	59414	Product	55.0	Lillian	60160.23984	10300	False
24	Lillian	Female	59414	Product	55.0	Lillian	60160.23984	10300	True

QUESTION-13

Remove the outlers using z-score by recalculating the z-score in updated ‘Experience’ and analyze with box plot.

DESCRIPTION:

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. If a Z-score is 0, it indicates that the data point's score is identical to the mean score.

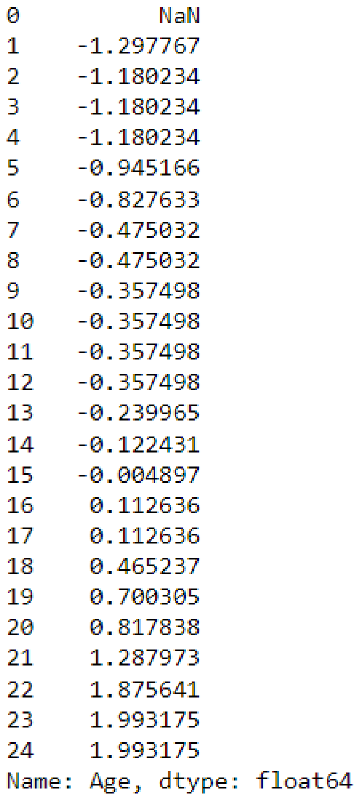
ALGORITHM:

Step 1: Calulate the z score by using the data with the column of Age by finding the mean value of age and dividing the standard deviation.

PROGRAM:

```
df_zscore = (data7['Age'] - data7['Age'].mean())/data7['Age'].std()

print(df_zscore)
```

OUTPUT:



```
0          NaN
1    -1.297767
2    -1.180234
3    -1.180234
4    -1.180234
5    -0.945166
6    -0.827633
7    -0.475032
8    -0.475032
9    -0.357498
10   -0.357498
11   -0.357498
12   -0.357498
13   -0.239965
14   -0.122431
15   -0.004897
16    0.112636
17    0.112636
18    0.465237
19    0.700305
20    0.817838
21    1.287973
22    1.875641
23    1.993175
24    1.993175
Name: Age, dtype: float64
```

QUESTION-14

Plot the heatmap using the correlation

DESCRIPTION:

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

ALGORITHM:

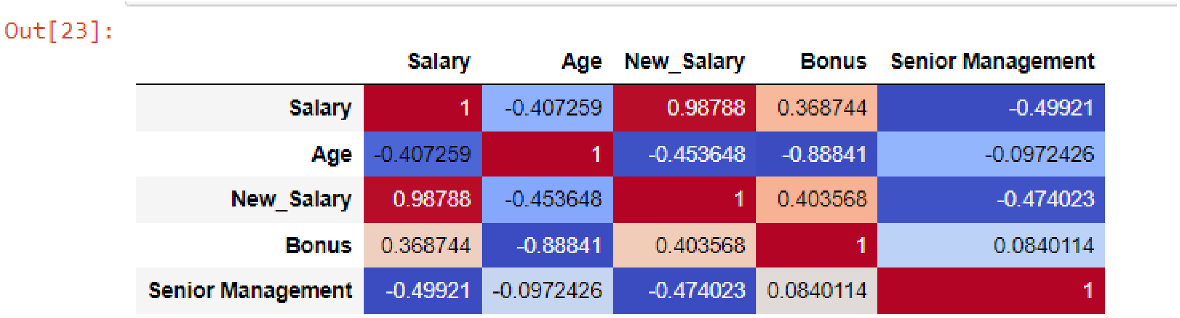
Step 1: To plot the heat map using the pandas by correleating the dataframe of the Emp_edu dataset. Create a style color of coolwarm to print the beautifully designed Heatmap.

PROGRAM:

corr = data7.corr()

corr.style.background_gradient(cmap='coolwarm')

OUTPUT:



Out[23]:

	Salary	Age	New_Salary	Bonus	Senior Management
Salary	1	-0.407259	0.98788	0.368744	-0.49921
Age	-0.407259	1	-0.453648	-0.88841	-0.0972426
New_Salary	0.98788	-0.453648	1	0.403568	-0.474023
Bonus	0.368744	-0.88841	0.403568	1	0.0840114
Senior Management	-0.49921	-0.0972426	-0.474023	0.0840114	1

QUESTION-15

Drop the last two rows added in the dataframe

DESCRIPTION:

The drop() method removes the specified row or column.

By specifying the column axis (axis='columns'), the drop() method removes the specified column.

By specifying the row axis (axis='index'), the drop() method removes the specified row.

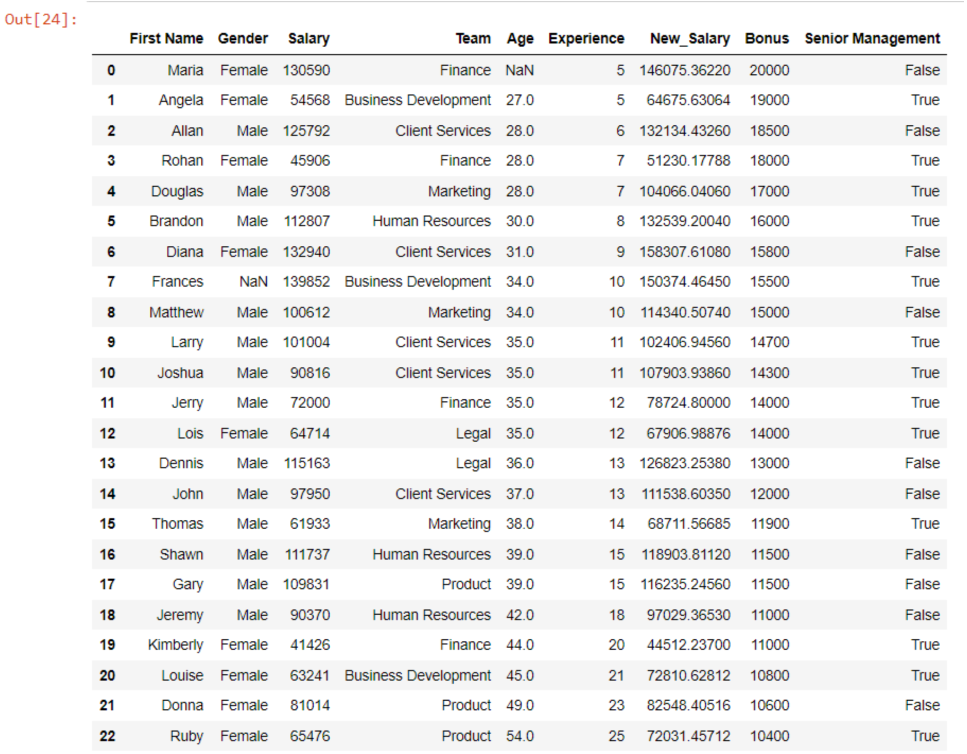
ALGORITHM:

Step 1: Calling the dataframe and attach the drop function and give the axis values. With another parameter as inplace=True which will not change the behaviour of the dataframe.

PROGRAM:

```
data2.drop([23, 24], inplace=True)

data2
```

OUTPUT:



Out[24]:

	First Name	Gender	Salary	Team	Age	Experience	New_Salary	Bonus	Senior Management
0	Maria	Female	130590	Finance	NaN	5	146075.36220	20000	False
1	Angela	Female	54568	Business Development	27.0	5	64675.63064	19000	True
2	Allan	Male	125792	Client Services	28.0	6	132134.43260	18500	False
3	Rohan	Female	45906	Finance	28.0	7	51230.17788	18000	True
4	Douglas	Male	97308	Marketing	28.0	7	104066.04060	17000	True
5	Brandon	Male	112807	Human Resources	30.0	8	132539.20040	16000	True
6	Diana	Female	132940	Client Services	31.0	9	158307.61080	15800	False
7	Frances	NaN	139852	Business Development	34.0	10	150374.46450	15500	True
8	Matthew	Male	100612	Marketing	34.0	10	114340.50740	15000	False
9	Larry	Male	101004	Client Services	35.0	11	102406.94560	14700	True
10	Joshua	Male	90816	Client Services	35.0	11	107903.93860	14300	True
11	Jerry	Male	72000	Finance	35.0	12	78724.80000	14000	True
12	Lois	Female	64714	Legal	35.0	12	67906.98876	14000	True
13	Dennis	Male	115163	Legal	36.0	13	126823.25380	13000	False
14	John	Male	97950	Client Services	37.0	13	111538.60350	12000	False
15	Thomas	Male	61933	Marketing	38.0	14	68711.56685	11900	True
16	Shawn	Male	111737	Human Resources	39.0	15	118903.81120	11500	False
17	Gary	Male	109831	Product	39.0	15	116235.24560	11500	False
18	Jeremy	Male	90370	Human Resources	42.0	18	97029.36530	11000	False
19	Kimberly	Female	41426	Finance	44.0	20	44512.23700	11000	True
20	Louise	Female	63241	Business Development	45.0	21	72810.62812	10800	True
21	Donna	Female	81014	Product	49.0	23	82548.40516	10600	False
22	Ruby	Female	65476	Product	54.0	25	72031.45712	10400	True

Result:

The program to execute the above programs are compiled and output is verified successfully.

Ex. No.5	Statistical Inference
7/09/22	

Batch-1

Dataset: diamonds.csv

Aim:

To work with dataset to perform Statistical Inference.

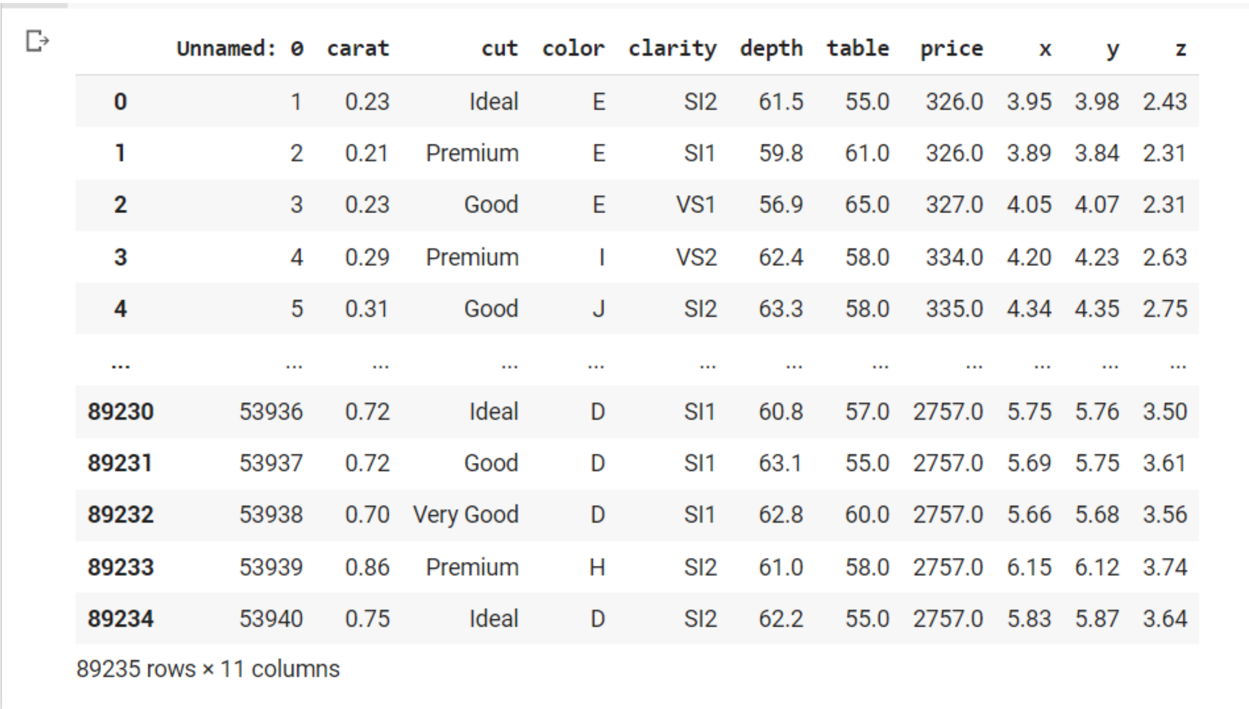
Algorithm:

- Importing the pandas, scipy, matplot, math and numpy libraries.
- Creating a variable to store the read_csv file as a Data Frame. Print the dataframe to check the data is printed.

