NAME:HARSH ARORA
ROLL NO:AE-1218
COURSE:BSc(Hons.) CS

Subject:GE(DATA Engineering and Analytics)

College: Acharya Narendra Dev College (DU)

INDEX:

Data Engineering & Analytics (ELGE-1B)
GE-1 (Electronics)
PROGRAM LIST

LAB 1-2: Basic Data Analysis using PANDAS python library

Write programs to perform following functions on the given Student datasets using Pandas library:

Student Name	Physics	Chemistry	Maths
Aman	78	80	88
Rahul	67	77	95
Shreya	85	85	100
Vinita	96	90	90

StudentDataset1

	Student Results					
Roll #	Student	Department	General Sc.	Math's	Computer	Remarks
1	Agarwal V	Computer Sc	78	80	66	Pass
2	Agnew P	Computer Sc	45	78	66	Pass
3	Ahmad H	Statistics	78	30	50	Fail
4	Alambritis G	Physics	65	68	70	Pass
5	Allen J	Economics	87	50	60	Pass
6	Anthony A	Math's	54	30	30	Fail
7	Antoniou V	Computer Sc	76	70	70	?
8	Archy J	Economics	76	40	40	Fail
9	Armstrong B	Math's	86	30	40	?
10	Arvanitakis I	Physics	68	70	80	Pass

StudentDataset2

1. Creating Data frame from the given dataset by using DataFrame( ) function in pandas
2. Reading data (certain number of rows and columns) from a Data frame using **Locate Row**, loc[ ] and iloc[ ] function
3. Adding name to each row using **'index'** and to access data of a row using row index name
4. To access data of a student by specifying 'Student Name
5. Reading student data from .xls/.xlsx file, .csv file and .json file
6. Determining the shape (rows, columns) of the dataset using shape function
7. Determining the size (no. of elements) in the given dataset using size function
8. Displaying first few rows/ last few rows of a dataframe using head/tail function
9. Displaying information about the type of data in a dataframe using info function
10. Listing names of all the columns in a dataset
11. Inserting a new column in an existing dataframe using/without using insert function
12. Inserting a new row in an existing dataframe using loc function
13. Concatenation of two dataframes using concat function
14. To evaluate percentage and display it in a separate column
15. Evaluation of the following statistical parameters using describe function: Count, Mean, Min, Max, Standard Deviation, 25th Percentile, 50th Percentile, 75th Percentile

Lab 1:

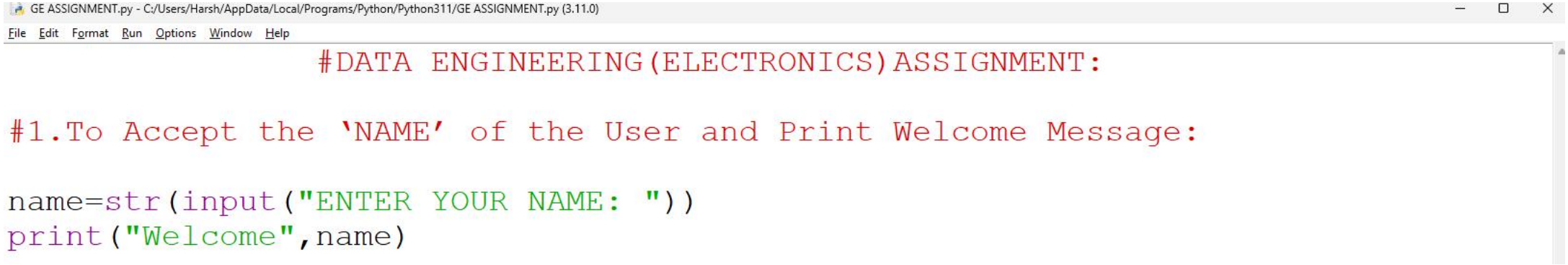
GE ASSIGNMENT.py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py (3.11.0)

File Edit Format Run Options Window Help

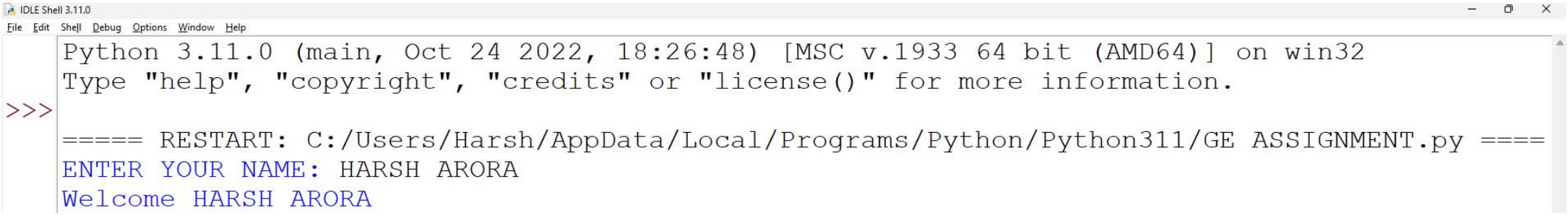
```
                                        #DATA ENGINEERING(ELECTRONICS)ASSIGNMENT:

#1.To Accept the 'NAME' of the User and Print Welcome Message:

name=str(input("ENTER YOUR NAME: "))
print("Welcome",name)


#2. To Accept Real (DECIMAL) Numbers From the User and Print Their Addition:

a=float(input("ENTER THE FIRST NUMBER: "))
b=float(input("ENTER THE SECOND NUMBER: "))
print("The Addition of These Numbers is",a+b)

#3. To Accept the Value of Current Temperature From The User:

t=int(input("ENTER THE CURRENT TEMPERATURE: "))
if t>27:
    print("WEATHER IS HOT")
if t==27:
    print("WEATHER IS COMFORTABLE")
if t<27:
    print("WEATHER IS COLD")

#4. To Design a SIMPLE CALCULATOR:

a=float(input("Enter the First Number: "))
b=float(input("Enter the Second  Number: "))
c=str(input("Put the Opeartion you Want to Perform(+,-,*,/): "))
if c=="+":
    print("The Sum of these Numbers is:",a+b)
elif c=="-":
    print("The Difference of these Numbers is:",a-b)
elif c=="*":
    print("The Multiplication of these Numbers is:",a*b)
elif c=="/":
    print("The Division of these Numbers is:",a/b)

#5. To Find The Greatest of Three Integers:

a=int(input("ENTER THE FIRST NUMBER: "))
b=int(input("ENTER THE SECOND NUMBER: "))
c=int(input("ENTER THE THIRD NUMBER: "))

if a>b and a>c:
    print("THE FIRST NUMBER Is THE GREATEST NUMBER")
if b>a and b>c:
    print("THE SECOND NUMBER Is THE GREATEST NUMBER")
if c>b and c>a:
    print("THE THIRD NUMBER Is THE GREATEST NUMBER")
```

5

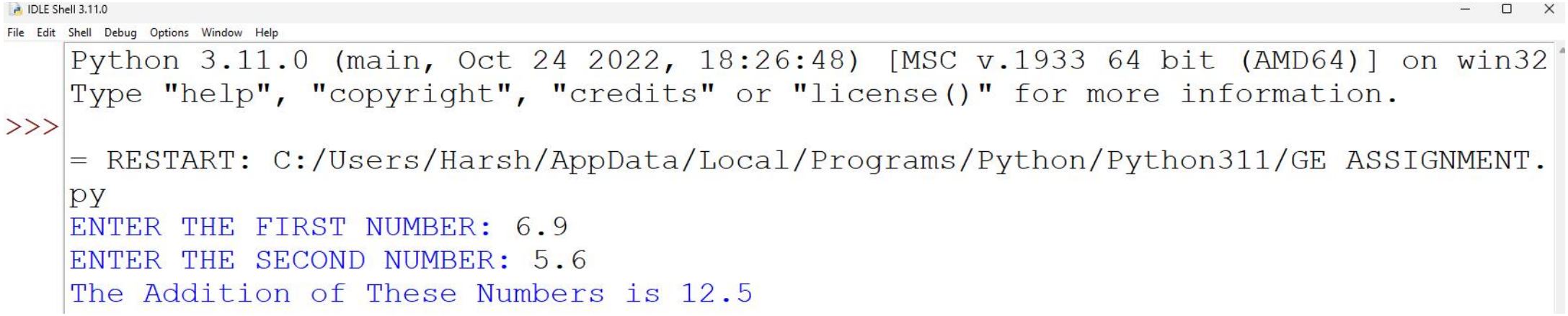
```
#6. To Swap Two Numbers:

a=int(input("ENTER THE FIRST NUMBER: "))
b=int(input("ENTER THE SECOND NUMBER: "))

print("NUMBERS BEFORE SWAPPING",(a,b))
a,b==b,a
print("NUMBERS AFTER SWAPPING",(b,a))

#7. To Check Whether a Number is Even OR Odd:

a=int(input("ENTER YOUR NUMBER: "))
if a%2==0:
    print("THE NUMBER IS EVEN")
else:
    print("THE NUMBER IS ODD")

#8.To Determine The Roots of a Quadratic Equation:
a=int(input("Enter The Cofficient of X2: "))
b=int(input("Enter The Cofficient of X: "))
c=int(input("Enter The Cofficient of Constant: "))
d=b*2-4*a*c
if d>0:
    x1=-b+(d)**0.5/2*a
    x2=-b-(d)**0.5/2*a
    print("The Roots are",x1,",",x2)
else:
    print("\nPlease Enter real roots!!!!\n")
```


GE ASSIGNMENT.py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py (3.11.0)
File Edit Format Run Options Window Help
#DATA ENGINEERING(ELECTRONICS)ASSIGNMENT:
#1.To Accept the 'NAME' of the User and Print Welcome Message:
name=str(input("ENTER YOUR NAME: "))
print("Welcome",name)

7


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
ENTER YOUR NAME: HARSH ARORA
Welcome HARSH ARORA

```
#2. To Accept Real (DECIMAL) Numbers From the User and Print Their Addition:

a=float(input("ENTER THE FIRST NUMBER: "))
b=float(input("ENTER THE SECOND NUMBER: "))
print("The Addition of These Numbers is",a+b)
```