Program:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats

import math

data = pd.read_csv("diamonds.csv")

data
```

Output:



```
       Unnamed: 0  carat        cut color clarity  depth  table   price     x     y     z
0               1   0.23      Ideal     E     SI2   61.5   55.0   326.0  3.95  3.98  2.43
1               2   0.21    Premium     E     SI1   59.8   61.0   326.0  3.89  3.84  2.31
2               3   0.23       Good     E     VS1   56.9   65.0   327.0  4.05  4.07  2.31
3               4   0.29    Premium     I     VS2   62.4   58.0   334.0  4.20  4.23  2.63
4               5   0.31       Good     J     SI2   63.3   58.0   335.0  4.34  4.35  2.75
...           ...    ...        ...   ...     ...    ...    ...     ...   ...   ...   ...
89230       53936   0.72      Ideal     D     SI1   60.8   57.0  2757.0  5.75  5.76  3.50
89231       53937   0.72       Good     D     SI1   63.1   55.0  2757.0  5.69  5.75  3.61
89232       53938   0.70  Very Good     D     SI1   62.8   60.0  2757.0  5.66  5.68  3.56
89233       53939   0.86    Premium     H     SI2   61.0   58.0  2757.0  6.15  6.12  3.74
89234       53940   0.75      Ideal     D     SI2   62.2   55.0  2757.0  5.83  5.87  3.64

89235 rows × 11 columns
```

QUESTION -1:

Calculate the sample mean for 'price' column with n=500 and observe

DESCRIPTION:

Python defines a set of functions that are used to generate or manipulate random numbers through the random module. Functions in the random module rely on a pseudo-random number generator function random (), which generates a random float number between 0.0 and 1.0.

mean() function can be used to calculate mean/average of a given list of numbers. It returns mean of the data set passed as parameters.
Arithmetic mean is the sum of data divided by the number of data-points. It is a measure of the central location of data in a set of values which vary in range.

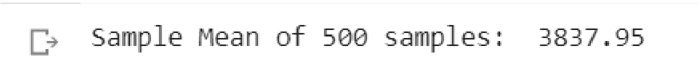
ALGORITHM:

- Initialize a variable n and assign a value to 100
- Use random function to generate random values according to the size of the column
- Calculate the mean for the random values.
- Print the mean value generated as a result to end the program.

PROGRAM:

```
n = 100

samples = np.random.choice(a=data["price"], size=n)

mean1 = samples.mean()

print("Sample Mean of 500 samples: ", mean1)
```

OUTPUT:



```
Sample Mean of 500 samples:  3837.95
```

QUESTION -2:

2) Calculate the sample mean for 'price' column with n=1000 and observe

DESCRIPTION:

Python defines a set of functions that are used to generate or manipulate random numbers through the random module. Functions in the random module rely on a pseudo-random number generator function random (), which generates a random float number between 0.0 and 1.0.

mean () function can be used to calculate mean/average of a given list of numbers. It returns mean of the data set passed as parameters.
Arithmetic mean is the sum of data divided by the number of data-points. It is a measure of the central location of data in a set of values which vary in range.

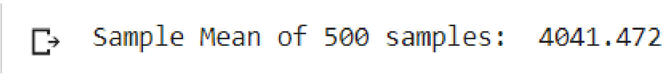
ALGORITHM:

- Initialize a variable n1 and assign a value 1000 to calculate a sample mean of thousand values.
- Initialize a variable of sample to generate a random value in it.
- Calculate the mean for the values.
- Print the results using the print function.

PROGRAM:

```
n1 = 1000

sample1 = np.random.choice(a=data["price"], size=n1)

mean2 = sample1.mean()

print("Sample Mean of 500 samples: ", mean2)
```

OUTPUT:



```
Sample Mean of 500 samples:  4041.472
```

QUESTION -3:

3) Calculate the population mean for 'price' column

DESCRIPTION:

The population mean is the mean or average of all values in the given population and is calculated by the sum of all values in population denoted by the summation of X divided by the number of values in population which is denoted by N.

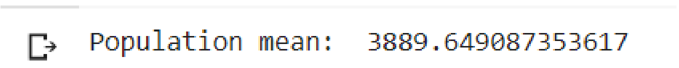
ALGORITHM:

- Initialize a variable population to store the price data mean from the dataset.
- Print the population variable inside the print function to obtained the output.

PROGRAM:

```
population = data["price"].mean()

print("Population mean: ", population)
```

OUTPUT:



```
Population mean:  3889.649087353617
```

QUESTION -4:

4) Calculate the confidence interval (CI) with sample mean for 'price' column of

# n=500 and confidence level of 95%. Observe whether the population mean lies in CI.

DESCRIPTION:

Z-score is also known as standard score gives us an idea of how far a data point is from the mean. It indicates how many standard deviations an element is from the mean. Hence, Z-Score is measured in terms of standard deviation from the mean.

A confidence interval is the mean of your estimate plus and minus the variation in that estimate. This is the range of values you expect your estimate to fall between if you redo your test, within a certain level of confidence. Confidence, in statistics, is another way to describe probability.

ALGORITHM:

- Initialize a variable and store the sample's standard deviation value.
- Initialize 0.95 as a confidence interval for the program
- Initialize an alpha value to calculate it using a formula.

PROGRAM:

```
print("Sample mean of 500 samples: ", mean1)

SD = samples.std()

print("Sample SD of 500 samples: ", SD)

CL = 0.95

alpha = (1-CL)/2

z_critical = round(stats.norm.ppf(1-alpha), 2)

print("Z_Score: ", z_critical)

er=z_critical*(SD/math.sqrt(n))
```

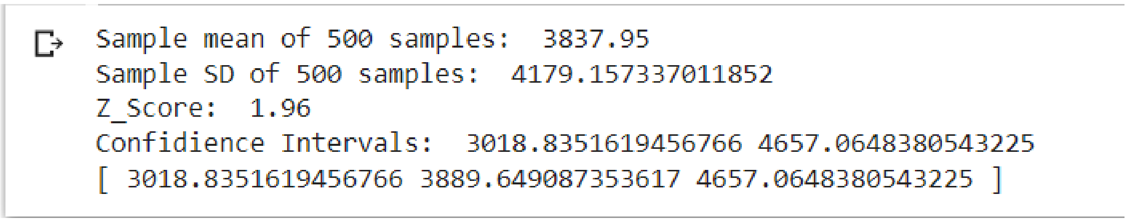
L=mean1-er

H=mean1+er

print("Confidience Intervals: ", L, H)

print("[",L,population,H,"]")

OUTPUT:



```
Sample mean of 500 samples:  3837.95
Sample SD of 500 samples:  4179.157337011852
Z_Score:  1.96
Confidience Intervals:  3018.8351619456766 4657.0648380543225
[ 3018.8351619456766 3889.649087353617 4657.0648380543225 ]
```

QUESTION -5:

5) Change the confidence level to 99% and observe the confidence interval for the same sample mean for ‘price’ column of n=500.

DESCRIPTION:

Z-score is also known as standard score gives us an idea of how far a data point is from the mean. It indicates how many standard deviations an element is from the mean. Hence, Z-Score is measured in terms of standard deviation from the mean.

A confidence interval is the mean of your estimate plus and minus the variation in that estimate. This is the range of values you expect your estimate to fall between if you redo your test, within a certain level of confidence. Confidence, in statistics, is another way to describe probability.

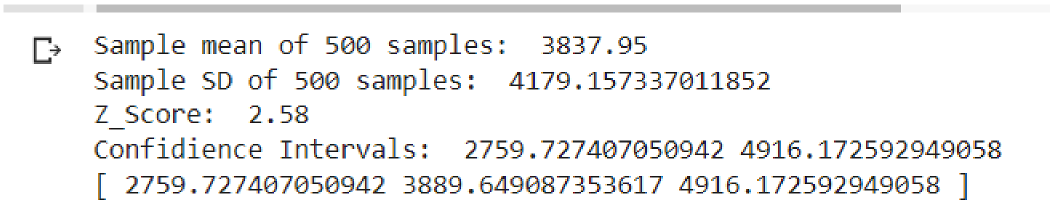
ALGORITHM:

- Initialize a variable SD to get the standard deviation of a sample values.
- Declare the confidence interval as 99 percentage.
- By Calculating the alpha formula for the confidence interval, calculate the z interval using the stats function from the scipy library.
- Print the confidence interval in the end to obtained the results.

PROGRAM:

```
print("Sample mean of 500 samples: ", mean1)

SD = samples.std()

print("Sample SD of 500 samples: ", SD)

CL = 0.99

alpha = (1-CL)/2

z_critical = round(stats.norm.ppf(1-alpha), 2)

print("Z_Score: ", z_critical)

er=z_critical*(SD/math.sqrt(n))

L=mean1-er

H=mean1+er

print("Confidience Intervals: ", L, H)

print("[",L,population,H,"]")
```

OUTPUT:



```
Sample mean of 500 samples:  3837.95
Sample SD of 500 samples:  4179.157337011852
Z_Score:  2.58
Confidience Intervals:  2759.727407050942 4916.172592949058
[ 2759.727407050942 3889.649087353617 4916.172592949058 ]
```

QUESTION -6:

Calculate and plot the Confidence Intervals for 25 Trials with n=500 and CI=95% for'price' column. Observe the results. [Note: Q7-Q8 consider the table to find the Correlation Coefficient]

DESCRIPTION:

A population is the complete set group of individuals, whether that group comprises a nation or a group of people with a common characteristic. In statistics, a population is the pool of individuals from which a statistical sample is drawn for a study.

ALGORITHM:

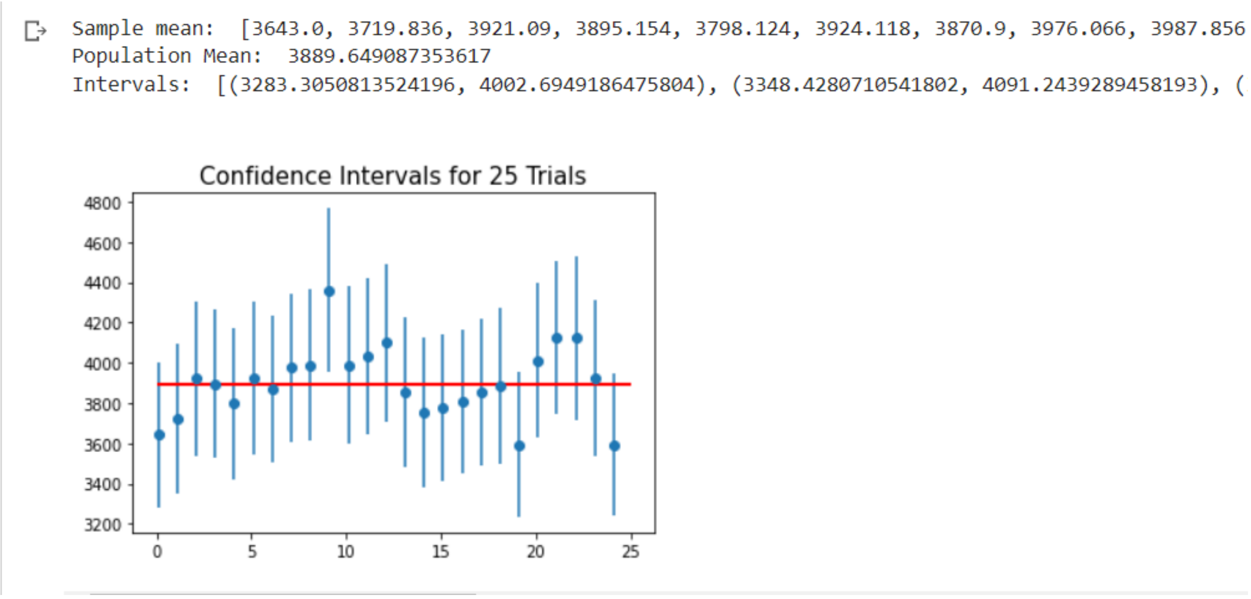
- Create two lists as interval and samples
- Take the confidence level as 0.95 and calculate the alpha as (1 – cl)/2
- Calculate zscore and calculate errors for all the 25 trails

PROGRAM:

```
sample_size=500

intervals = []

sample_means = []
```

```
CL = 0.95
ALPHA = (1-CL)/2
z_critical = round(stats.norm.ppf(1-ALPHA), 2)
p = data["price"].mean()

for samp in range(25):
  samp = np.random.choice(a=data["price"], size=sample_size)
  samp_mean = samp.mean()
  sample_means.append(samp_mean)
  sample_std = samp.std()
  margin_of_error = z_critical * (sample_std/math.sqrt(sample_size))
  confidence_interval = (samp_mean - margin_of_error, samp_mean + margin_of_error)
  intervals.append(confidence_interval)

print("Sample mean: ", sample_means)
print("Population Mean: ", p)
print("Intervals: ", intervals)

plt.errorbar(x=np.arange(0.1, 25, 1), y=sample_means, yerr=[(top-bot)/2 for top, bot in intervals], fmt='o')
print()
plt.hlines(xmin=0, xmax=25, y=p, linewidth=2.0, color="red")
print()
plt.title("Confidence Intervals for 25 Trials", fontsize=15)
plt.show()
```

OUTPUT:



```
Sample mean:  [3643.0, 3719.836, 3921.09, 3895.154, 3798.124, 3924.118, 3870.9, 3976.066, 3987.856
Population Mean:  3889.649087353617
Intervals:  [(3283.3050813524196, 4002.6949186475804), (3348.4280710541802, 4091.2439289458193), (
```

QUESTION -7:

Calculate the Correlation Coefficient using Pearson for the given table

DESCRIPTION:

Use pearsonr() function to calculate the Correlation Coefficient using Pearson

ALGORITHM:

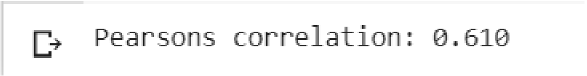
- From scipy.stats import pearsonr and from scipy.stats import spearmanr
- Use pearsonr() function to calculate the Correlation Coefficient using Pearson

PROGRAM:

from scipy.stats import pearsonr

```
from scipy.stats import spearmanr

import matplotlib.pyplot as plt

x=[150, 169, 175, 180, 200] #weight

y=[125, 130, 160, 169, 150] #blood pressure

corr, _ = pearsonr(x,y)

print("Pearsons correlation: %.3f" %corr)
```

OUTPUT:



```
Pearsons correlation: 0.610
```

QUESTION -8:

Calculate the Correlation Coefficient using Spearman for the given table

DESCRIPTION:

Spearman's rank correlation can be calculated in Python using the spearmanr() SciPy function. The function takes two real-valued samples as arguments and returns both the correlation coefficient in the range between -1 and 1 and the p-value for interpreting the significance of the coefficient.

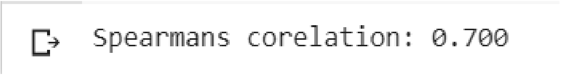
ALGORITHM:

- From scipy.stats import pearsonr and from scipy.stats import spearmanr
- Use spearmanr() function to calculate the Correlation Coefficient using spearman

PROGRAM:

```
corr, _ = spearmanr(x,y)

print("Spearmans corelation: %.3f" %corr)
```

OUTPUT:



```
Spearmans corelation: 0.700
```

QUESTION -9:

Calculate the Covariance Matrix for the given data and analyse it

DESCRIPTION:

Cov method is used to calculate the Covariance Matrix for the given data

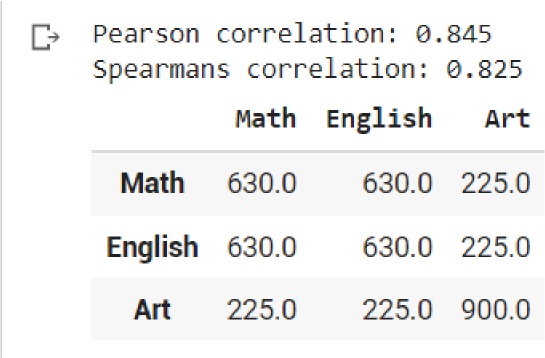
ALGORITHM:

- Use Cov method to calculate the covariance matrix.

PROGRAM:

```
x = pd. Series([90,90,60,60,30])

y = pd. Series([60,90,60,60,30])


p=x.corr(y, method="pearson")

s=x.corr(y, method='spearman')


print('Pearson correlation: %.3f' % p)
print('Spearmans correlation: %.3f' % s)

# relationship

df = pd.DataFrame({'Math': [90,90,60,60,30],'English':[90,90,60,60,30],'Art':[90,30,60,90,30]})


cov_matrix = df.cov()

cov_matrix
```

OUTPUT:



```
Pearson correlation: 0.845
Spearmans correlation: 0.825
```

	Math	English	Art
Math	630.0	630.0	225.0
English	630.0	630.0	225.0
Art	225.0	225.0	900.0

QUESTION -10:

Perform a hypothesis testing with Z-test The mean breaking strength of the cables supplied by a manufacture is 1800 with a S.D of 100. By a new technique in the manufacturing process, it is claimed that the breaking strength of the cable has increased. In order to test this claim, a sample of 50 cables is tested and it is found that the mean breaking strength is 1850. Can we support the claim at 1 % level?

DESCRIPTION:

A statistical hypothesis is an assumption about any aspect of a population. It could be the parameters of a distribution like mean of normal distribution, describing the population, the parameters of two or more populations, correlation or association between two or more characteristics of a population like age and height etc..

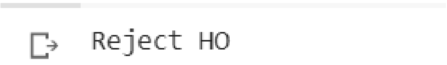
ALGORITHM:

- Identify the sample mean, standard deviation, population mean
- Calculate the Z if z is greater than the 2.33 then null hypothesis is rejected else null hypothesis is accepted

PROGRAM:

```
xbar=1
mu=50
n=1800
SD=100
z=abs(((xbar-mu)/(SD/math.sqrt(n))))
if(z>2.58):
  print("Reject HO")
else:
  print("Accept HO")
  print(z)
```

OUTPUT:



```
Reject HO
```

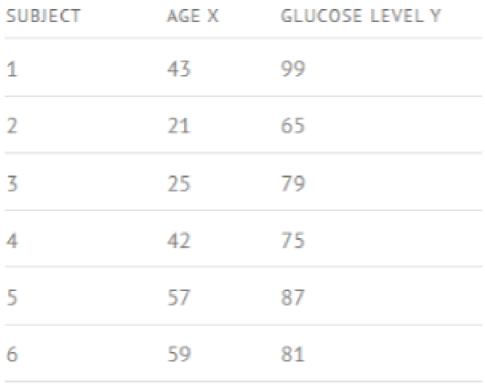
Result:

The programs to work on Statistical Inference are successful and the output is verified.

Ex. No.6	**Simple Linear Regression**
21/09/22	

QUESTION -1:

Develop the linear regression model for the given data.



SUBJECT	AGE X	GLUCOSE LEVEL Y
1	43	99
2	21	65
3	25	79
4	42	75
5	57	87
6	59	81

DESCRIPTION:

linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).

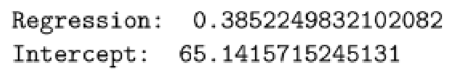
ALGORITHM:

- Calculating the sum for the x and y datas.
- Dividing the sum answers and considering it as regression
- Differentiating the mean with the regression and mean of the x values.
- Printing the Regression and the intercepts.

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
x=[43,21,25,42,57,59]
y=[99,65,79,75,87,81]
x = np.array(x)
y = np.array(y)
meanx = np.mean(x)
meany = np.mean(y)
xx = x-meanx
yy = y-meany
xy = xx * yy
xx2 = xx*xx
sumxy = sum(xy)
sumxx = sum(xx2)
regression = sumxy / sumxx
intercept = meany - (regression * meanx)
print("Regression: ", regression)
print("Intercept: ", intercept)
# b1 = regression , b0 = intercept
y_prediction = intercept + regression * x
```

OUTPUT:



```
Regression:  0.3852249832102082
Intercept:  65.1415715245131
```

QUESTION -1(a):

Calculate the intercept and regression coefficients in y=b0+xb1

DESCRIPTION:

The scatter() method in the matplotlib library is used to draw a scatter plot. Scatter plots are widely used to represent relation among variables and how change in one affects the other.

The plot() function is used to draw points (markers) in a diagram. By default, the plot() function draws a line from point to point. The function takes parameters for specifying points in the diagram. Parameter 1 is an array containing the points on the x-axis.

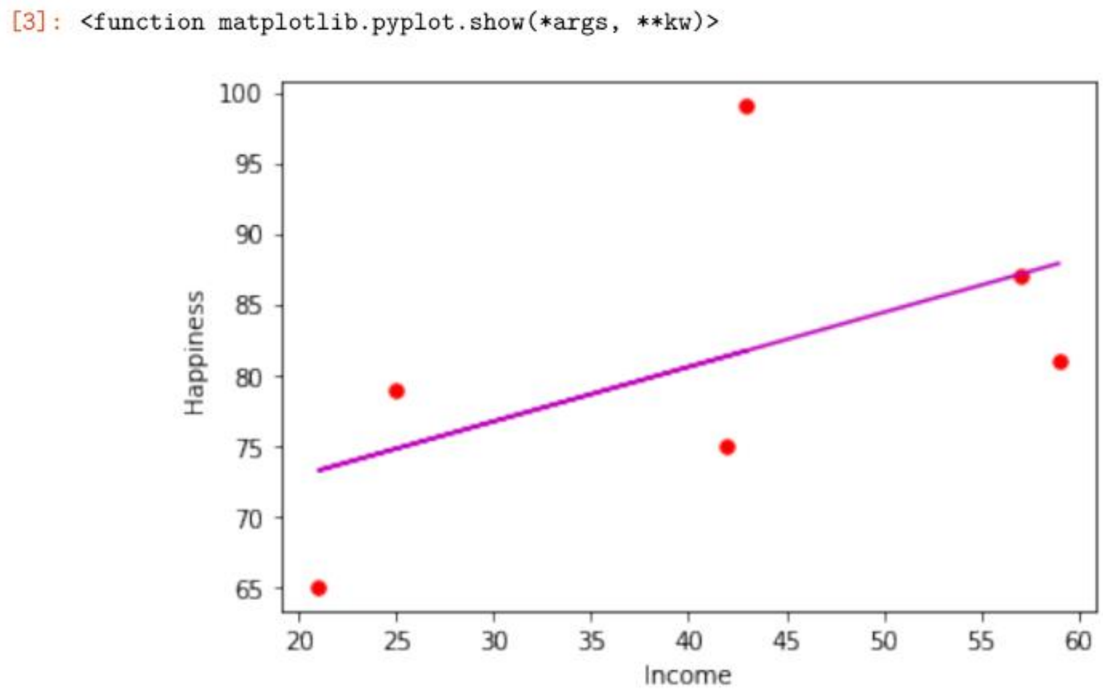
ALGORITHM:

- Use scatter to mark scatter plot.
- Use plot to make line plot.

PROGRAM:

```
plt.scatter(x, y, color="r", marker="o", s=30)

plt.plot(x, y_prediction, color="m")

plt.xlabel("Income")

plt.ylabel("Happiness")

plt.show
```

OUTPUT:



```
[3]: <function matplotlib.pyplot.show(*args, **kw)>
```

QUESTION -1(b):

Analyze the various performance metrics (Mean squared error, Mean Absolute Error, Root Mean Squared Error, and R-Squared)

DESCRIPTION:

Mean absolute error (MAE) is a loss function used for regression. Use MAE when you are doing regression and don't want outliers to play a big role. The loss is the mean over the absolute differences between true and predicted values, deviations in either direction from the true value are treated the same way.

The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss. Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.

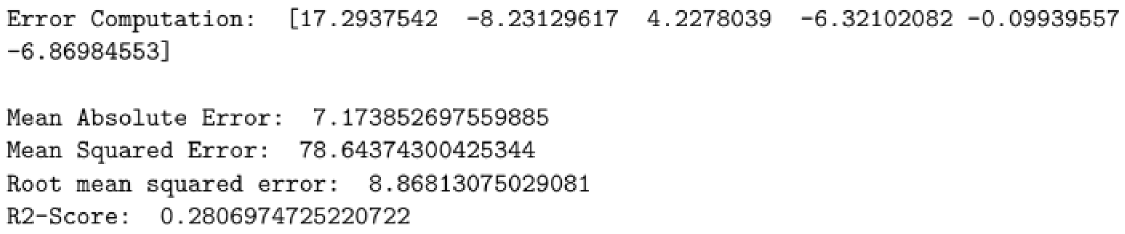
ALGORITHM:

- Declare err and calculate the y prediction
- Print the Mean absolute error using the mean absolute method in python
- Print the Mean squared error using the mean squared error in python
- Print the Root mean squared error using the math.sqrt method.

PROGRAM:

```
err = y-y_prediction
print("Error Computation: ", err)
print()
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import math
print("Mean Absolute Error: ", mean_absolute_error(y, y_prediction))
print("Mean Squared Error: ", mean_squared_error(y, y_prediction))
print("Root mean squared error: ", math.sqrt(mean_squared_error(y,y_prediction)))
print("R2-Score: ", r2_score(y, y_prediction))
```

OUTPUT:



```
Error Computation:  [17.2937542  -8.23129617  4.2278039  -6.32102082 -0.09939557
 -6.86984553]

Mean Absolute Error:  7.173852697559885
Mean Squared Error:  78.64374300425344
Root mean squared error:  8.86813075029081
R2-Score:  0.2806974725220722
```

QUESTION -2:

Develop the linear regression model for the income dataset using the scikit-learn

DESCRIPTION:

A simple way to store big data sets is to use CSV files (comma separated files). CSV files contains plain text and is a well know format that can be read by everyone including Pandas. In our examples we will be using a CSV file called 'data.csv'. Download data.csv.

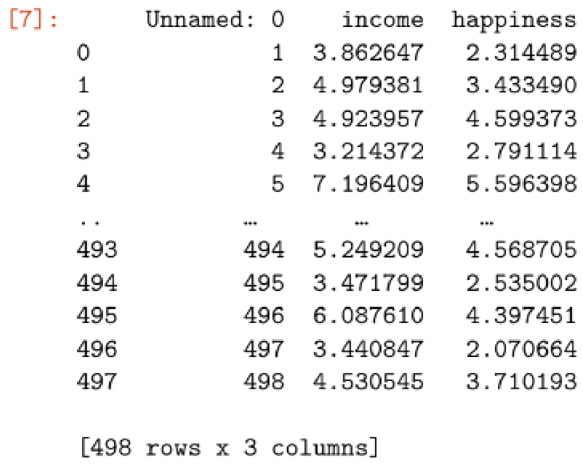
ALGORITHM:

- Import the pandas library
- Import the linear regression form the sklearn library
- Read the data using read_csv method

PROGRAM:

```
import pandas as pd

from sklearn.linear_model import LinearRegression

data = pd.read_csv("income-data.csv")

data
```

OUTPUT:



```
[7]:      Unnamed: 0    income  happiness
     0             1  3.862647   2.314489
     1             2  4.979381   3.433490
     2             3  4.923957   4.599373
     3             4  3.214372   2.791114
     4             5  7.196409   5.596398
     ..          ...       ...        ...
     493         494  5.249209   4.568705
     494         495  3.471799   2.535002
     495         496  6.087610   4.397451
     496         497  3.440847   2.070664
     497         498  4.530545   3.710193

     [498 rows x 3 columns]
```

QUESTION -2(a):

Divide the data into training (75%) and testing data (25%)

DESCRIPTION:

The train_test_split() method is used to split our data into train and test sets. First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X_train,X_test , y_train and y_test. X_train and y_train sets are used for training and fitting the model.

ALGORITHM:

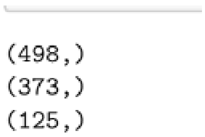
- Get the shape of the x and y data table.
- Import the train_test_split from the sklearn library
- Train the values.
- Print the shapes using the train model

PROGRAM:

```
data.head()
x = data['income']
y = data['happiness']
print(x.shape)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25,random_state=1)
print(X_train.shape)
print(X_test.shape)
```

```
from sklearn.linear_model import LinearRegression

import numpy as np

X_train=np.array(X_train).reshape(-1,1)

y_train=np.array(y_train).reshape(-1,1)
```

OUTPUT:



```
(498,)
(373,)
(125,)
```

QUESTION -2(b):

Analyze the impact of income to the happiness and display the intercept and regression coefficients.