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py
ENTER THE FIRST NUMBER: 6.9
ENTER THE SECOND NUMBER: 5.6
The Addition of These Numbers is 12.5

```
#3. To Accept the Value of Current Temperature From The User:

t=int(input("ENTER THE CURRENT TEMPERATURE: "))
if t>27:
    print("WEATHER IS HOT")
if t==27:
    print("WEATHER IS COMFORTABLE")
if t<27:
    print("WEATHER IS COLD")
```

```
>>>
= RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.
py
ENTER THE CURRENT TEMPERATURE: 45
WEATHER IS HOT
>>>
= RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.
py
ENTER THE CURRENT TEMPERATURE: 9
WEATHER IS COLD
>>>
= RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.
py
ENTER THE CURRENT TEMPERATURE: 27
WEATHER IS COMFORTABLE
```

```
#4. To Design a SIMPLE CALCULATOR:

a=float(input("Enter the First Number: "))
b=float(input("Enter the Second  Number: "))
c=str(input("Put the Opeartion you Want to Perform(+,-,*,/): "))
if c=="+":
    print("The Sum of these Numbers is:",a+b)
elif c=="-":
    print("The Difference of these Numbers is:",a-b)
elif c=="*":
    print("The Multiplication of these Numbers is:",a*b)
elif c=="/":
    print("The Division of these Numbers is:",a/b)
```

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
Enter the First Number: 1.2
Enter the Second  Number: 6.5
Put the Opeartion you Want to Perform(+,-,*,/): *
The Multiplication of these Numbers is: 7.8
```

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
Enter the First Number: 6.7
Enter the Second  Number: 3.4
Put the Opeartion you Want to Perform(+,-,*,/): +
The Sum of these Numbers is: 10.1
```

```
#5. To Find The Greatest of Three Integers:

a=int(input("ENTER THE FIRST NUMBER: "))
b=int(input("ENTER THE SECOND NUMBER: "))
c=int(input("ENTER THE THIRD NUMBER: "))

if a>b and a>c:
    print("THE FIRST NUMBER Is THE GREATEST NUMBER")
if b>a and b>c:
    print("THE SECOND NUMBER Is THE GREATEST NUMBER")
if c>b and c>a:
    print("THE THIRD NUMBER Is THE GREATEST NUMBER")
```

16

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
ENTER THE FIRST NUMBER: 108
ENTER THE SECOND NUMBER: 290
ENTER THE THIRD NUMBER: 527
THE THIRD NUMBER Is THE GREATEST NUMBER
```

```
#6. To Swap Two Numbers:

a=int(input("ENTER THE FIRST NUMBER: "))
b=int(input("ENTER THE SECOND NUMBER: "))

print("NUMBERS BEFORE SWAPPING",(a,b))
a,b==b,a
print("NUMBERS AFTER SWAPPING",(b,a))
```

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
ENTER THE FIRST NUMBER: 5
ENTER THE SECOND NUMBER: 9
NUMBERS BEFORE SWAPPING (5, 9)
NUMBERS AFTER SWAPPING (9, 5)
```

```
#7. To Check Whether a Number is Even OR Odd:

a=int(input("ENTER YOUR NUMBER: "))
if a%2==0:
    print("THE NUMBER IS EVEN")
else:
    print("THE NUMBER IS ODD")
```

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
ENTER YOUR NUMBER: 78
THE NUMBER IS EVEN

===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
ENTER YOUR NUMBER: 69
THE NUMBER IS ODD
```

```
#8.To Determine The Roots of a Quadratic Equation:
a=int(input("Enter The Cofficient of X2: "))
b=int(input("Enter The Cofficient of X: "))
c=int(input("Enter The Cofficient of Constant: "))
d=b*2-4*a*c
if d>0:
    x1=-b+(d)**0.5/2*a
    x2=-b-(d)**0.5/2*a
    print("The Roots are",x1,",",x2)
else:
    print("\nPlease Enter real roots!!!!\n")
```

```
===== RESTART: C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMENT.py ====
Enter The Cofficient of X2: 1
Enter The Cofficient of X: -7
Enter The Cofficient of Constant: -30
The Roots are 12.1478150704935 , 1.8521849295064996
```

```
Enter The Cofficient of X2: 4
Enter The Cofficient of X: 4
Enter The Cofficient of Constant: 1

Please Enter real roots!!!!
```

LAB 2:

In [ ]:
NAME : HARSH ARORA
ROLL NO:AE-1218

In [2]:
```
# Creating a DataFrame from the given dataset by using DataFrame() function in pandas
import pandas as pd
StudentData = {'Student Name':['Aman','Rahul', 'Shreya', 'Vinita'],
               'Physics':[78,67,85,96],
               'Chemistry': [80,77,85,90],
               'Maths':[88,95,100,90]}
STUDENT_DATA = pd.DataFrame(StudentData)
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
2       Shreya       85         85    100
3       Vinita       96         90     90
```

In [11]:
```
#reading data from a data frame using locate row,loc[] and iloc[] function
StudentData = {'Student Name':['Aman','Rahul', 'Shreya', 'Vinita'],
               'Physics':[78,67,85,96],
               'Chemistry': [80,77,85,90],
               'Maths':[88,95,100,90]}
STUDENT_DATA = pd.DataFrame(StudentData)
print(STUDENT_DATA.loc[[1]])#can use index name after adding index
print(STUDENT_DATA.iloc[[0]])
```

```
  Student Name  Physics  Chemistry  Maths
1        Rahul       67         77     95
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
```

In [4]:
```
#Adding name to each row using 'index'
StudentData = {'Student Name':['Aman','Rahul', 'Shreya', 'Vinita'],
               'Physics':[78,67,85,96],
               'Chemistry': [80,77,85,90],
               'Maths':[88,95,100,90]}
STUDENT_DATA = pd.DataFrame(StudentData,index = ['std1','std2','std3','std4'])
print(STUDENT_DATA.to_string())
print(STUDENT_DATA.loc[['std1']])
```

```
     Student Name  Physics  Chemistry  Maths
std1         Aman       78         80     88
std2        Rahul       67         77     95
std3       Shreya       85         85    100
std4       Vinita       96         90     90
     Student Name  Physics  Chemistry  Maths
std1         Aman       78         80     88
```

In [13]:
```
# To access data of a row using row index name
StudentData = {'Student Name':['Aman','Rahul', 'Shreya', 'Vinita'],
               'Physics':[78,67,85,96],
               'Chemistry': [80,77,85,90],
               'Maths':[88,95,100,90]}
STUDENT_DATA = pd.DataFrame(StudentData,index = ['std1','std2','std3','std4'])
print(STUDENT_DATA.loc[['std1']])
```

```
     Student Name  Physics  Chemistry  Maths
std1         Aman       78         80     88
```

In [14]:
```
#to access data of a student by specifying student name
StudentData = {'Student Name':['Aman','Rahul', 'Shreya', 'Vinita'],
               'Physics':[78,67,85,96],
               'Chemistry': [80,77,85,90],
               'Maths':[88,95,100,90]}
STUDENT_DATA = pd.DataFrame(StudentData)
STUDENT_DATA=STUDENT_DATA.set_index(['Student Name'])
print(STUDENT_DATA.loc[['Rahul']])
```

```
              Physics  Chemistry  Maths
Student Name
Rahul              67         77     95
```

In [23]:
```
# Reading data from a spreadsheet (.xlsx/.xls) file
import pandas as pd
import openpyxl
STUDENT_DATA = pd.read_excel(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.xlsx")#using r so that we can read the file
print(STUDENT_DATA)
```

```
  Student Name  Physics   Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
2       Shreya       85         85    100
3       Vinita       96         90     90
```

In [24]:
```
#reading data from a csv file
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")#using r so that we can read the file
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
2       Shreya       85         85    100
3       Vinita       96         90     90
```

In [25]:
```
#reading data from a json file
STUDENT_DATA = pd.read_json(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.json")#using r so that we can read the file
print(STUDENT_DATA)
```

```
  Student Names  Physics  Chemistry  Maths
0          Aman       78         80     88
1         Rahul       67         77     95
2        Shreya       85         85    100
3        Vinita       96         90     90
```

In [44]:
```
# To determine the shape ( no.of ROWs x no. of COLUMNS) in a given data frame
STUDENT_DATA = pd.DataFrame(StudentData)
print("(ROWS, COLUMNS) in the given data set is",STUDENT_DATA.shape)
```

```
(ROWS, COLUMNS) in the given data set is (4, 4)
```

In [27]:
```
# To determine the size ( no.of data elements) in a given data frame
STUDENT_DATA = pd.DataFrame(StudentData)
print("No. of elements in the given data set are",STUDENT_DATA.size)
```

```
No. of elements in the given data set are 16
```

In [28]:
```
#display first few rows of a dataframe using head function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
print(STUDENT_DATA.head(2))
```

```
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
```

In [29]:
```
#display last few rows of a dataframe using tail function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
print(STUDENT_DATA.tail(2))
```

```
  Student Name  Physics  Chemistry  Maths
2       Shreya       85         85    100
3       Vinita       96         90     90
```

In [30]:
```
#displaying information about the type of data in a dataframe using info function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
print(STUDENT_DATA.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Student Name  4 non-null      object
 1   Physics       4 non-null      int64 
 2   Chemistry     4 non-null      int64 
 3   Maths         4 non-null      int64 
dtypes: int64(3), object(1)
memory usage: 256.0+ bytes
None
```

In [31]:
```
#listing names of all the columns in a dataset
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
print(STUDENT_DATA.columns)
```