DESCRIPTION:

model. fit() : fit training data. For supervised learning applications, this accepts two arguments: the data X and the labels y (e.g. model. fit(X, y)). For unsupervised learning applications, this accepts only a single argument, the data X

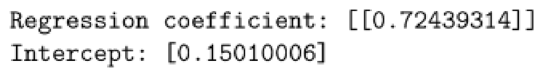
ALGORITHM:

- Create a model variable and import the LinearRegression() function
- Fit the model into the X train and the Y train
- Print the Regression coefficient and the intercept results.

PROGRAM:

```
model=LinearRegression()

model.fit(X_train,y_train)

print("Regression coefficient:",model.coef_)

print("Intercept:",model.intercept_)
```

OUTPUT:



```
Regression coefficient: [[0.72439314]]
Intercept: [0.15010006]
```

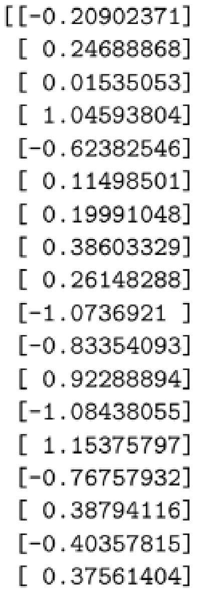
QUESTION -2(c):

Predict the y value (y') for the testing set (x)

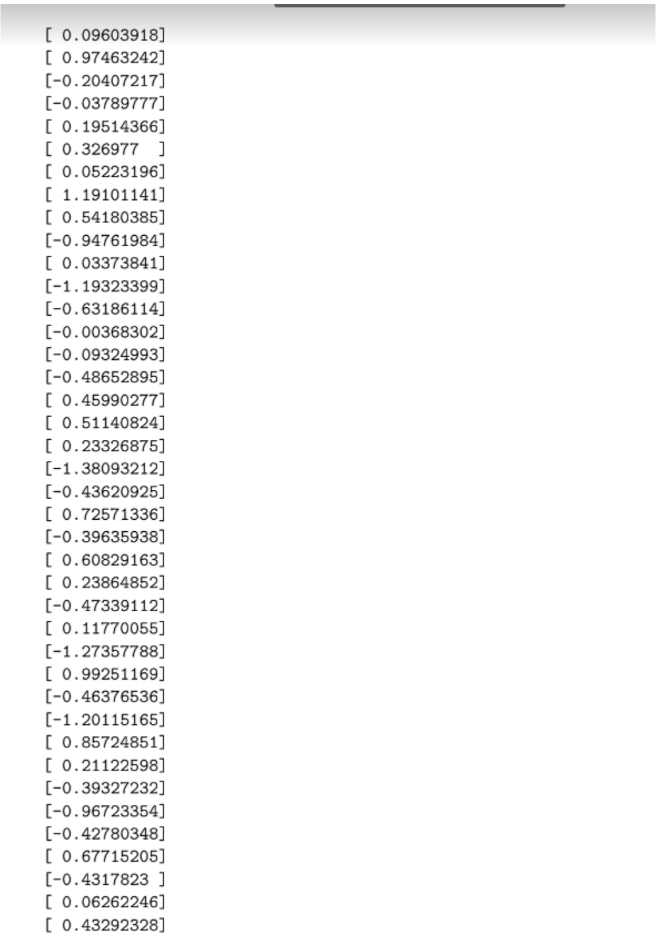
PROGRAM:

```
X_test=np.array(X_test).reshape(-1,1)

y_pred=model.predict(X_test)

y_test=np.array(y_test).reshape(-1,1)

err=y_test-y_pred

print(err)
```

OUTPUT:



```
[[-0.20902371]
 [ 0.24688868]
 [ 0.01535053]
 [ 1.04593804]
 [-0.62382546]
 [ 0.11498501]
 [ 0.19991048]
 [ 0.38603329]
 [ 0.26148288]
 [-1.0736921 ]
 [-0.83354093]
 [ 0.92288894]
 [-1.08438055]
 [ 1.15375797]
 [-0.76757932]
 [ 0.38794116]
 [-0.40357815]
 [ 0.37561404]
```



```
[ 0.09603918]
[ 0.97463242]
[-0.20407217]
[-0.03789777]
[ 0.19514366]
[ 0.326977  ]
[ 0.05223196]
[ 1.19101141]
[ 0.54180385]
[-0.94761984]
[ 0.03373841]
[-1.19323399]
[-0.63186114]
[-0.00368302]
[-0.09324993]
[-0.48652895]
[ 0.45990277]
[ 0.51140824]
[ 0.23326875]
[-1.38093212]
[-0.43620925]
[ 0.72571336]
[-0.39635938]
[ 0.60829163]
[ 0.23864852]
[-0.47339112]
[ 0.11770055]
[-1.27357788]
[ 0.99251169]
[-0.46376536]
[-1.20115165]
[ 0.85724851]
[ 0.21122598]
[-0.39327232]
[-0.96723354]
[-0.42780348]
[ 0.67715205]
[-0.4317823 ]
[ 0.06262246]
[ 0.43292328]
```

QUESTION -2(d):

plotting for prediction

DESCRIPTION:

A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

The plot() function is used to draw points (markers) in a diagram. By default, the plot() function draws a line from point to point. The function takes parameters for specifying points in the diagram. Parameter 1 is an array containing the points on the x-axis.

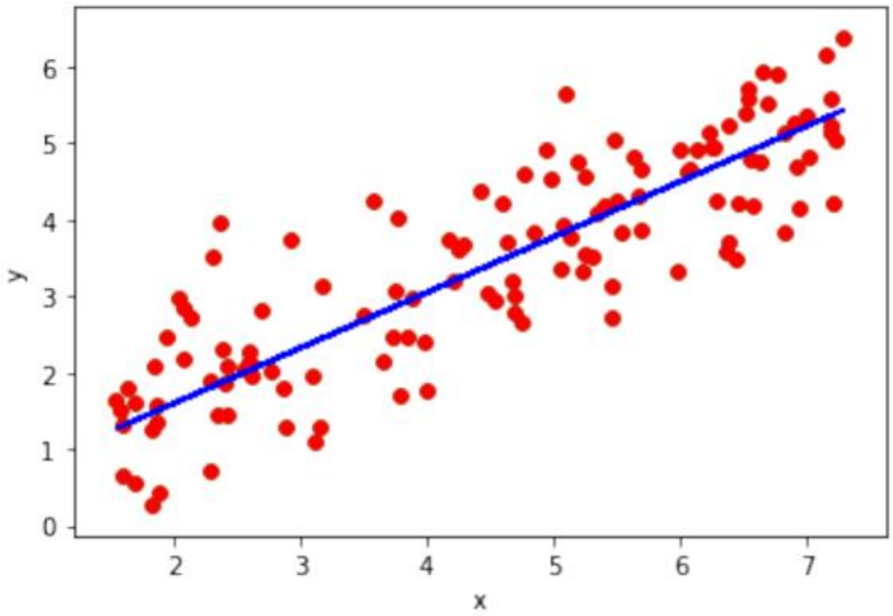
ALGORITHM:

- Import the matplot library
- Plot the scatter plot using the scatter method in pyhon. And plot the line plot using the plot method in python.
- Finally Show the plot details.

PROGRAM:

```
#plotting for prediction

import matplotlib.pyplot as plt

plt.scatter (X_test, y_test, color = "g", marker = "o", s=30)

# plotting the regression line

plt.scatter(X_test, y_test, color="r",marker="o", s=30)

plt.plot (X_test, y_pred, color="b")

plt.xlabel('x')
plt.ylabel('y')

plt.show()
```

OUTPUT:



QUESTION -2(e):

Analyse the performance metrics with the actual alue(y) and predict values, (y')

DESCRIPTION:

Mean absolute error (MAE) is a loss function used for regression. Use MAE when you are doing regression and don't want outliers to play a big role. The loss is the mean over the absolute differences between true and predicted values, deviations in either direction from the true value are treated the same way.

The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss. Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.

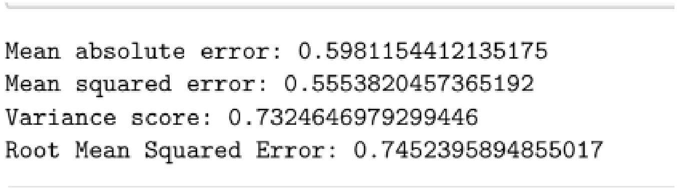
ALGORITHM:

Printing the Mean absolute error, Mean squared error, and variance error using the method.

PROGRAM:

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import math
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print("Mean squared error:", mean_squared_error(y_test, y_pred))
print('Variance score:' ,r2_score(y_test, y_pred))
print('Root Mean Squared Error:',math.sqrt(mean_squared_error(y_test, y_pred)))
```

OUTPUT:



```
Mean absolute error: 0.5981154412135175
Mean squared error: 0.5553820457365192
Variance score: 0.7324646979299446
Root Mean Squared Error: 0.7452395894855017
```

Result:

Therefore, the Simple linear regression is verified for the different programs.

Ex. No.7	**Performance analysis on KNN classification technique**
28/09/22	

AIM:

To work with a data set to create a performance analysis on KNN classification technique.

Dataset: cancer.csv

QUESTION -1(a):

Develop a KNN classification mode of the cancer dataset using the scikit-learn

Use the columns: ‘radius_mean’, ‘texture_mean’, ‘perimeter_mean’, ‘area_mean’, ‘smoothness_mean’, ‘compactness_mean’, ‘concavity_mean’, ‘concave_points_mean’, ‘symmetry_mean’, ‘fractal_dimension_mean’ as the independent variables.

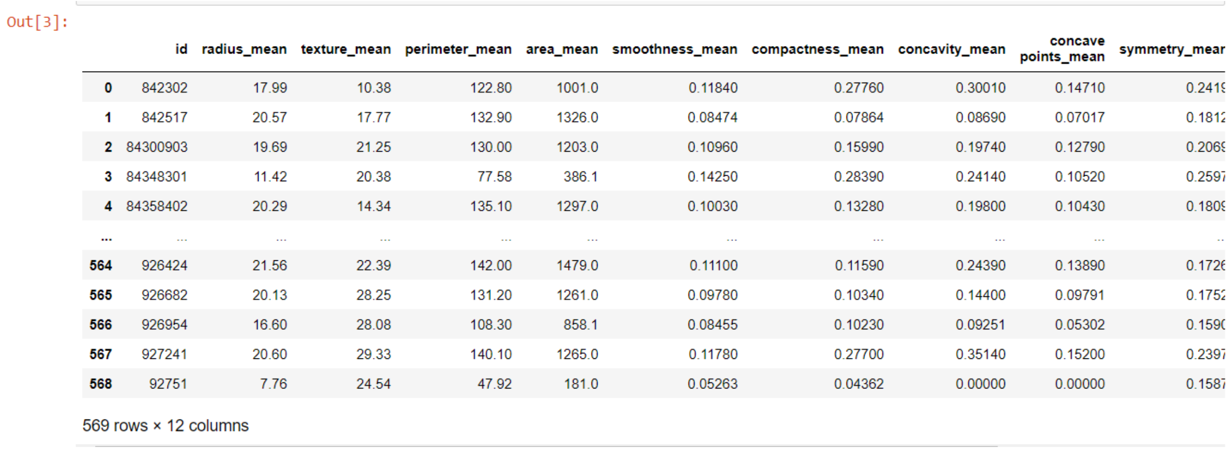
ALGORITHM:

- Importing the required libraries of pandas, numpy, matplotlib, sklearn and scipy libraries to perform an analysis on KNN clustering technique.
- Define the specific columns using the iloc command to declare the start rows and end rows of the particular columns.
- Print the head results after taking the columns as an output.

PROGRAM:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.metrics import recall_score

from sklearn.metrics import precision_score

from sklearn.metrics import f1_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import roc_curve

from sklearn.metrics import auc

from scipy import stats

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

df


x=df.iloc[:,0:11]

x.head()
```

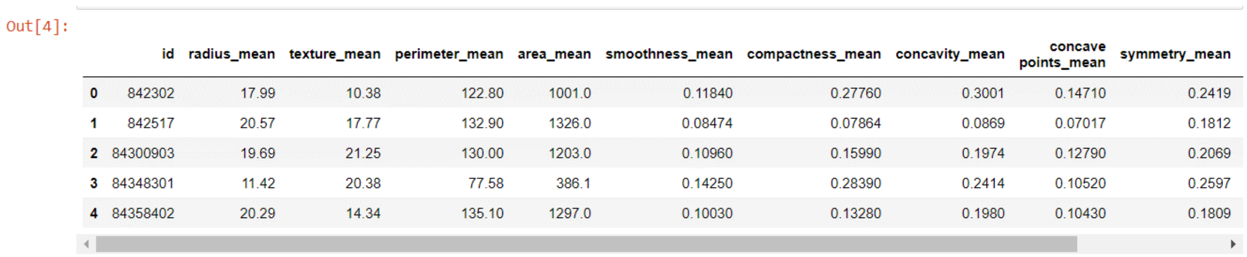
OUTPUT:



Out[3]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mear
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...	..
564	926424	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	926682	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	926954	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	927241	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	92751	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 12 columns



Out[4]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

QUESTION -1(b):

Use the target variable as ‘diagnosis’ (Malignant – M, Benign – B)

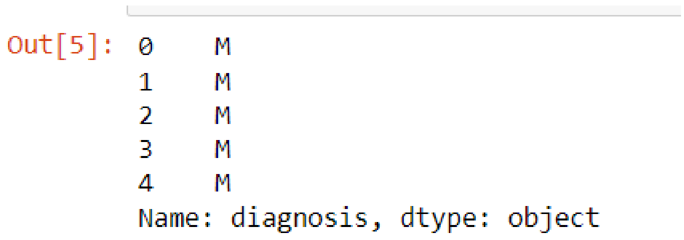
ALGORITHM:

- To use the target variable as diagnosis in (Malignant – M and Benign – B)
- Define a variable as y and add an iloc command as follows.
- Inside the square bracket, initialize the start values and the end values, Here to get the diagnosis column, Initialize the start value as empty and end value as 11
- Print the head value to get the correct output.