```
Index(['Student Name', 'Physics', 'Chemistry', 'Maths'], dtype='object')
```

In [32]:
```
#inserting a new column in an existing dataframe using insert function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
STUDENT_DATA.insert(4,'English',[96,75,88,80],True)
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths  English
0         Aman       78         80     88       96
1        Rahul       67         77     95       75
2       Shreya       85         85    100       88
3       Vinita       96         90     90       80
```

In [34]:
```
#inserting a new column in an existing dataframe without using insert function
English =[96,75,88,80]
STUDENT_DATA['English']=English
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths  English
0         Aman       78         80     88       96
1        Rahul       67         77     95       75
2       Shreya       85         85    100       88
3       Vinita       96         90     90       80
```

In [37]:
```
#inserting a new row in an existing dataframe using loc function#
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
STUDENT_DATA.loc[4]=['Vishal',88,97,100]
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
2       Shreya       85         85    100
3       Vinita       96         90     90
4       Vishal       88         97    100
```

In [38]:
```
#concatenation of two dataframes using concat function
StudentData1 ={'Student Name':['Aman','Rahul','Shreya'],
               'Physics':[78,67,85],
               'Chemistry':[80,77,85],
               'Maths':[88,95,100]}
StudentData2 ={'Student Name':['Vinita','Vishal'],
               'Physics':[96,88],
               'Chemistry':[90,97],
               'Maths':[90,100]}
STUDENT_DATA1=pd.DataFrame(StudentData1)
STUDENT_DATA2=pd.DataFrame(StudentData2)
STUDENT_DATA = pd.concat([STUDENT_DATA1,STUDENT_DATA2],ignore_index=True)
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths
0         Aman       78         80     88
1        Rahul       67         77     95
2       Shreya       85         85    100
3       Vinita       96         90     90
4       Vishal       88         97    100
```

In [41]:
```
# To evaluate percentage and display it in a separate column
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet1_LAB2.csv")
Percentage =(STUDENT_DATA['Physics']+
             STUDENT_DATA['Chemistry']+
             STUDENT_DATA['Maths'])/3
STUDENT_DATA['Percentage']=Percentage
print(STUDENT_DATA)
```

```
  Student Name  Physics  Chemistry  Maths  Percentage
0         Aman       78         80     88   82.000000
1        Rahul       67         77     95   79.666667
2       Shreya       85         85    100   90.000000
3       Vinita       96         90     90   92.000000
```

In [ ]:
```
#NAME: HARSH ARORA
#ROLL NOAE-1218
# COURSEBSC.(HONS.) COMPUTER SCIENCE
```

In [2]:
```
# Creating a DataFrame from the given dataset by using DataFrame() function in pandas
import pandas as pd
StudentData = {'Roll':[1,2,3,4,5,6,7,8,9,10],
               'Student':['Agarwal V','Agnew P','Ahmad H','Alambritis G','Allen J','Anthony A','Antoniou V','Archy J','Armstrong B','A
               'Department':['Computer Sc','Computer Sc','Statistics','Physics','Economics','Maths','Computer Sc','Economics','Math','
               'General Sc': [78,45,78,65,87,54,76,76,86,68],
               'Maths':[80,78,30,68,50,30,70,40,30,70],
               'Computer':[66,66,50,70,60,30,70,40,40,80],
               'Remarks':['Pass','Pass','Fail','Pass','Pass','Fail','?','Fail','?','Pass']}
STUDENT_DATA = pd.DataFrame(StudentData)
print(STUDENT_DATA)
```

```
   Roll        Student   Department  General Sc  Maths  Computer Remarks
0     1      Agarwal V  Computer Sc          78     80        66    Pass
1     2        Agnew P  Computer Sc          45     78        66    Pass
2     3        Ahmad H   Statistics          78     30        50    Fail
3     4   Alambritis G      Physics          65     68        70    Pass
4     5        Allen J    Economics          87     50        60    Pass
5     6      Anthony A        Maths          54     30        30    Fail
6     7     Antoniou V  Computer Sc          76     70        70       ?
7     8        Archy J    Economics          76     40        40    Fail
8     9    Armstrong B         Math          86     30        40       ?
9    10  Arvanitakis I      Physics          68     70        80    Pass
```

In [53]:
```
#reading data from a data frame using locate row,loc[] and iloc[] function
STUDENT_DATA = pd.DataFrame(StudentData)
print(STUDENT_DATA.loc[[1]])#can use index name after adding index
print()
print(STUDENT_DATA.iloc[[0]])
```

```
   Roll  Student   Department  General Sc  Maths  Computer Remarks
1     2  Agnew P  Computer Sc          45     78        66    Pass

   Roll    Student   Department  General Sc  Maths  Computer Remarks
0     1  Agarwal V  Computer Sc          78     80        66    Pass
```

In [3]:
```
#Adding name to each row using 'index'
STUDENT_DATA = pd.DataFrame(StudentData,index = ['std1','std2','std3','std4','std5','std6','std7','std8','std9','std10'])
print(STUDENT_DATA.to_string())
print(STUDENT_DATA.loc[['std1']])
```

```
       Roll        Student   Department  General Sc  Maths  Computer Remarks
std1      1      Agarwal V  Computer Sc          78     80        66    Pass
std2      2        Agnew P  Computer Sc          45     78        66    Pass
std3      3        Ahmad H   Statistics          78     30        50    Fail
std4      4   Alambritis G      Physics          65     68        70    Pass
std5      5        Allen J    Economics          87     50        60    Pass
std6      6      Anthony A        Maths          54     30        30    Fail
std7      7     Antoniou V  Computer Sc          76     70        70       ?
std8      8        Archy J    Economics          76     40        40    Fail
std9      9    Armstrong B         Math          86     30        40       ?
std10    10  Arvanitakis I      Physics          68     70        80    Pass
      Roll    Student   Department  General Sc  Maths  Computer Remarks
std1     1  Agarwal V  Computer Sc          78     80        66    Pass
```

In [8]:
```
# To access data of a row using row index name
STUDENT_DATA = pd.DataFrame(StudentData,index = ['std1','std2','std3','std4','std5','std6','std7','std8','std9','std10'])
print(STUDENT_DATA.loc[['std1']])
```

```
      Roll    Student   Department  General Sc  Maths  Computer Remarks
std1     1  Agarwal V  Computer Sc          78     80        66    Pass
```

In [11]:
```
#to access data of a student by specifying student name
STUDENT_DATA = pd.DataFrame(StudentData)
STUDENT_DATA=STUDENT_DATA.set_index(['Student'])
print(STUDENT_DATA.loc[['Agarwal V']])
```

```
           Roll   Department  General Sc  Maths  Computer Remarks
Student
Agarwal V     1  Computer Sc          78     80        66    Pass
```

In [12]:
```
# Reading data from a spreadsheet (.xlsx/.xls) file
import pandas as pd
import openpyxl
STUDENT_DATA = pd.read_excel(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.xlsx")#using r so that we can read the file
print(STUDENT_DATA)
```

```
   Roll        Student   Department  General Sc.  Maths's  Computer Remarks
0     1      Agarwal V  Computer Sc           78       80        66    Pass
1     2        Agnew P  Computer Sc           45       78        66    Pass
2     3        Ahmad H   Statistics           78       30        50    Fail
3     4   Alambritis G      Physics           65       68        70    Pass
4     5        Allen J    Economics           87       50        60    Pass
5     6      Anthony A       Math's           54       30        30    Fail
6     7     Antoniou V  Computer Sc           76       70        70       ?
7     8        Archy J    Economics           76       40        40    Fail
8     9    Armstrong B       Math's           86       30        40       ?
9    10  Arvanitakis I      Physics           68       70        80    Pass
```

In [56]:
```
#reading data from a csv file
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")#using r so that we can read the file
print(STUDENT_DATA)
```

```
   Roll       Students   Department  General Sc.  Maths  Computer Remarks
0     1      Agarwal V  Computer Sc           78     80        66    Pass
1     2        Agnew P  Computer Sc           45     78        66    Pass
2     3        Ahmad H   Statistics           78     30        50    Fail
3     4   Alambritis G      Physics           65     68        70    Pass
4     5        Allen J    Economics           87     50        60    Pass
5     6      Anthony A       Math's           54     30        30    Fail
6     7     Antoniou V  Computer Sc           76     70        70       ?
7     8        Archy J    Economics           76     40        40    Fail
8     9    Armstrong B       Math's           86     30        40       ?
9    10  Arvanitakis I      Physics           68     70        80    Pass
```

In [57]:
```
#reading data from a json file
STUDENT_DATA = pd.read_json(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.json")#using r so that we can read the file
print(STUDENT_DATA)
```

```
   Roll #        Student   Department  General Sc.  Math's  Computer Remarks
0       1      Agarwal V  Computer Sc           78      80        66    Pass
1       2        Agnew P  Computer Sc           45      78        66    Pass
2       3        Ahmad H   Statistics           78      30        50    Fail
3       4   Alambritis G      Physics           65      68        70    Pass
4       5        Allen J    Economics           87      50        60    Pass
5       6      Anthony A       Math's           54      30        30    Fail
6       7     Antoniou V  Computer Sc           76      70        70       ?
7       8        Archy J    Economics           76      40        40    Fail
8       9    Armstrong B       Math's           86      30        40       ?
9      10  Arvanitakis I      Physics           68      70        80    Pass
```

In [29]:
```
# To determine the shape ( no.of ROWs x no. of COLUMNS) in a given data frame
STUDENT_DATA = pd.DataFrame(StudentData)
print("(ROWS, COLUMNS) in the given data set is",STUDENT_DATA.shape)
```

```
(ROWS, COLUMNS) in the given data set is (10, 7)
```

In [30]:
```
# To determine the size ( no.of data elements) in a given data frame
STUDENT_DATA = pd.DataFrame(StudentData)
print("No. of elements in the given data set are",STUDENT_DATA.size)
```

```
No. of elements in the given data set are 70
```

In [32]:
```
#display first few rows of a dataframe using head function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
print(STUDENT_DATA.head(2))
```

```
   Roll   Students   Department  General Sc.  Math's  Computer Remarks
0     1  Agarwal V  Computer Sc           78      80        66    Pass
1     2    Agnew P  Computer Sc           45      78        66    Pass
```

In [33]:
```
#display last few rows of a dataframe using tail function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
print(STUDENT_DATA.tail(2))
```

```
   Roll       Students Department  General Sc.  Math's  Computer Remarks
8     9    Armstrong B     Math's           86      30        40       ?
9    10  Arvanitakis I    Physics           68      70        80    Pass
```

In [34]:
```
#displaying information about the type of data in a dataframe using info function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
print(STUDENT_DATA.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Roll         10 non-null     int64 
 1   Students     10 non-null     object
 2   Department   10 non-null     object
 3   General Sc.  10 non-null     int64 
 4   Math's       10 non-null     int64 
 5   Computer     10 non-null     int64 
 6   Remarks      10 non-null     object
dtypes: int64(4), object(3)
memory usage: 688.0+ bytes
None
```

In [35]:
```
#listing names of all the columns in a dataset
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
print(STUDENT_DATA.columns)
```

```
Index(['Roll', 'Students', 'Department', 'General Sc.', 'Math's', 'Computer',
       'Remarks'],
      dtype='object')
```

In [40]:
```
#inserting a new column in an existing dataframe using insert function
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
STUDENT_DATA.insert(6,'English',[96,75,88,80,45,67,89,78,68,90],True)
print(STUDENT_DATA)
```

```
   Roll       Students   Department  General Sc.  Math's  Computer  English  \
0     1      Agarwal V  Computer Sc           78      80        66       96   
1     2        Agnew P  Computer Sc           45      78        66       75   
2     3        Ahmad H   Statistics           78      30        50       88   
3     4   Alambritis G      Physics           65      68        70       80   
4     5        Allen J    Economics           87      50        60       45   
5     6      Anthony A       Math's           54      30        30       67   
6     7     Antoniou V  Computer Sc           76      70        70       89   
7     8        Archy J    Economics           76      40        40       78   
8     9    Armstrong B       Math's           86      30        40       68   
9    10  Arvanitakis I      Physics           68      70        80       90   

  Remarks  
0    Pass  
1    Pass  
2    Fail  
3    Pass  
4    Pass  
5    Fail  
6       ?  
7    Fail  
8       ?  
9    Pass  
```

In [39]:
```
#inserting a new column in an existing dataframe without using insert function
English =[96,75,88,80,45,67,89,78,68,90]
STUDENT_DATA['English']=English
print(STUDENT_DATA)
```

```
   Roll       Students   Department  General Sc.  Math's  Computer Remarks  \
0     1      Agarwal V  Computer Sc           78      80        66    Pass   
1     2        Agnew P  Computer Sc           45      78        66    Pass   
2     3        Ahmad H   Statistics           78      30        50    Fail   
3     4   Alambritis G      Physics           65      68        70    Pass   
4     5        Allen J    Economics           87      50        60    Pass   
5     6      Anthony A       Math's           54      30        30    Fail   
6     7     Antoniou V  Computer Sc           76      70        70       ?   
7     8        Archy J    Economics           76      40        40    Fail   
8     9    Armstrong B       Math's           86      30        40       ?   
9    10  Arvanitakis I      Physics           68      70        80    Pass   

   English  
0       96  
1       75  
2       88  
3       80  
4       45  
5       67  
6       89  
7       78  
8       68  
9       90  
```

In [43]:
```
#inserting a new row in an existing dataframe using loc function#
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
STUDENT_DATA.loc[4]=[5,'Vishal','Computer Sc',88,97,100,'Pass']
print(STUDENT_DATA)
```

```
   Roll       Students   Department  General Sc.  Math's  Computer Remarks
0     1      Agarwal V  Computer Sc           78      80        66    Pass
1     2        Agnew P  Computer Sc           45      78        66    Pass
2     3        Ahmad H   Statistics           78      30        50    Fail
3     4   Alambritis G      Physics           65      68        70    Pass
4     5         Vishal  Computer Sc           88      97       100    Pass
5     6      Anthony A       Math's           54      30        30    Fail
6     7     Antoniou V  Computer Sc           76      70        70       ?
7     8        Archy J    Economics           76      40        40    Fail
8     9    Armstrong B       Math's           86      30        40       ?
9    10  Arvanitakis I      Physics           68      70        80    Pass
```

In [44]:
```
#concatenation of two dataframes using concat function
StudentData1 ={'Roll':[1,2,3,4,5],
               'Student':['Agarwal V','Agnew P','Ahmad H','Alambritis G','Allen J'],
               'Department':['Computer Sc','Computer Sc','Statistics','Physics','Economics'],
               'General Sc': [78,45,78,65,87],
               'Maths':[80,78,30,68,50],
               'Computer':[66,66,50,70,60],
               'Remarks':['Pass','Pass','Fail','Pass','Pass']}
StudentData2 ={'Roll':[6,7,8,9,10],
               'Student':['Anthony A','Antoniou V','Archy J','Armstrong B','Arvanitakis I'],
               'Department':['Maths','Computer Sc','Economics','Math','Physics'],
               'General Sc': [54,76,76,86,68],
               'Maths':[30,70,40,30,70],
               'Computer':[30,70,40,40,80],
               'Remarks':['Fail','?','Fail','?','Pass']}
STUDENT_DATA1=pd.DataFrame(StudentData1)
STUDENT_DATA2=pd.DataFrame(StudentData2)
STUDENT_DATA = pd.concat([STUDENT_DATA1,STUDENT_DATA2],ignore_index=True)
print(STUDENT_DATA)
```

```
   Roll        Student   Department  General Sc  Maths  Computer Remarks
0     1      Agarwal V  Computer Sc          78     80        66    Pass
1     2        Agnew P  Computer Sc          45     78        66    Pass
2     3        Ahmad H   Statistics          78     30        50    Fail
3     4   Alambritis G      Physics          65     68        70    Pass
4     5        Allen J    Economics          87     50        60    Pass
5     6      Anthony A        Maths          54     30        30    Fail
6     7     Antoniou V  Computer Sc          76     70        70       ?
7     8        Archy J    Economics          76     40        40    Fail
8     9    Armstrong B         Math          86     30        40       ?
9    10  Arvanitakis I      Physics          68     70        80    Pass
```

In [51]:
```
# To evaluate percentage and display it in a separate column
STUDENT_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\lab 2\StudentDataSet2.csv")
Percentage =(STUDENT_DATA['General Sc.']+
             STUDENT_DATA['Maths']+
             STUDENT_DATA['Computer'])/3
STUDENT_DATA['Percentage']=Percentage
print(STUDENT_DATA)
```

```
   Roll       Students   Department  General Sc.  Maths  Computer Remarks  \
0     1      Agarwal V  Computer Sc           78     80        66    Pass
1     2        Agnew P  Computer Sc           45     78        66    Pass
2     3        Ahmad H   Statistics           78     30        50    Fail
3     4   Alambritis G      Physics           65     68        70    Pass
4     5        Allen J    Economics           87     50        60    Pass
5     6      Anthony A       Math's           54     30        30    Fail
6     7     Antoniou V  Computer Sc           76     70        70       ?
7     8        Archy J    Economics           76     40        40    Fail
8     9    Armstrong B       Math's           86     30        40       ?
9    10  Arvanitakis I      Physics           68     70        80    Pass

   Percentage
0   74.666667
1   63.000000
2   52.666667
3   67.666667
4   65.666667
5   38.000000
6   72.000000
7   52.000000
8   52.000000
9   72.666667
```

LAB 3 - 4: Data Representation using Vectors and Matrices using NUMPY Library

1. Write a program to perform the following functions

$$A = \begin{bmatrix} -5 & 1 & -3 \\ 6 & 0 & 2 \\ 2 & 6 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 4 & 5 \\ -8 & 10 & 3 \\ -2 & -3 & -9 \end{bmatrix}$$

- Create matrices A and B as shown in the attachment using array() function in numpy
- Write program to find add. subtract and multiply A and B
- Write a program to find the determinant of A and B

2. Determine the Rank and Nullity of matrices 1 to 5

1. $A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 4 & 2 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 4 & 0 & 3 & 0 \end{bmatrix}$

2. $Z = \begin{bmatrix} 5 & 4 & 7 \\ 5 & -6 & 5 \\ 4 & 2 & -3 \end{bmatrix}$

3. $A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 0 & -1 & 2 \end{bmatrix}$

4. $A = \begin{bmatrix} 1 & 1 & 2 \\ 3 & 4 & -1 \\ -1 & -2 & 5 \end{bmatrix}$

5. $A = \begin{bmatrix} 1 & 3 & 4 \\ 3 & 9 & 12 \\ 1 & 3 & 4 \end{bmatrix}$

3. Consider the following two vectors

u = (0.5, 0.4, 0.4, 0.5, 0.1, 0.4, 0.1) and v = (-1,-2, 1,-2, 3, 1,-5)

i. Check if u or v is a unit vector.
ii. Calculate the dot product, <u, v>
iii. Are u and v orthogonal?

4. Consider the following three vectors

v = (1, 2, 5, 2,-3, 1, 2, 6, 2)
u = (-4, 3, -2, 2, 1, -3, 4, 1, -2)
w = (3, 3, 3, -1, 6,-1, 2,-5,-7)

i. Evaluate dot product <v,w>
ii. Are any pair of vectors orthogonal, and if so which ones?

5. Consider the following three matrices A,B and C

$$A = \begin{bmatrix} 2 & -2 \\ -3 & 1 \\ 5 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 4 & 4 \\ -2 & 3 & -7 \\ 2 & 5 & -7 \end{bmatrix} \quad C = \begin{bmatrix} 4 & -1 & 2 \\ -8 & 2 & -4 \\ 2 & 1 & -4 \end{bmatrix}$$

Evaluate the following

i. A^TB
ii. $C + B$
iii. Which matrices are full rank?
iv. B^{-1}

**Full Rank matrices are those for which Rank = No. of columns=No. of Rows*

LAB 3:

GEE.py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GEE.py (3.11.0)

File Edit Format Run Options Window Help

```
#GE ASSIGNMENT LAB3
import numpy as np

#write a program to perform the following functions
arrA=np.array([[-5,1,-3],[6,0,2],[2,6,1]])
arrB=np.array([[2,4,5],[-8,10,3],[-2,-3,-9]])
print(arrA)
print(arrB)


#Write Program To Add,Subtract and Multiply Matrix A and B
arrA=np.array([[-5,1,-3],[6,0,2],[2,6,1]])
arrB=np.array([[2,4,5],[-8,10,3],[-2,-3,-9]])
print('Addition of These Two Matrices\n',arrA+arrB)
print('Subtraction of These Two Matrices\n',arrA-arrB)
arrC=np.dot(arrA,arrB)
print('Multiplication of These Two Matrices\n',arrC)