PROGRAM:

```
y=df.iloc[:,11]

y.head()
```

OUTPUT:



```
Out[5]: 0    M
        1    M
        2    M
        3    M
        4    M
        Name: diagnosis, dtype: object
```

QUESTION -1(c):

Encode the categorical value of the target column to numerical value.

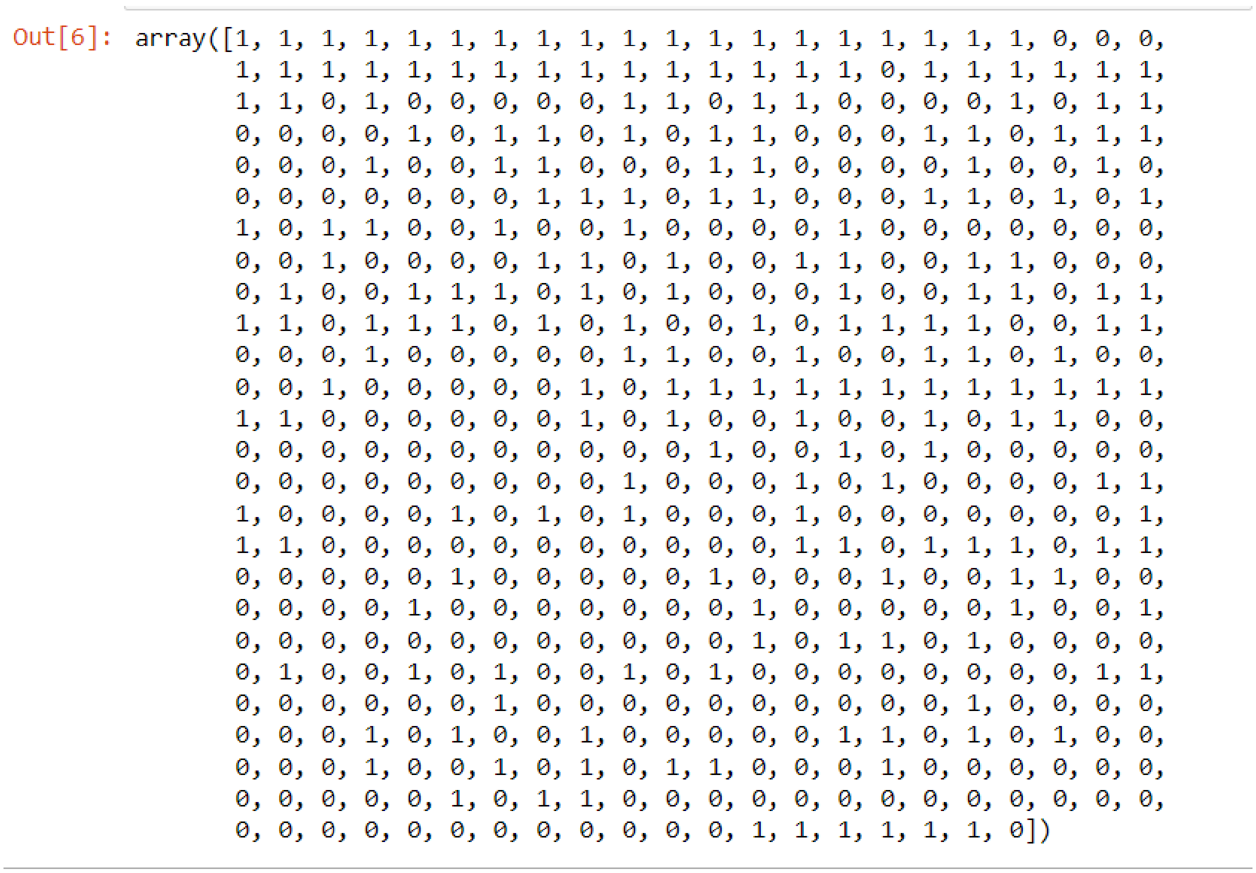
ALGORITHM:

- To encode the categorical value of the target column to numerical value.
- Declare a variable le and initialize a LabelEncoder() inbuild function.
- Fit the transformation to the y axis
- Print the y axis to get the array results.

PROGRAM:

```
le=LabelEncoder()

y=le.fit_transform(y)

y
```

OUTPUT:



```
Out[6]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
               0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
               0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
               1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
               0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
               1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
               0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
               0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
               1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
               0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
               0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
               0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
               0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
               0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
               0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0])
```

QUESTION -1(d):

Divide the data into training (75%) and testing set (25%)

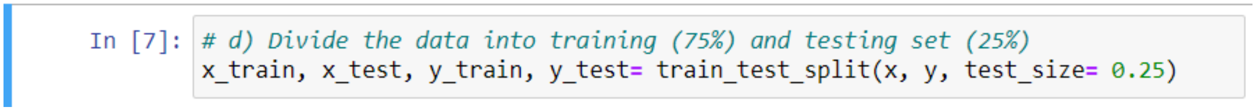
ALGORITHM:

- To divide the data into 75% and 25% create three variables.
- First create three variables for train and test for the two axis and using the split function, split the values into 75% and 25%

PROGRAM:

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25)

OUTPUT:



```
In [7]: # d) Divide the data into training (75%) and testing set (25%)
        x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25)
```

QUESTION -1(e):

Perform the classification with K=3

ALGORITHM:

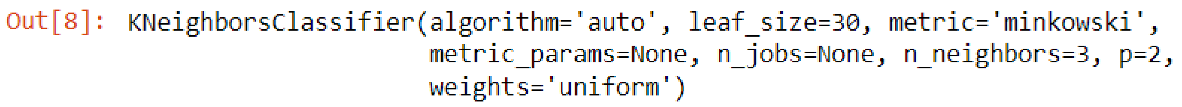
- To perform the classification with the K value as 3
- Initialize a variable as KNN and assign the KNeighboursClassifier function to it.
- Use the fit function to fit the train values of x and y to print the results.

PROGRAM:

knn=KNeighborsClassifier(n_neighbors=3)

knn.fit(x_train,y_train)

OUTPUT:



```
Out[8]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                             weights='uniform')
```

QUESTION -1(f):

Analyse the performance of the classifier with various performance measures such as confusion matrix, accuracy, recall, precision, specificity, f-score, Receiver operating characteristic (ROC) curve and Area Under Curve (AUC) score.

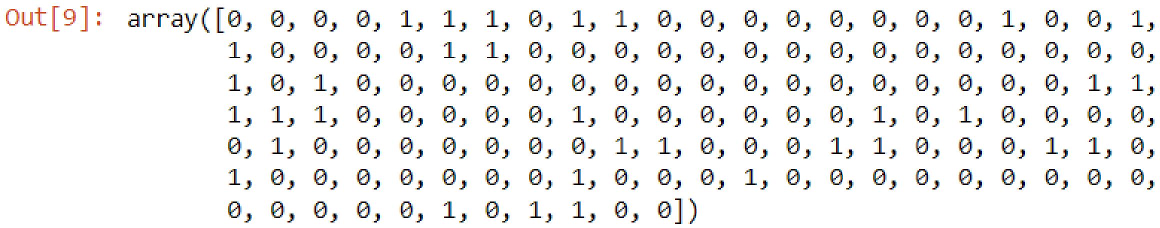
ALGORITHM:

- Create different variables to derive different functions.
- Create a variable y prediction to predict the text that defined previously.
- Declare the confusion matrix and to find the y test and y prediction results.
- Print the confusion matrix as an array.

PROGRAM:

```
y_pred=knn.predict(x_test)

y_pred


conf_matrix=confusion_matrix(y_test,y_pred)

cm=conf_matrix

conf_matrix
```

OUTPUT:



```
Out[9]: array([0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
               1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
               1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
               0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
               1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0])
```

QUESTION -1(g):

Perform feature scaling on independent variables and analyse the performance.

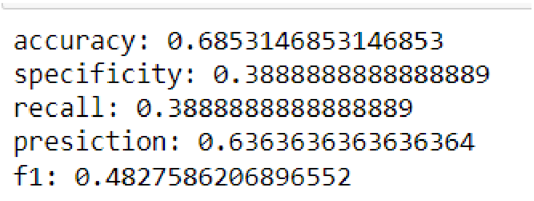
ALGORITHM:

- Finding the accuracy score using the function accuracy_score with the y test and y prediction values as a parameter.
- Print the accuracy, specificity, recall, precision, and the f scores

PROGRAM:

```
accc=accuracy_score(y_test,y_pred)

fsc=f1_score(y_test,y_pred)

print("accuracy:",accuracy_score(y_test,y_pred))

print("specificity:",recall_score(y_test,y_pred))

print("recall:",recall_score(y_test,y_pred))

print("presiction:",precision_score(y_test,y_pred))

print("f1:",f1_score(y_test,y_pred))
```

OUTPUT:



```
accuracy: 0.6853146853146853
specificity: 0.3888888888888889
recall: 0.3888888888888889
presiction: 0.6363636363636364
f1: 0.4827586206896552
```

QUESTION -1(h):

Change the value of K in KNN with 5,7,9,11 and tabulate the various TP, TN, accuracy, f-score, and AUC score obtained.

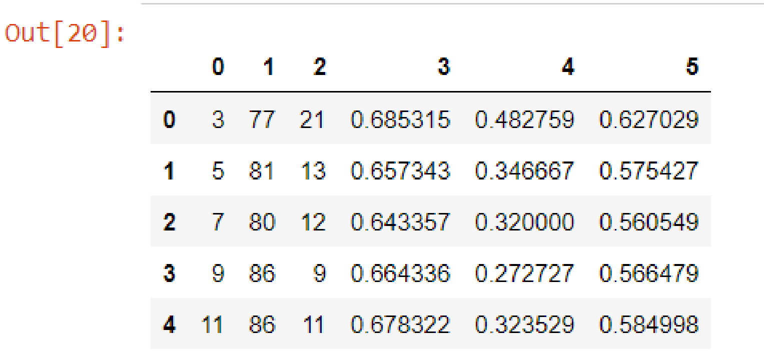
ALGORITHM:

- Using for loop to loop over the data set to find the K nearest neighbour classification.
- Define a variable for y prediction, confusion matrix, f score, accuracy and the ftn to get the values.
- For each iteration the values will be printed into the dataframe as a results.
- Once the results is calculated, then it will be printed into a dataframe.

PROGRAM:

```
for i in range(3,12,2):
 knn=KNeighborsClassifier(n_neighbors=i)
 knn.fit(x_train,y_train)
 y_pred=knn.predict(x_test)
 conf_matrix=confusion_matrix(y_test,y_pred)
 cm=conf_matrix
 accc=accuracy_score(y_test,y_pred)
 fsc=f1_score(y_test,y_pred)
 auc1,auc2,thresholds=roc_curve(y_test,y_pred)
 auccc=auc(auc1,auc2)
 ins=[i,cm[0][0],cm[1][1],accc,fsc,auccc]
 ftn.append(ins)
ftdn=pd.DataFrame(ftn)
ftdn
```

OUTPUT:



Out[20]:

	0	1	2	3	4	5
0	3	77	21	0.685315	0.482759	0.627029
1	5	81	13	0.657343	0.346667	0.575427
2	7	80	12	0.643357	0.320000	0.560549
3	9	86	9	0.664336	0.272727	0.566479
4	11	86	11	0.678322	0.323529	0.584998

Result:

Therefore, the performance analysis on KNN classification technique is verified and obtained the required output.

Ex. No.8	**Performance analysis on Decision Tree classification technique**
12/10/2022	

AIM:

To work with a data set to create a performance analysis on Decision tree classification technique.

Dataset: cancer.csv

QUESTION -1(a):

Develop a KNN classification mode of the cancer dataset using the scikit-learn

Use the columns: ‘radius_mean’, ‘texture_mean’, ‘perimeter_mean’, ‘area_mean’, ‘smoothness_mean’, ‘compactness_mean’, ‘concavity_mean’, ‘concave_points_mean’, ‘symmetry_mean’, ‘fractal_dimension_mean’ as the independent variables.