#Write a Program To Find the Determinant of A and B
DETA=np.linalg.det(arrA)
DETB=np.linalg.det(arrB)
print('Determinant of Matrix A is\n',DETA)
print('Determinant of Matrix B is\n',DETB)


#Determine The Rank and Nullity of MATRIX_1
arr1=np.array([[0,0,0,0],[4,2,3,0],[1,0,0,0],[4,0,3,0]])
RANK1=np.linalg.matrix_rank(arr1)
columns1=arr1.shape[1]
print('The Matrix is\n',arr1)
print('Rank of The Matrix is\n',RANK1)
print('Nullity of The Matrix is\n',columns1-RANK1)

#Determine The Rank and Nullity of MATRIX_2
arr2=np.array([[5,4,7],[5,-6,5],[4,2,-3]])
RANK2=np.linalg.matrix_rank(arr2)
columns2=arr2.shape[1]
print('The Matrix is\n',arr2)
print('Rank of The Matrix is\n',RANK2)
print('Nullity of The Matrix is\n',columns2-RANK2)

#Determine The Rank and Nullity of MATRIX_3
arr3=np.array([[1,0,1],[2,1,0],[0,-1,2]])
RANK3=np.linalg.matrix_rank(arr3)
columns3=arr3.shape[1]
print('The Matrix is\n',arr3)
print('Rank of The Matrix is\n',RANK3)
print('Nullity of The Matrix is\n',columns3-RANK3)
```

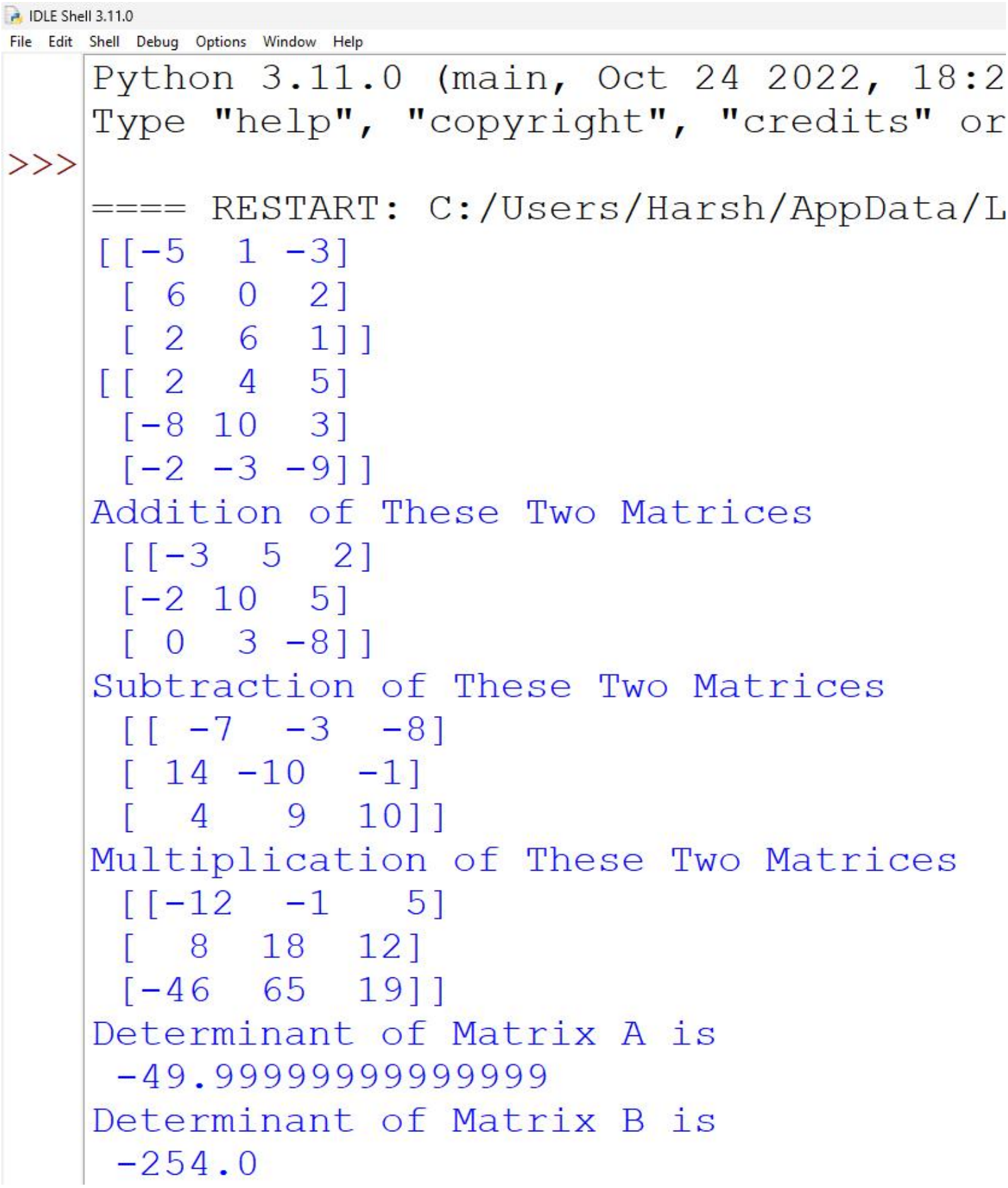
NAME: HARSH ARORA
ROLL NUMBER: AE-1218
COURSE:BSC(HONS.) CS

```
#Determine The Rank and Nullity of MATRIX_4
arr4=np.array([[1,1,2],[3,4,-1],[-1,-2,5]])
RANK4=np.linalg.matrix_rank(arr4)
columns4=arr4.shape[1]
print('The Matrix is\n',arr4)
print('Rank of The Matrix is\n',RANK4)
print('Nullity of The Matrix is\n',columns4-RANK4)

#Determine The Rank and Nullity of MATRIX_5
arr5=np.array([[1,3,4],[3,9,12],[1,3,4]])
RANK5=np.linalg.matrix_rank(arr5)
columns5=arr5.shape[1]
print('The Matrix is\n',arr5)
print('Rank of The Matrix is\n',RANK5)
print('Nullity of The Matrix is\n',columns5-RANK5)
```

GEE.py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GEE.py (3.11.0)

File Edit Format Run Options Window Help

```
                                        #GE ASSIGNMENT LAB3
import numpy as np

#write a program to perform the following functions
arrA=np.array([[-5,1,-3],[6,0,2],[2,6,1]])
arrB=np.array([[2,4,5],[-8,10,3],[-2,-3,-9]])
print(arrA)
print(arrB)


#Write Program To Add,Subtract and Multiply Matrix A and B
arrA=np.array([[-5,1,-3],[6,0,2],[2,6,1]])
arrB=np.array([[2,4,5],[-8,10,3],[-2,-3,-9]])
print('Addition of These Two Matrices\n',arrA+arrB)
print('Subtraction of These Two Matrices\n',arrA-arrB)
arrC=np.dot(arrA,arrB)
print('Multiplication of These Two Matrices\n',arrC)


#Write a Program To Find the Determinant of A and B
DETA=np.linalg.det(arrA)
DETB=np.linalg.det(arrB)
print('Determinant of Matrix A is\n',DETA)
print('Determinant of Matrix B is\n',DETB)
```


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:2
Type "help", "copyright", "credits" or
>>>
==== RESTART: C:/Users/Harsh/AppData/L
[[-5 1 -3]
[6 0 2]
[2 6 1]]
[[2 4 5]
[-8 10 3]
[-2 -3 -9]]
Addition of These Two Matrices
[[-3 5 2]
[-2 10 5]
[0 3 -8]]
Subtraction of These Two Matrices
[[-7 -3 -8]
[14 -10 -1]
[4 9 10]]
Multiplication of These Two Matrices
[[-12 -1 5]
[8 18 12]
[-46 65 19]]
Determinant of Matrix A is
-49.99999999999999
Determinant of Matrix B is
-254.0

```
#Determine The Rank and Nullity of MATRIX_1
arr1=np.array([[0,0,0,0],[4,2,3,0],[1,0,0,0],[4,0,3,0]])
RANK1=np.linalg.matrix_rank(arr1)
columns1=arr1.shape[1]
print('The Matrix is\n',arr1)
print('Rank of The Matrix is\n',RANK1)
print('Nullity of The Matrix is\n',columns1-RANK1)
```

```
========== RESTART: C:/Users
The Matrix is
  [[0 0 0 0]
  [4 2 3 0]
  [1 0 0 0]
  [4 0 3 0]]
Rank of The Matrix is
  3
Nullity of The Matrix is
  1
```

```
#Determine The Rank and Nullity of MATRIX_2
arr2=np.array([[5,4,7],[5,-6,5],[4,2,-3]])
RANK2=np.linalg.matrix_rank(arr2)
columns2=arr2.shape[1]
print('The Matrix is\n',arr2)
print('Rank of The Matrix is\n',RANK2)
print('Nullity of The Matrix is\n',columns2-RANK2)
```

```
The Matrix is
  [[ 5  4  7]
  [ 5 -6  5]
  [ 4  2 -3]]
Rank of The Matrix is
  3
Nullity of The Matrix is
  0
```

```
#Determine The Rank and Nullity of MATRIX_3
arr3=np.array([[1,0,1],[2,1,0],[0,-1,2]])
RANK3=np.linalg.matrix_rank(arr3)
columns3=arr3.shape[1]
print('The Matrix is\n',arr3)
print('Rank of The Matrix is\n',RANK3)
print('Nullity of The Matrix is\n',columns3-RANK3)
```

```
The Matrix is
  [[ 1  0  1]
  [ 2  1  0]
  [ 0 -1  2]]
Rank of The Matrix is
  2
Nullity of The Matrix is
  1
```

```
#Determine The Rank and Nullity of MATRIX_4
arr4=np.array([[1,1,2],[3,4,-1],[-1,-2,5]])
RANK4=np.linalg.matrix_rank(arr4)
columns4=arr4.shape[1]
print('The Matrix is\n',arr4)
print('Rank of The Matrix is\n',RANK4)
print('Nullity of The Matrix is\n',columns4-RANK4)
```

```
The Matrix is
  [[ 1   1   2]
  [ 3   4  -1]
  [-1  -2   5]]
Rank of The Matrix is
  2
Nullity of The Matrix is
  1
```

```
#Determine The Rank and Nullity of MATRIX_5
arr5=np.array([[1,3,4],[3,9,12],[1,3,4]])
RANK5=np.linalg.matrix_rank(arr5)
columns5=arr5.shape[1]
print('The Matrix is\n',arr5)
print('Rank of The Matrix is\n',RANK5)
print('Nullity of The Matrix is\n',columns5-RANK5)
```

```
The Matrix is
  [[ 1  3  4]
  [ 3  9 12]
  [ 1  3  4]]
Rank of The Matrix is
  1
Nullity of The Matrix is
  2
```

LAB 4

GEEELAB4.py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GEEELAB4.py (3.11.0)

File Edit Format Run Options Window Help

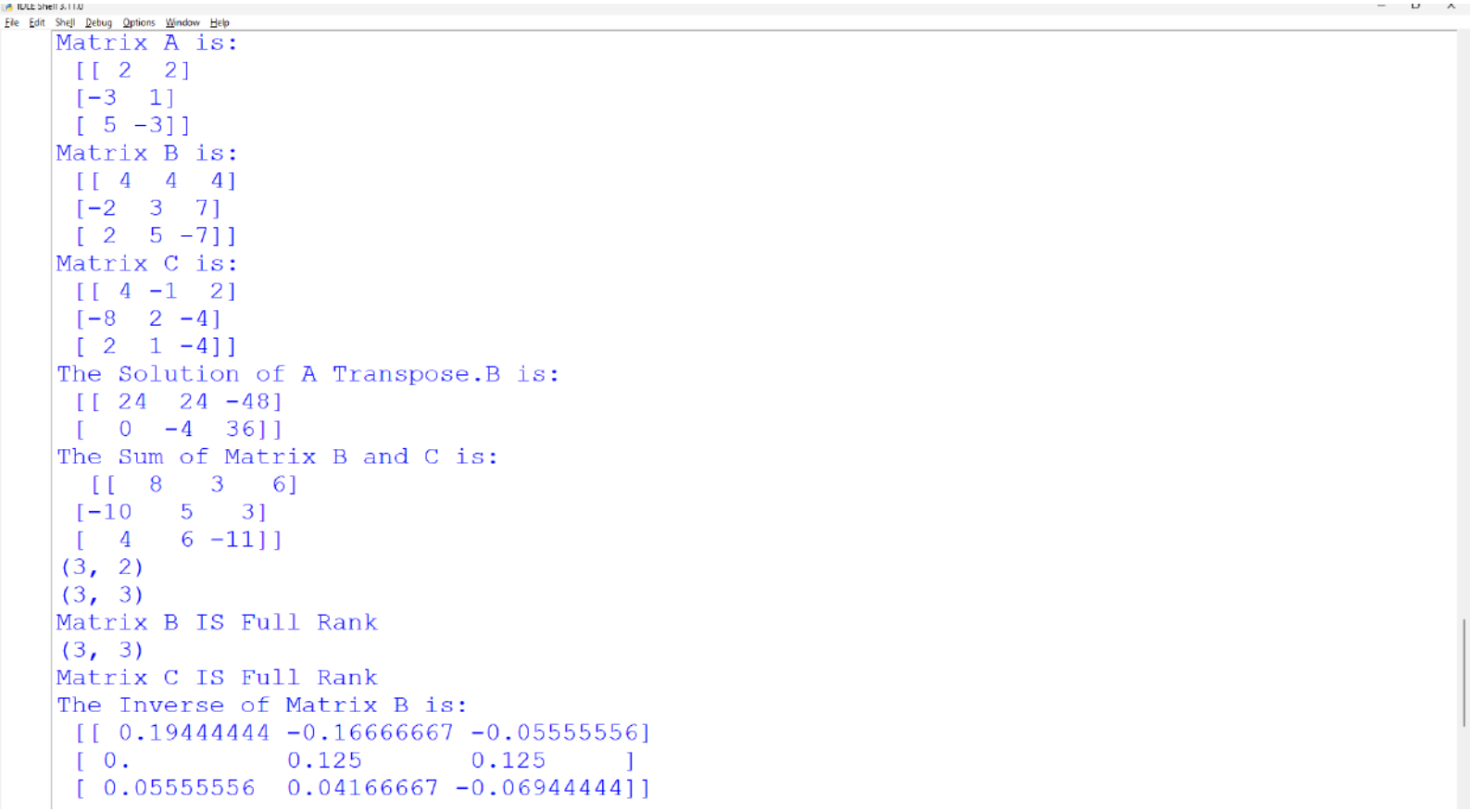
```
                                #GE ASSIGNMENT LAB 4:
import numpy as np



#1 Consider The Following Vectors:
vectorA= [0.5, 0.4, 0.4, 0.5, 0.1, 0.4, 0.1]
vectorB= [-1, -2, 1, -2, 3, 1,-5]
MagA=np.linalg.norm (vectorA)
MagB=np.linalg.norm (vectorB)
print ("The Magnitude of Vector A is: ", MagA)
print ("The Magnitude of Vector B is: ", MagB)
AdotB= np.dot(vectorA,vectorB)
print ("The Dot Product of Vector A and Vector B is : ",AdotB)
if AdotB!=0:
    print ("Vector A and B are Not Orthogonal")
print ("-----------------------------------------------------")
```

```
The Magnitude of Vector A is:  1.0
The Magnitude of Vector B is:  6.708203932499369
The Dot Product of Vector A and Vector B is :  -1.6999999999999997
Vector A and B are Not Orthogonal
-----------------------------------------------------
```

```
#2 Consider The Following Three Vectors:
vectorV= [1, 2, 5, 2,-3, 1, 2, 6, 2]
vectorU= [-4, 3, -2, 2, 1, -3, 4, 1, -2]
vectorW= [3, 3, 3, -1, 6,-1, 2,-5, -7]
VdotW= np.dot (vectorV, vectorW)
print ("The Dot Product of Vector V and Vector W is : ",VdotW)
VdotU= np.dot(vectorV, vectorU)
print ("The Dot Product of Vector V and Vector U is :",VdotU)
if VdotU==0:
    print ("Vector V and U are Orthogonal")
WdotU= np.dot (vectorW, vectorU)
print ("The Dot Product of Vector U and Vector W is : ",WdotU)
print("-------------------------------------------------------")
```

```
The Dot Product of Vector V and Vector W is :  -37
The Dot Product of Vector V and Vector U is : 0
Vector V and U are Orthogonal
The Dot Product of Vector U and Vector W is :  15
--------------------------------------------------
```

```
#3 CONSIDER THE THREE MATRICES:
arr1 = np.array([[2, 2], [-3,1], [5,-3]])
print ("Matrix A is: \n",arr1)
arr2=np.array([[4, 4, 4], [-2, 3, 7], [2,5, -7]])
print ("Matrix B is: \n", arr2)
arr3=np.array([[4,-1,2], [-8,2,-4], [2,1,-4]])
print ("Matrix C is: \n", arr3)
arrl_transpose=arr1.transpose()
arrl_transposedotarr2=np.dot(arrl_transpose, arr2)
print ("The Solution of A Transpose.B is: \n", arrl_transposedotarr2)
SUM=arr2+arr3
print ("The Sum of Matrix B and C is: \n ", SUM)
print(arr1.shape)
print(arr2.shape)
print("Matrix B IS Full Rank")
print(arr3.shape)
print("Matrix C IS Full Rank")
Inverse= np.linalg.inv (arr2)
print ("The Inverse of Matrix B is: \n", Inverse)
print("-----------------------------------------------------")
```


File Edit Shell Debug Options Window Help
Matrix A is:
[[2 2]
[-3 1]
[5 -3]]
Matrix B is:
[[4 4 4]
[-2 3 7]
[2 5 -7]]
Matrix C is:
[[4 -1 2]
[-8 2 -4]
[2 1 -4]]
The Solution of A Transpose.B is:
[[24 24 -48]
[0 -4 36]]
The Sum of Matrix B and C is:
[[8 3 6]
[-10 5 3]
[4 6 -11]]
(3, 2)
(3, 3)
Matrix B IS Full Rank
(3, 3)
Matrix C IS Full Rank
The Inverse of Matrix B is:
[[0.19444444 -0.16666667 -0.05555556]
[0. 0.125 0.125]
[0.05555556 0.04166667 -0.06944444]]

LAB 5: Data Representation using Vectors and Matrices using NUM

1. Create the dataframe shown below and perform the following functions using P

EMP ID	EMP NAME	SALARY	START DATE
1	Satish	50000	01-11-2017
2	Reeya	75000	12-05-2016
3	Jay	100000	22-09-2015
4	Rahul		11-10-2016
5	Roy	45000	08-01-2017
6	Jay	100000	22-09-2015
7	Vishal		05-01-2016
8	Serah	55000	06-02-2018

i. Display the column names and the number of records.
ii. Display the first 4 records of the dataset.

iii. For each numeric attribute, evaluate various statistical parameters function
iv. Check for the presence of missing values in the dataset and replace ther numeric value
v. Find and remove duplicate records (if any) in the dataset.

2. Download **Pima Indians Diabetes Dataset** using the link given below:

https://www.kaggle.com/datasets/kumargh/pimaindiansdiabetescsv

Perform the following operations:

i. Display the column names and the number of records.
ii. Display the first 10 records of the dataset.
iii. For each numeric attribute, evaluate various statistical parameters using describe() function
iv. Check for the presence of missing values in the dataset and replace them with some valid numeric value
v. Find and remove duplicate records (if any) in the dataset.
vi. Show scatter plot depicting relationship between two numeric columns of your choice.

LAB 5

NAME:HARSH ARORA
ROLL NO:AE-1218
COURSE:BSC(HONS.) COMPUTER SCIENCE

In [3]:
```
import pandas as pd
EmployeeData = {'EMP ID':[1,2,3,4,5,6,7,8],
                'EMP NAME':['Satish','Reeya','Jay','Rahul','Roy','Jay','Vishal','Serah'],
                'SALARY': [50000,75000,100000,None,45000,100000,None,55000],
                'START DATE':['1-11-2017','12-5-2016','22-9-2015','11-10-2016','8-1-2017','22-9-2015','5-1-2016','6-2-2018']}
EMPLOYEE_DATA = pd.DataFrame(EmployeeData)
#changing the start date data from string to yyyy/mm/dd type date