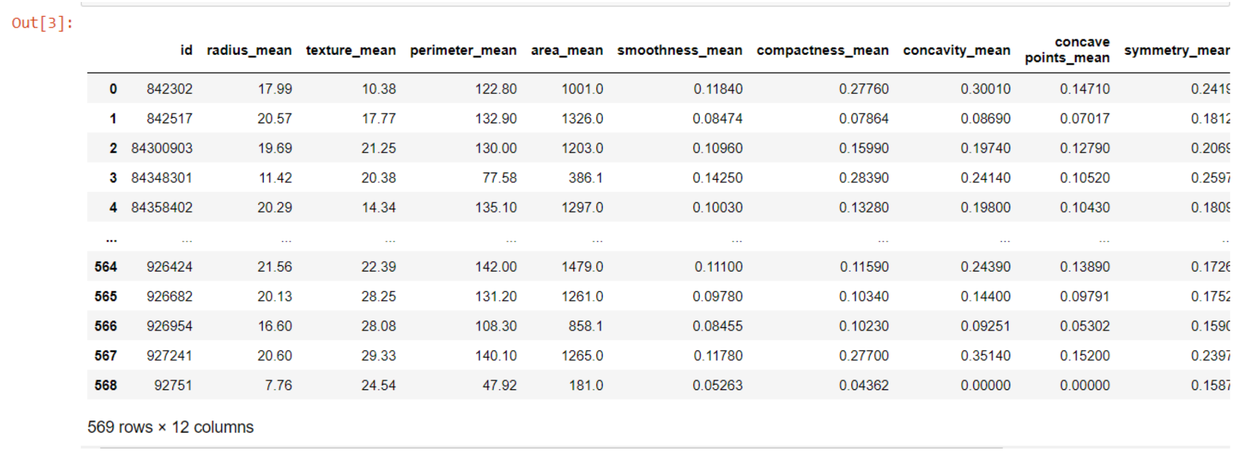
ALGORITHM:

- Importing the required libraries of pandas, numpy, matplotlib, sklearn and scipy libraries to perform an analysis on KNN clustering technique.
- Define the specific columns using the iloc command to declare the start rows and end rows of the particular columns.
- Print the head results after taking the columns as an output.

PROGRAM:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import tree
from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.metrics import recall_score

from sklearn.metrics import precision_score

from sklearn.metrics import f1_score

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import roc_curve

from sklearn.metrics import auc

from scipy import stats

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

df


x=df.iloc[:,0:11]

x.head()
```

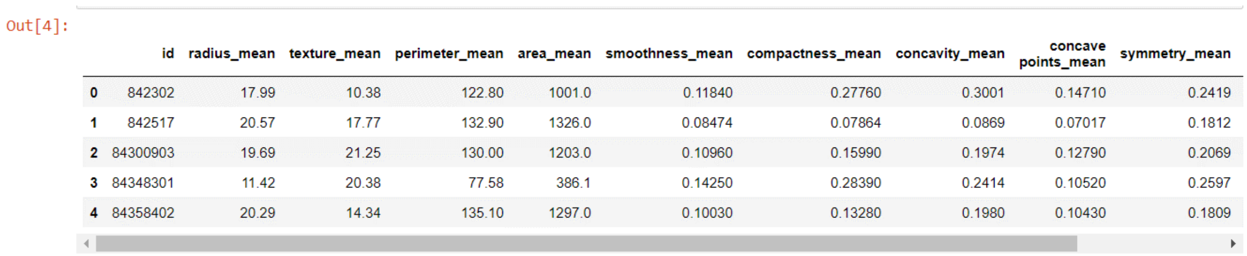
OUTPUT:



Out[3]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...	...
564	926424	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	926682	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	926954	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	927241	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	92751	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 12 columns



Out[4]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

QUESTION -1(b):

Use the target variable as ‘diagnosis’ (Malignant – M, Benign – B)

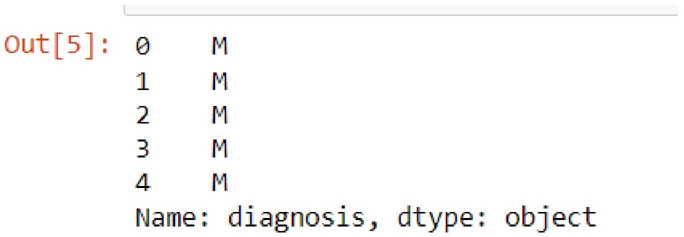
ALGORITHM:

- To use the target variable as diagnosis in (Malignant – M and Benign – B)
- Define a variable as y and add an iloc command as follows.
- Inside the square bracket, initialize the start values and the end values, Here to get the diagnosis column, Initialize the start value as empty and end value as 11
- Print the head value to get the correct output.

PROGRAM:

```
y=df.iloc[:,11]

y.head()
```

OUTPUT:



```
Out[5]: 0    M
        1    M
        2    M
        3    M
        4    M
        Name: diagnosis, dtype: object
```

QUESTION -1(c):

Encode the categorical value of the target column to numerical value.

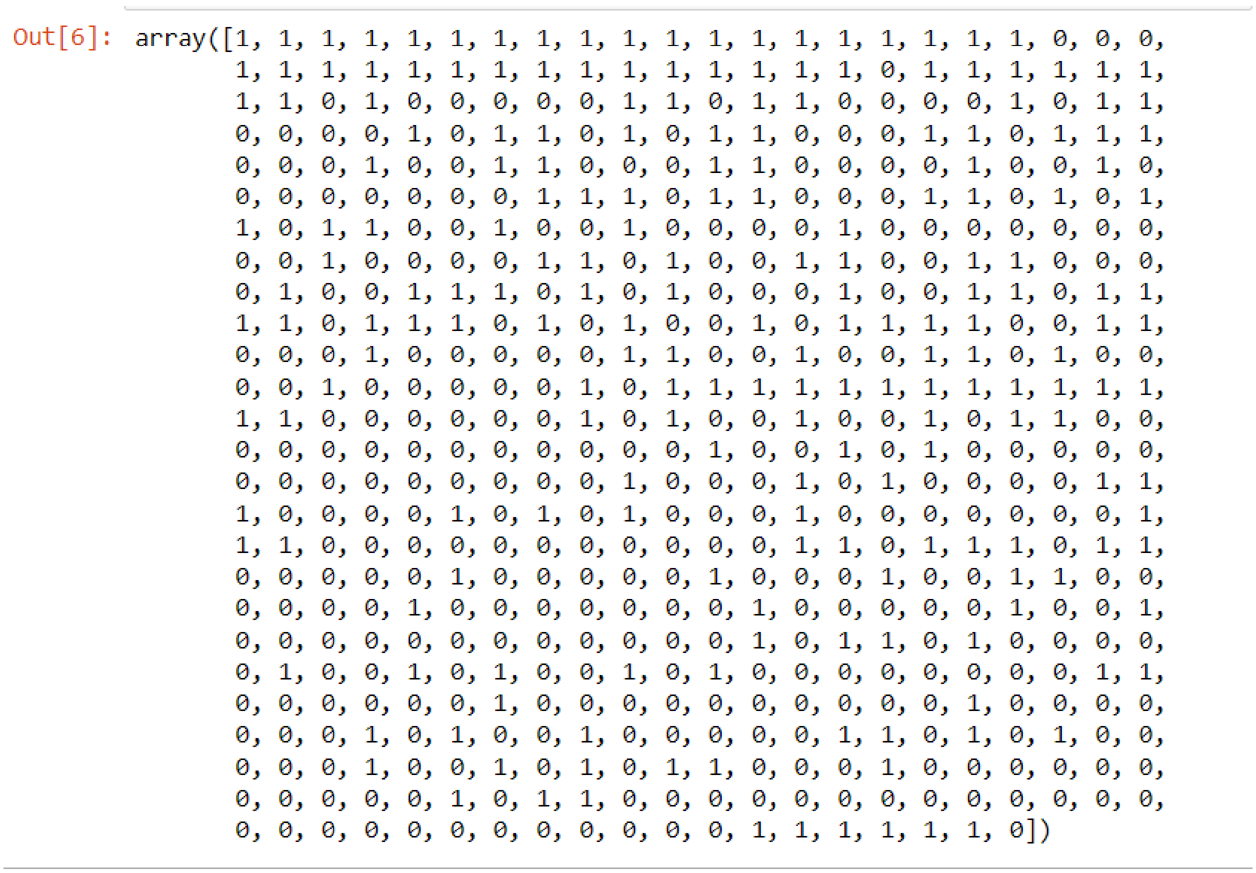
ALGORITHM:

- To encode the categorical value of the target column to numerical value.
- Declare a variable le and initialize a LabelEncoder() inbuild function.
- Fit the transformation to the y axis
- Print the y axis to get the array results.

PROGRAM:

```
le=LabelEncoder()

y=le.fit_transform(y)

y
```

OUTPUT:



```
Out[6]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
               0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
               0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
               1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
               0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
               1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
               0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
               0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
               1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
               0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
               0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,
               0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
               0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
               0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
               0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0])
```

QUESTION -1(d):

Divide the data into training (75%) and testing set (25%)

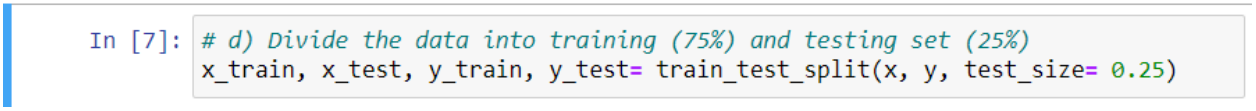
ALGORITHM:

- To divide the data into 75% and 25% create three variables.
- First create three variables for train and test for the two axis and using the split function, split the values into 75% and 25%

PROGRAM:

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25)

OUTPUT:



```
In [7]: # d) Divide the data into training (75%) and testing set (25%)
        x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25)
```

QUESTION -1(e):

Analyse the performance of the classifier with various performance measures ⎵ ,suchasconfusion matrix, accuracy, recall, precision, specificity, f-score, Receiver operatingcharacteristic (ROC) curve,and Area Under Curve (AUC) score.

ALGORITHM:

- To perform the classification with the K value as 3
- Initialize a variable as KNN and assign the KNeighboursClassifier function to it.
- Use the fit function to fit the train values of x and y to print the results.

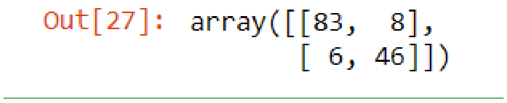
PROGRAM:

conf_matrix=confusion_matrix(y_test,y_pred)

cm=conf_matrix

conf_matrix

OUTPUT:



```
Out[27]: array([[83,  8],
               [ 6, 46]])
```

QUESTION -1(f):

Analyse the performance of the classifier with various performance measures such as confusion matrix, accuracy, recall, precision, specificity, f-score, Receiver operating characteristic (ROC) curve and Area Under Curve (AUC) score.

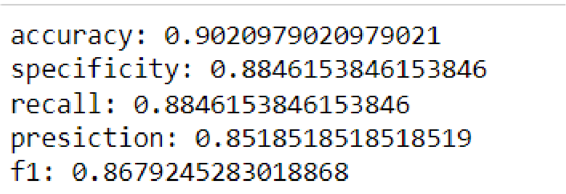
ALGORITHM:

- Create different variables to derive different functions.
- Create a variable y prediction to predict the text that defined previously.
- Declare the confusion matrix and to find the y test and y prediction results.
- Print the confusion matrix as an array.

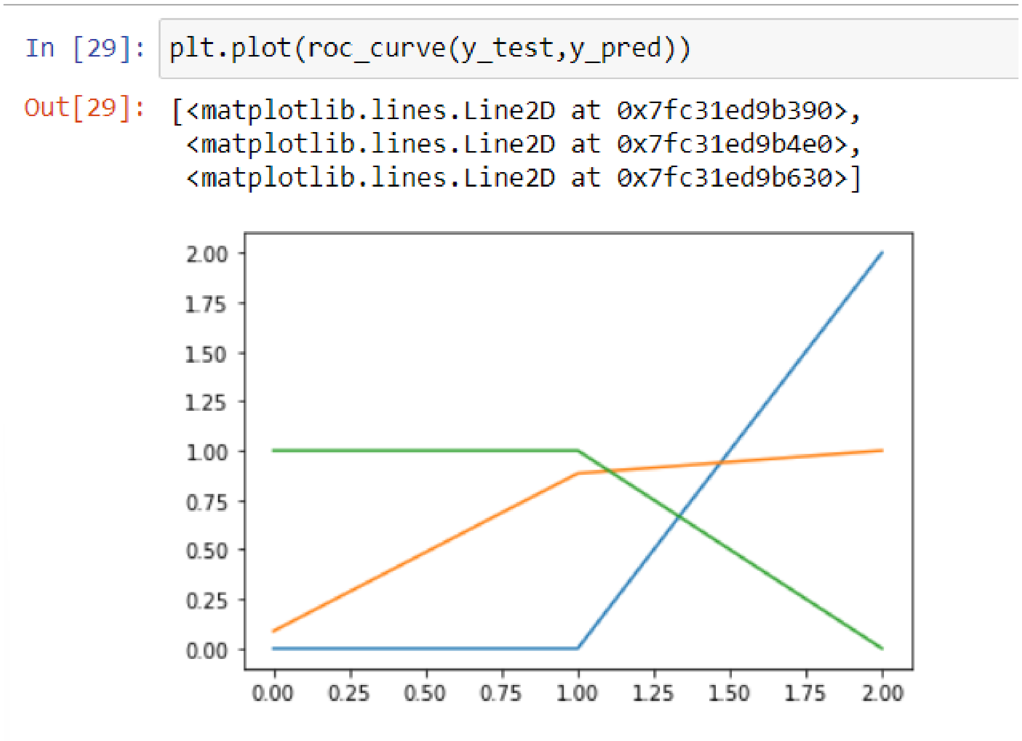
PROGRAM:

```
accc=accuracy_score(y_test,y_pred)

fsc=f1_score(y_test,y_pred)

print("accuracy:",accuracy_score(y_test,y_pred))

print("specificity:",recall_score(y_test,y_pred))

print("recall:",recall_score(y_test,y_pred))

print("presiction:",precision_score(y_test,y_pred))

print("f1:",f1_score(y_test,y_pred))
```

OUTPUT:



```
accuracy: 0.9020979020979021
specificity: 0.8846153846153846
recall: 0.8846153846153846
presiction: 0.8518518518518519
f1: 0.8679245283018868
```



```
In [29]: plt.plot(roc_curve(y_test,y_pred))

Out[29]: [<matplotlib.lines.Line2D at 0x7fc31ed9b390>,
          <matplotlib.lines.Line2D at 0x7fc31ed9b4e0>,
          <matplotlib.lines.Line2D at 0x7fc31ed9b630>]
```

QUESTION -1(g):

Display the constructed decision tree.

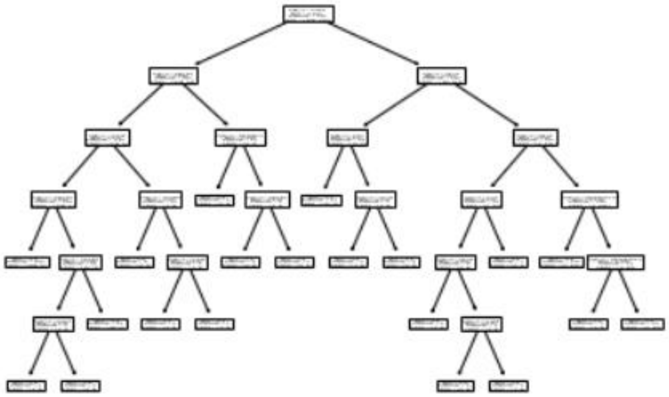
ALGORITHM:

- Finding the accuracy score using the function accuracy_score with the y test and y prediction values as a parameter.
- Print the accuracy, specificity, recall, precision, and the f scores and print it as a decision tree.

PROGRAM:

tree.plot_tree(clf);

OUTPUT:



QUESTION -1(h):

Prune the tree with maximum depth as 3,5,7 and tabulate the various TP, TN,accuracy, f-score and AUC score obtained.

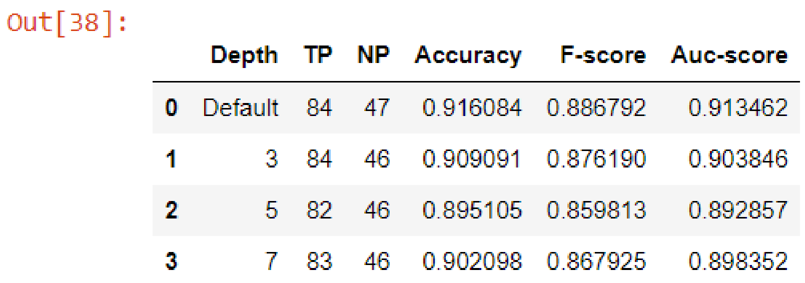
ALGORITHM:

- Using for loop to loop over the data set to find the Decision tree classification.
- Define a variable for y prediction, confusion matrix, f score, accuracy and the ftn to get the values.
- For each iteration the values will be printed into the dataframe as a results.
- Once the results is calculated, then it will be printed into a dataframe.

PROGRAM:

```
for i in range(1,8,2):

 if i==1:
  clf=tree.DecisionTreeClassifier()
 else:
  clf=tree.DecisionTreeClassifier(max_depth=i)
 clf.fit(x_train,y_train)
 y_pred=clf.predict(x_test)
 conf_matrix=confusion_matrix(y_test,y_pred)
 cm=conf_matrix
 accc=accuracy_score(y_test,y_pred)
 fsc=f1_score(y_test,y_pred)
 auc1,auc2,thresholds=roc_curve(y_test,y_pred)
 auccc=auc(auc1,auc2)
 if i==1:
  ins=['Default',cm[0][0],cm[1][1],accc,fsc,auccc]
 else:
  ins=[i,cm[0][0],cm[1][1],accc,fsc,auccc]
 ftclfs.append(ins)
ftdn=pd.DataFrame(ftclfs,columns=['Depth','TP','NP','Accuracy','F-score','Auc-score'])
ftdn
```

OUTPUT:



Out[38]:

	Depth	TP	NP	Accuracy	F-score	Auc-score
0	Default	84	47	0.916084	0.886792	0.913462
1	3	84	46	0.909091	0.876190	0.903846
2	5	82	46	0.895105	0.859813	0.892857
3	7	83	46	0.902098	0.867925	0.898352

Result:

Therefore, the performance analysis on KNN classification technique is verified and obtained the required output.

Ex. No.9	Clustering of Data using K-means Clustering Technique
19/10/2022	

AIM:

To work with a data set to create a performance analysis on Decision tree classification technique.

Dataset: cancer.csv

QUESTION -1(a):

Develop a K-Means clustering model for the Iris dataset using the scikit learn library.

Use the columns: ‘SepalLengthCm’, ‘SepalWidthCm’, ‘PetalLengthCm’, ‘PetalWidthCm’, as the input variables

ALGORITHM:

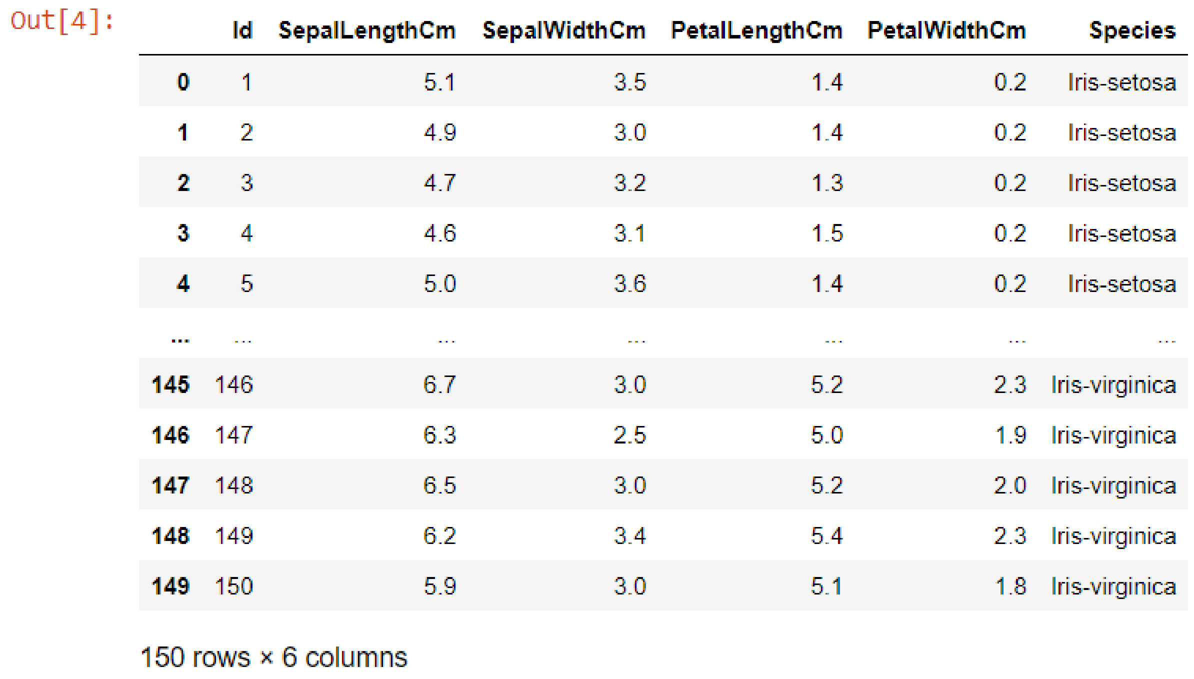
- Import the librarys pandas to read csv file, matplot lib to visualize the datas and sklearn cluster library to perform the clustering operations.
- Use the columns given in the dataset, from the third column to the last column.
- Display the dataset after importing.

PROGRAM:

```
import pandas as pd

import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans

data = pd.read_csv("Iris.csv")

data



X = data.iloc[:, [3,4]].values

wcss_list = []
```

OUTPUT:



Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

QUESTION -1(b):

Compute the optimal number of clusters ‘K’ with Elbow method.

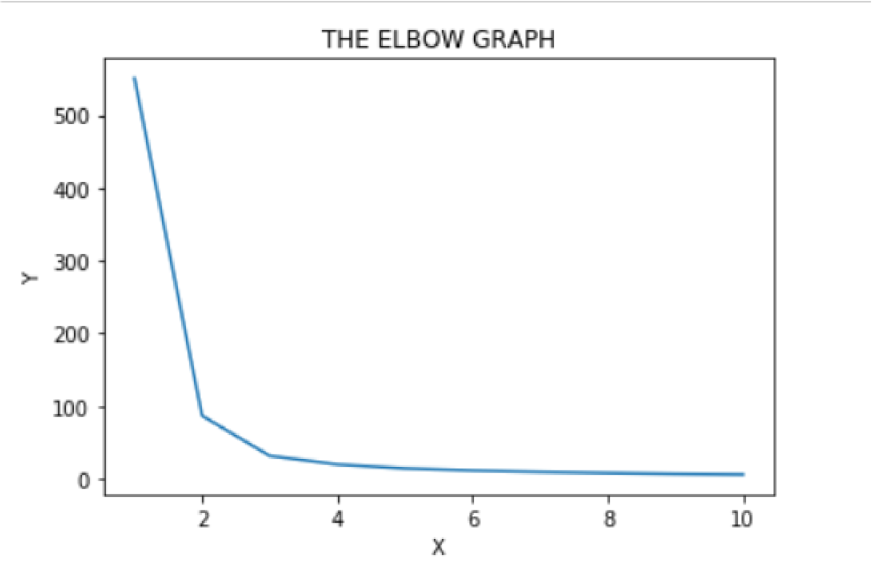
ALGORITHM:

- Use for loop to fetch from the first column to the last column.
- Declare the K means with the parameter of n_clusters, k-means initialization and the random state.
- Fit the K means columns into the datas.
- Plot the details from range 1 to 11
- Plot the title, x label and y label, finally, Show the visualized data.

PROGRAM:

```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(X)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title("THE ELBOW GRAPH")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

OUTPUT:



QUESTION -1(c):

Visualize the data representation of K-means clustering.

ALGORITHM:

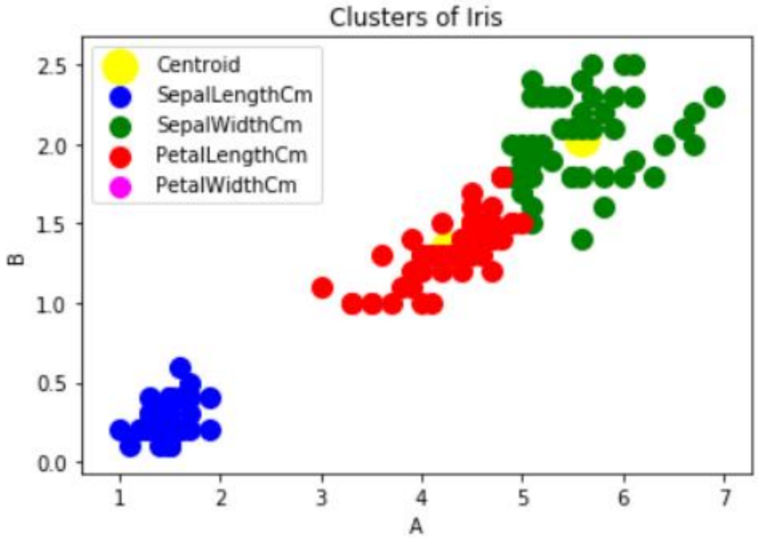
- Using the matplot library to plot the data's in graphical format.

PROGRAM:

```
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=300, c='yellow',
label='Centroid')


plt.scatter(X[y_predict == 0, 0], X[y_predict == 0, 1], s=100, c='blue',

label='SepalLengthCm')
```

```
plt.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s=100, c='green',
label='SepalWidthCm')

plt.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=100, c='red', label='PetalLengthCm')

plt.scatter(X[y_predict == 3, 0], X[y_predict == 3, 1], s=100, c='magenta',
label='PetalWidthCm')

plt.title('Clusters of Iris')
plt.xlabel('A')
plt.ylabel('B')
plt.legend()
plt.show()
```

OUTPUT:



QUESTION – 1(d):

Display the cluster centeroids

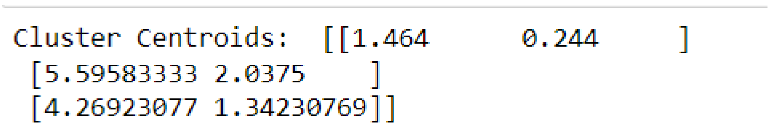
ALGORITHM:

- To cacluate the cluster centroids of the dataset, use the cluster_centers_ from the kmeans library.
- Print the cluster centeroids values as a calculated output.

PROGRAM:

```
centers = kmeans.cluster_centers_

print("Cluster Centroids: ", centers)
```

OUTPUT:



```
Cluster Centroids:  [[1.464      0.244     ]
 [5.59583333 2.0375    ]
 [4.26923077 1.34230769]]
```

QUESTION -1(e):

Change the value of K in K-means with different values and tabulate the silhouette_score and davies_bouldin_score obtained.

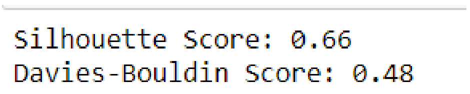
ALGORITHM:

- Import the silhouette_score and davies_bouldin_score from the sklearn.metrics library.
- Define the variable of s_score, d_score and calculate the s_score with the parameter of x, kmeans.labels_ and the metric as Euclidean
- Print the silhouette score and the davies-bouldin score result as a required output.