EMPLOYEE_DATA['START DATE']=pd.to_datetime(EMPLOYEE_DATA['START DATE'])
print(EMPLOYEE_DATA)
```

```
   EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
3       4    Rahul       NaN 2016-11-10
4       5      Roy   45000.0 2017-08-01
5       6      Jay  100000.0 2015-09-22
6       7   Vishal       NaN 2016-05-01
7       8    Serah   55000.0 2018-06-02
```

In [43]:
```
#Display the column names and the number of records.
COLUMNcount=EMPLOYEE_DATA.count()#gives no. of non-empty values in each column
print(COLUMNcount)
```

```
EMP ID        8
EMP NAME      8
SALARY        6
START DATE    8
dtype: int64
```

In [36]:
```
#Display the first 4 records of the dataset.
print(EMPLOYEE_DATA.head(4))
```

```
   EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
3       4    Rahul       NaN 2016-11-10
```

In [16]:
```
#For each numeric attribute, evaluate various statistical parameters using describe() function
print(EMPLOYEE_DATA.describe())
#CAN ALSO USE print(EMPLOYEE_DATA['SALARY'].describe()) to get result only for salary
```

```
         EMP ID         SALARY
count  8.00000       6.000000
mean   4.50000   70833.333333
std    2.44949   24782.386218
min    1.00000   45000.000000
25%    2.75000   51250.000000
50%    4.50000   65000.000000
75%    6.25000   93750.000000
max    8.00000  100000.000000
```

In [10]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print(EMPLOYEE_DATA.isnull())
EMPLOYEE_DATA_filled=EMPLOYEE_DATA.fillna(50000)
print('Original data table\n',EMPLOYEE_DATA)
print()
print('filled data table\n',EMPLOYEE_DATA_filled)
```

```
   EMP ID  EMP NAME  SALARY  START DATE
0   False     False   False       False
1   False     False   False       False
2   False     False   False       False
3   False     False    True       False
4   False     False   False       False
5   False     False   False       False
6   False     False    True       False
7   False     False   False       False
Original data table
    EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
3       4    Rahul       NaN 2016-11-10
4       5      Roy   45000.0 2017-08-01
5       6      Jay  100000.0 2015-09-22
6       7   Vishal       NaN 2016-05-01
7       8    Serah   55000.0 2018-06-02

filled data table
    EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
3       4    Rahul   50000.0 2016-11-10
4       5      Roy   45000.0 2017-08-01
5       6      Jay  100000.0 2015-09-22
6       7   Vishal   50000.0 2016-05-01
7       8    Serah   55000.0 2018-06-02
```

In [8]:
```
#Find and remove duplicate records (if any) in the dataset.
print(EMPLOYEE_DATA.duplicated())
EMPLOYEE_DATA_clean=EMPLOYEE_DATA.dropna()
print('Original data table\n',EMPLOYEE_DATA)
print()
print('filled data table\n',EMPLOYEE_DATA_clean)
```

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
dtype: bool
Original data table
    EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
3       4    Rahul       NaN 2016-11-10
4       5      Roy   45000.0 2017-08-01
5       6      Jay  100000.0 2015-09-22
6       7   Vishal       NaN 2016-05-01
7       8    Serah   55000.0 2018-06-02

filled data table
    EMP ID EMP NAME    SALARY START DATE
0       1   Satish   50000.0 2017-01-11
1       2    Reeya   75000.0 2016-12-05
2       3      Jay  100000.0 2015-09-22
4       5      Roy   45000.0 2017-08-01
5       6      Jay  100000.0 2015-09-22
7       8    Serah   55000.0 2018-06-02
```

In [4]:
```
#Pima Indians Diabetes Dataset
DIABETES_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\diabetes.csv")
```

In [47]:
```
#Display the column names and the number of records.
print(DIABETES_DATA.count())
```

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [7]:
```
#Display the first 10 records of the dataset.
print(DIABETES_DATA.head(10))
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1
5            5      116             74              0        0  25.6
6            3       78             50             32       88  31.0
7           10      115              0              0        0  35.3
8            2      197             70             45      543  30.5
9            8      125             96              0        0   0.0

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
5                     0.201   30        0
6                     0.248   26        1
7                     0.134   29        0
8                     0.158   53        1
9                     0.232   54        1
```

In [49]:
```
#For each numeric attribute, evaluate various statistical parameters using describe() function
print(DIABETES_DATA.describe())
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

In [5]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print(DIABETES_DATA.isnull())
DIABETES_DATA_filled=DIABETES_DATA.fillna(100)
print('Original data table\n',DIABETES_DATA)
print()
print('filled data table\n',DIABETES_DATA_filled)
```

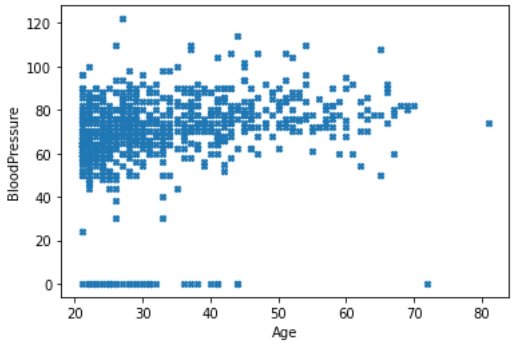
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0          False    False          False          False    False  False
1          False    False          False          False    False  False
2          False    False          False          False    False  False
3          False    False          False          False    False  False
4          False    False          False          False    False  False
..           ...      ...            ...            ...      ...    ...
763        False    False          False          False    False  False
764        False    False          False          False    False  False
765        False    False          False          False    False  False
766        False    False          False          False    False  False
767        False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        ...    ...      ...
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]
Original data table
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
..            ...      ...            ...            ...      ...   ...
763            10      101             76             48      180  32.9
764             2      122             70             27        0  36.8
765             5      121             72             23      112  26.2
766             1      126             60              0        0  30.1
767             1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]

filled data table
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
..            ...      ...            ...            ...      ...   ...
763            10      101             76             48      180  32.9
764             2      122             70             27        0  36.8
765             5      121             72             23      112  26.2
766             1      126             60              0        0  30.1
767             1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

In [6]:
```
#Find and remove duplicate records (if any) in the dataset.
print(DIABETES_DATA.duplicated())
DIABETES_DATA_clean=DIABETES_DATA.dropna()
print('Original data table\n',DIABETES_DATA)
print()
print('filled data table\n',DIABETES_DATA_clean)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
763    False
764    False
765    False
766    False
767    False
Length: 768, dtype: bool
Original data table
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
..            ...      ...            ...            ...      ...   ...
763            10      101             76             48      180  32.9
764             2      122             70             27        0  36.8
765             5      121             72             23      112  26.2
766             1      126             60              0        0  30.1
767             1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]

filled data table
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
..            ...      ...            ...            ...      ...   ...
763            10      101             76             48      180  32.9
764             2      122             70             27        0  36.8
765             5      121             72             23      112  26.2
766             1      126             60              0        0  30.1
767             1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

58

In [57]:
```
#Show scatter plot depicting relationship between two numeric columns of your choice.
# Scatter Plot between Current Vs Voltage from the given dataset

import pandas as pd
import matplotlib.pyplot as plt

DATA = pd.read_csv((r"C:\Users\MSI\Downloads\diabetes.csv"))
DATA.plot(kind='scatter', x='Age', y='BloodPressure',marker='X')
plt.show()
```



LAB 6: Data Munging , Data Aggregation and Grouping Operations
using PANDAS, MATPLOTLIB and SEABORN library)

1. Perform the following operations on the given dataset **CaloriesDataSet.csv**


CaloriesDataSet - Notepad

File Edit Format View Help

```
Duration,Date,Pulse,Maxpulse,Calories
60,'2020/12/01',110,130,409.1
60,'2020/12/02',117,145,479
60,'2020/12/03',103,135,340
45,'2020/12/04',109,175,282.4
45,'2020/12/05',117,148,406
60,'2020/12/06',102,127,300
60,'2020/12/07',110,136,374
450,'2020/12/08',104,134,253.3
30,'2020/12/09',109,133,195.1
60,'2020/12/10',98,124,269
60,'2020/12/11',103,147,329.3
60,'2020/12/12',100,120,250.7
60,'2020/12/12',100,120,250.7
60,'2020/12/13',106,128,345.3
60,'2020/12/14',104,132,379.3
60,'2020/12/15',98,123,275
60,'2020/12/16',98,120,215.2
60,'2020/12/17',100,120,300
45,'2020/12/18',90,112,
60,'2020/12/19',103,123,323
45,'2020/12/20',97,125,243
```

- Loading the Dataset
- Implement descriptive and summary statistics (to calculate Count, Mean, Max and Min, Percentile, Variance and Standard Deviation)
- Perform the following **DATA MUNGING and DATA CLEANING** operations (using PANDAS):
  - Check for the presence of missing values in the dataset and replace them with some valid numeric value
  - Find and remove duplicate records (if any) in the dataset.
  - Determine the correlation matrix for different columns (attributes) in a given dataset
- Perform the following **DATA VISUALISATION** operations (using Matplotlib and seaborn library)
  - Plot histogram, bar plot, distplot for various features attributes of the dataset
  - Plot Heatmap for the correlation between different attributes in the dataset
- Perform the following **DATA AGGREGATION and Grouping** operations:
  - using agg(), aggregate function to calculate sum, min and max of each column
  - group the dataset as per 'Duration' column
    - display the number (count) of values for each 'Duration'
    - display the sum of all the values for each 'Duration'
    - perform various Data Aggregation functions
    - perform various Data Aggregation functions for a particular column (attribute)