PROGRAM:

```
from sklearn.metrics import silhouette_score

from sklearn.metrics import davies_bouldin_score

s_score = silhouette_score(X, kmeans.labels_, metric='euclidean')

d_score = davies_bouldin_score(X, kmeans.labels_)

print("Silhouette Score: %.2f" %s_score)

print("Davies-Bouldin Score: %.2f" %d_score)
```

OUTPUT:



```
Silhouette Score: 0.66
Davies-Bouldin Score: 0.48
```

Result:

Therefore, the performance analysis on KNN classification technique is verified and obtained the required output.

Ex. No.10	Design of Content-based Recommender system
26/10/2022	

AIM:

To design a udemy course recommender system with the content-based recommendation using the scikit-learn libaray.

QUESTION -1(a):

Use the column: ‘course_title’

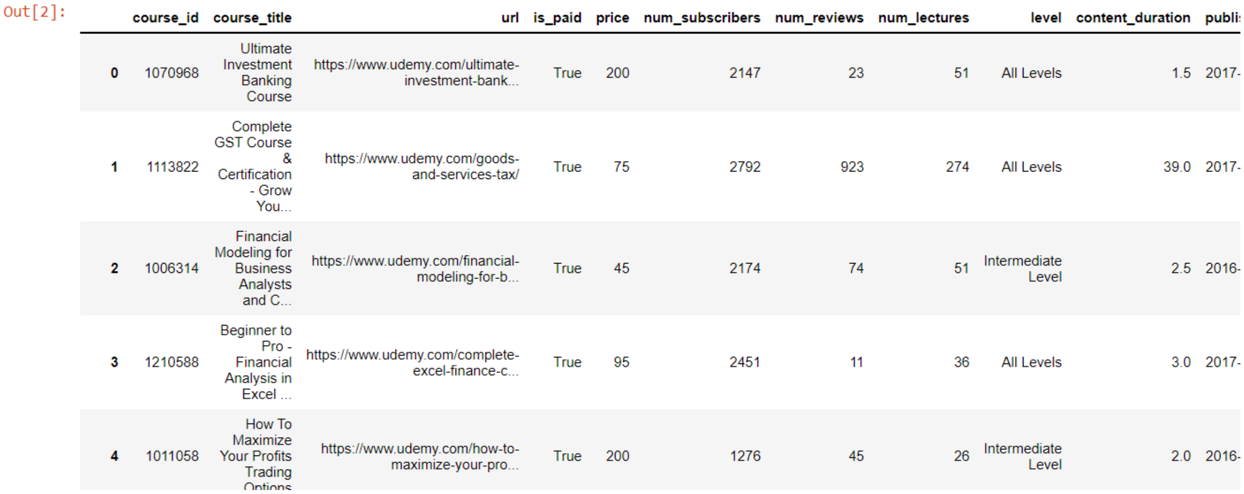
ALGORITHM:

- Import the libraries as pandas, numpy and matplot
- Read the dataset as a dataframe and print the data frame.
- Use the column as course_title using the iloc command.

PROGRAM:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

data = pd.read_csv('udemy_courses.csv')

data
```

OUTPUT:



Out[2]:

	course_id	course_title	url	is_paid	price	num_subscribers	num_reviews	num_lectures	level	content_duration	publi:
0	1070968	Ultimate Investment Banking Course	https://www.udemy.com/ultimate-investment-bank...	True	200	2147	23	51	All Levels	1.5	2017-
1	1113822	Complete GST Course & Certification - Grow You...	https://www.udemy.com/goods-and-services-tax/	True	75	2792	923	274	All Levels	39.0	2017-
2	1006314	Financial Modeling for Business Analysts and C...	https://www.udemy.com/financial-modeling-for-b...	True	45	2174	74	51	Intermediate Level	2.5	2016-
3	1210588	Beginner to Pro - Financial Analysis in Excel ...	https://www.udemy.com/complete-excel-finance-c...	True	95	2451	11	36	All Levels	3.0	2017-
4	1011058	How To Maximize Your Profits Trading Options	https://www.udemy.com/how-to-maximize-your-pro...	True	200	1276	45	26	Intermediate Level	2.0	2016-

QUESTION -1(b):

Remove the leading and trailing whitespaces in that column.

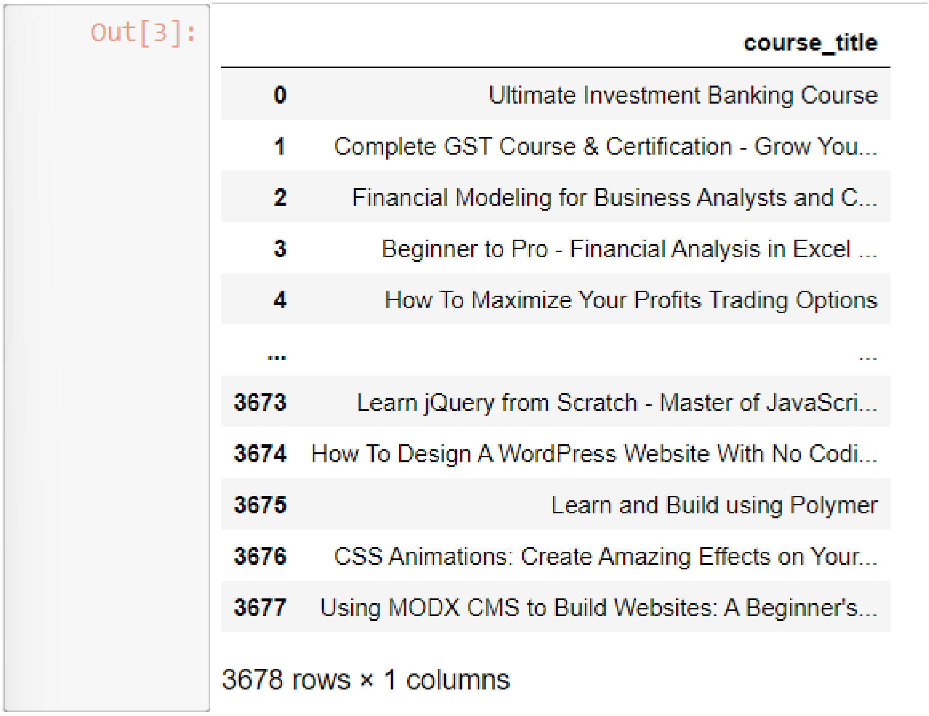
ALGORITHM:

- Remove the trailing and leading whitespace by fetching the correct columns and the correct function as str.strip()

PROGRAM:

```
courses = data.iloc[:, 1:2]

courses['course_title'] = courses['course_title'].str.strip()

courses
```

OUTPUT:



Out[3]:

	course_title
0	Ultimate Investment Banking Course
1	Complete GST Course & Certification - Grow You...
2	Financial Modeling for Business Analysts and C...
3	Beginner to Pro - Financial Analysis in Excel ...
4	How To Maximize Your Profits Trading Options
...	...
3673	Learn jQuery from Scratch - Master of JavaScri...
3674	How To Design A WordPress Website With No Codi...
3675	Learn and Build using Polymer
3676	CSS Animations: Create Amazing Effects on Your...
3677	Using MODX CMS to Build Websites: A Beginner's...

3678 rows × 1 columns

QUESTION -1(c):

Perform feature extraction using the Term frequency inverse document frequency (TF – IDF)

ALGORITHM:

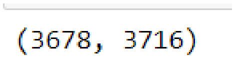
- Import the sklearn feature extraction library from TfidVectorizer
- Declare the TfidVectorizer and fit the transformation into the column called course_title.
- Print the tfid Vectorizer shape.

PROGRAM:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tf = TfidfVectorizer()

tfidf_matrix = tf.fit_transform(courses['course_title'])

print(tfidf_matrix.shape)
```

OUTPUT:



```
(3678, 3716)
```

QUESTION – 1(d):

Compute the cosine similarity.

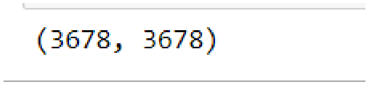
ALGORITHM:

- Import the pairwise from the cosine_similarity libaray.
- Define the cosine_similarity with the two parameters of tfidf_matrix.
- Print the cosine similarity with the shape attribute.

PROGRAM:

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

print(cosine_sim.shape)
```

OUTPUT:



```
(3678, 3678)
```

QUESTION -1(e):

Display the top ‘n’ suggestions with the similarity score for the given user input.

ALGORITHM:

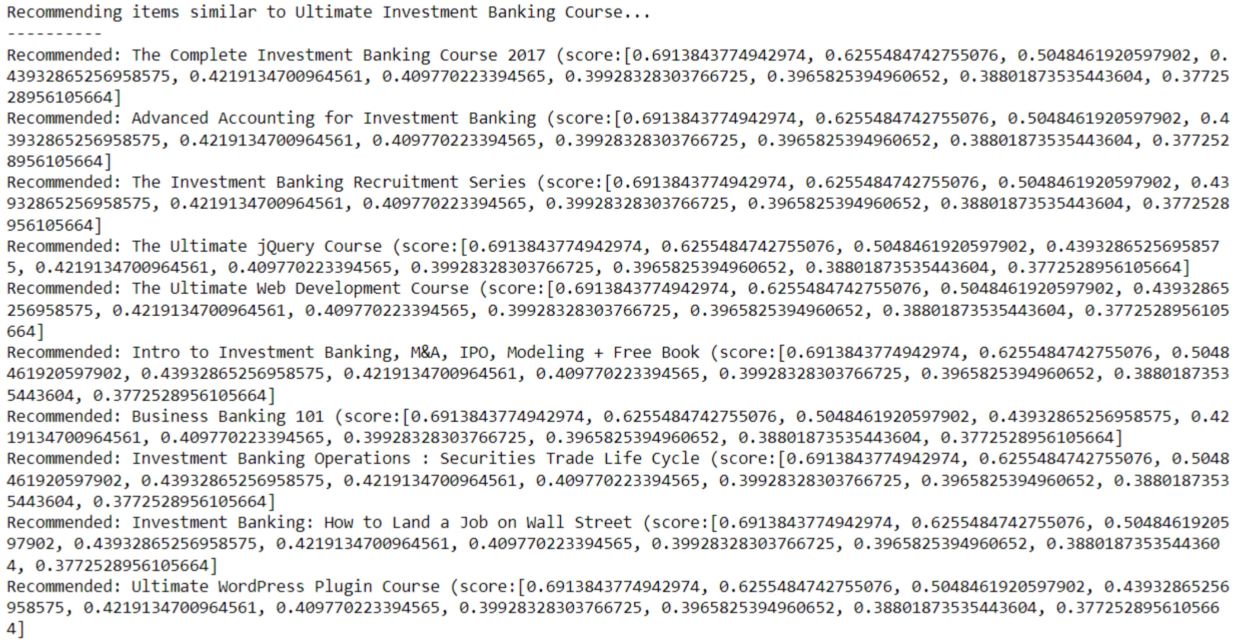
- To display the top n suggestions, use the column name and the title of the courses.
- Get the product and the number as an input from the user.
- Declare the idx as the indices of the particular product.
- Declare the sim scores as the list of enumerate function and the sorted functions.
- Print the recommended similar items using the for loop towards the input as the output.

PROGRAM:

```
products = courses['course_title']

indices = pd.Series(courses.index, index=courses['course_title'])


product = input("Enter the items related to recommend: ")

num = int(input("Number of recommendations: "))
```

```
idx = indices[product]

sim_scores = list(enumerate(cosine_sim[idx]))

sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

sim_scores = sim_scores[1:num+1]

items_indices = [i[0] for i in sim_scores]



scores = [i[1] for i in sim_scores]

print("Recommending items similar to " + product + "...")

print("----------")



for rec in range(num):

    print("Recommended: " + products[items_indices[rec]] + " (score:" + str(scores))
```

OUTPUT:



```
Recommending items similar to Ultimate Investment Banking Course...
----------
Recommended: The Complete Investment Banking Course 2017 (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.
43932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.37725
28956105664]
Recommended: Advanced Accounting for Investment Banking (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.4
3932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.377252
8956105664]
Recommended: The Investment Banking Recruitment Series (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.43
932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.3772528
956105664]
Recommended: The Ultimate jQuery Course (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.4393286525695857
5, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.3772528956105664]
Recommended: The Ultimate Web Development Course (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.43932865
256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.3772528956105
664]
Recommended: Intro to Investment Banking, M&A, IPO, Modeling + Free Book (score:[0.6913843774942974, 0.6255484742755076, 0.5048
461920597902, 0.43932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.3880187353
5443604, 0.3772528956105664]
Recommended: Business Banking 101 (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.43932865256958575, 0.42
19134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.3772528956105664]
Recommended: Investment Banking Operations : Securities Trade Life Cycle (score:[0.6913843774942974, 0.6255484742755076, 0.5048
461920597902, 0.43932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.3880187353
5443604, 0.3772528956105664]
Recommended: Investment Banking: How to Land a Job on Wall Street (score:[0.6913843774942974, 0.6255484742755076, 0.50484619205
97902, 0.43932865256958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.3880187353544360
4, 0.3772528956105664]
Recommended: Ultimate WordPress Plugin Course (score:[0.6913843774942974, 0.6255484742755076, 0.5048461920597902, 0.43932865256
958575, 0.4219134700964561, 0.409770223394565, 0.39928328303766725, 0.3965825394960652, 0.38801873535443604, 0.377252895610566
4]
```

Result:

Therefore, the design on the content based recommender system is coded and verified successfully by obtained the required output.