2. Download **Pima Indians Diabetes Dataset** using the link given below:
   https://www.kaggle.com/datasets/kumargh/pimaindiansdiabetescsv
   Perform similar Data Munging and Data Aggregation operations on the **Pima Indians Diabetes Dataset** as performed on CaloriesDataSet.csv

3. Download **Boston House Prices Dataset** using the link given below:
   https://www.kaggle.com/datasets/vikrishnan/boston-house-prices
   Perform similar Data Munging and Data Aggregation operations on the **Boston House Prices Dataset** as performed on CaloriesDataSet.csv in Q.1

LAB 6

In [ ]:
```
#Harsh Arora
#AE-1218
#BSc.(Hons.)Computer Science
```

In [27]:
```
import pandas as pd
#Loading the Dataset
CALORIES_DATA = pd.read_csv(r"C:\Users\MS\Downloads\Calories Set.csv")
```

In [5]:
```
# To Implement  The descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
# To Print Count
print('Count is\n',CALORIES_DATA.count(),'\n')
#To Print Mean
print('mean is\n',CALORIES_DATA.mean(),'\n')
#To Print Max
print('max is\n',CALORIES_DATA.max(),'\n')
#To Print Min
print('min is\n',CALORIES_DATA.min(),'\n')
#To Print Standard Deviation
print('standard deviation is\n',CALORIES_DATA.std(),'\n')

#print(Calories.describe()) #can also use to get the required output
```

```
Count is
 Duration    32
Date         31
Pulse        32
Maxpulse     32
Calories     30
dtype: int64

mean is
 Duration     68.4375
Pulse        103.5000
Maxpulse     128.5000
Calories     304.6800
dtype: float64

max is
 Duration    450.0
Pulse        130.0
Maxpulse     175.0
Calories     479.0
dtype: float64

min is
 Duration     30.0
Pulse         90.0
Maxpulse     101.0
Calories     195.1
dtype: float64

standard deviation is
 Duration    70.039591
Pulse         7.832933
Maxpulse     12.998759
Calories     66.003779
dtype: float64
```

In [8]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',CALORIES_DATA.isnull(),'\n')
CALORIES_DATA['Date'].fillna('2020/12/04',inplace=True)
CALORIES_DATA['Duration'].fillna('60',inplace=True)
CALORIES_DATA['Pulse'].fillna('100',inplace=True)
CALORIES_DATA['Maxpulse'].fillna('110',inplace=True)
CALORIES_DATA['Calories'].fillna('300',inplace=True)
print('updated dataset',CALORIES_DATA)
```

```
checking null values
     Duration   Date  Pulse  Maxpulse  Calories
0      False  False  False     False     False
1      False  False  False     False     False
2      False  False  False     False     False
3      False  False  False     False     False
4      False  False  False     False     False
5      False  False  False     False     False
6      False  False  False     False     False
7      False  False  False     False     False
8      False  False  False     False     False
9      False  False  False     False     False
10     False  False  False     False     False
11     False  False  False     False     False
12     False  False  False     False     False
13     False  False  False     False     False
14     False  False  False     False     False
15     False  False  False     False     False
16     False  False  False     False     False
17     False  False  False     False     False
18     False  False  False     False      True
19     False  False  False     False     False
20     False  False  False     False     False
21     False  False  False     False     False
22     False   True  False     False     False
23     False  False  False     False     False
24     False  False  False     False     False
25     False  False  False     False     False
26     False  False  False     False     False
27     False  False  False     False     False
28     False  False  False     False      True
29     False  False  False     False     False
30     False  False  False     False     False
31     False  False  False     False     False

updated dataset     Duration          Date  Pulse  Maxpulse Calories
0          60  '2020/12/01'    110       130    409.1
1          60  '2020/12/02'    117       145      479
2          60  '2020/12/03'    103       135      340
3          45  '2020/12/04'    109       175    282.4
4          45  '2020/12/05'    117       148      406
5          60  '2020/12/06'    102       127      300
6          60  '2020/12/07'    110       136      374
7         450  '2020/12/08'    104       134    253.3
8          30  '2020/12/09'    109       133    195.1
9          60  '2020/12/10'     98       124      269
10         60  '2020/12/11'    103       147    329.3
11         60  '2020/12/12'    100       120    250.7
12         60  '2020/12/12'    100       120    250.7
13         60  '2020/12/13'    106       128    345.3
14         60  '2020/12/14'    104       132    379.3
15         60  '2020/12/15'     98       123      275
16         60  '2020/12/16'     98       120    215.2
17         60  '2020/12/17'    100       120      300
18         45  '2020/12/18'     90       112      300
19         60  '2020/12/19'    103       123      323
20         45  '2020/12/20'     97       125      243
21         60  '2020/12/21'    108       131    364.2
22         45    2020/12/04    100       119      282
23         60  '2020/12/23'    130       101      300
24         45  '2020/12/24'    105       132      246
25         60  '2020/12/25'    102       126    334.5
26         60      20201226    100       120      250
27         60  '2020/12/27'     92       118      241
28         60  '2020/12/28'    103       132      300
29         60  '2020/12/29'    100       132      280
30         60  '2020/12/30'    102       129    380.3
31         60  '2020/12/31'     92       115      243
```

In [13]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',CALORIES_DATA.duplicated())
print()
CALORIES_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(CALORIES_DATA)
```

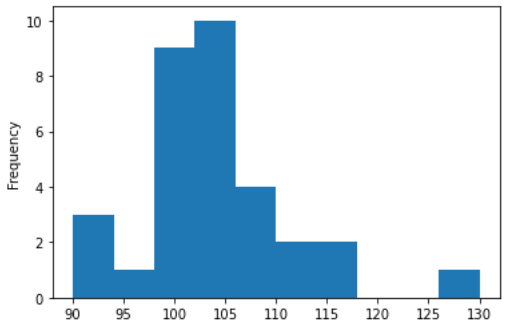
```
checking duplicate values
 0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
dtype: bool

dataset after removing duplicate values

    Duration          Date  Pulse  Maxpulse Calories
0         60  '2020/12/01'    110       130    409.1
1         60  '2020/12/02'    117       145      479
2         60  '2020/12/03'    103       135      340
3         45  '2020/12/04'    109       175    282.4
4         45  '2020/12/05'    117       148      406
5         60  '2020/12/06'    102       127      300
6         60  '2020/12/07'    110       136      374
7        450  '2020/12/08'    104       134    253.3
8         30  '2020/12/09'    109       133    195.1
9         60  '2020/12/10'     98       124      269
10        60  '2020/12/11'    103       147    329.3
11        60  '2020/12/12'    100       120    250.7
13        60  '2020/12/13'    106       128    345.3
14        60  '2020/12/14'    104       132    379.3
15        60  '2020/12/15'     98       123      275
16        60  '2020/12/16'     98       120    215.2
17        60  '2020/12/17'    100       120      300
18        45  '2020/12/18'     90       112      300
19        60  '2020/12/19'    103       123      323
20        45  '2020/12/20'     97       125      243
21        60  '2020/12/21'    108       131    364.2
22        45    2020/12/04    100       119      282
23        60  '2020/12/23'    130       101      300
24        45  '2020/12/24'    105       132      246
25        60  '2020/12/25'    102       126    334.5
26        60      20201226    100       120      250
27        60  '2020/12/27'     92       118      241
28        60  '2020/12/28'    103       132      300
29        60  '2020/12/29'    100       132      280
30        60  '2020/12/30'    102       129    380.3
31        60  '2020/12/31'     92       115      243
```

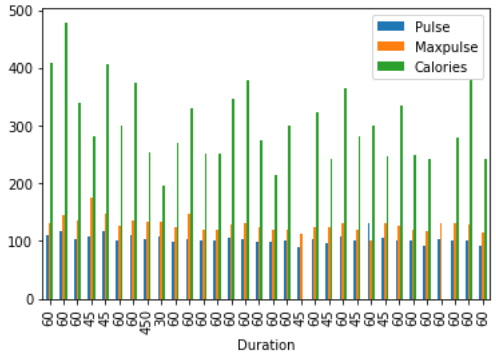
In [28]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
CALORIES_DATA.corr()
```

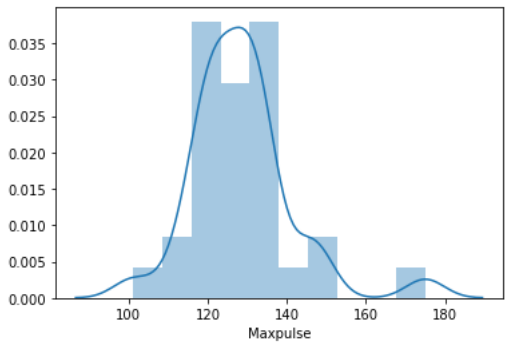
Out[28]:

	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	0.004410	0.049959	-0.114169
Pulse	0.004410	1.000000	0.276583	0.513186
Maxpulse	0.049959	0.276583	1.000000	0.357460
Calories	-0.114169	0.513186	0.357460	1.000000

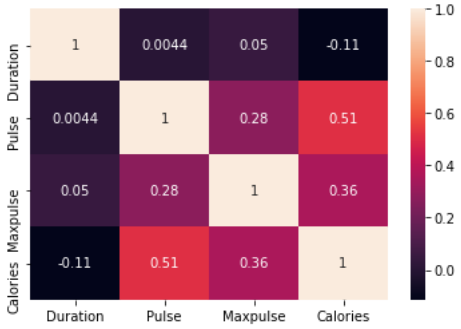
In [31]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
CALORIES_DATA['Pulse'].plot(kind = 'hist')
plt.show()
#bar
CALORIES_DATA.plot(kind = 'bar', x='Duration')
plt.show()
#distplot
import seaborn as sns
sns.distplot(CALORIES_DATA['Maxpulse'])
plt.show()
```







In [33]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(CALORIES_DATA.corr(),annot=True)
plt.show()
```



In [34]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
CALORIES_DATA.agg(['sum', 'min', 'max'])
```

Out[34]:

	Duration	Pulse	Maxpulse	Calories
sum	2190	3312	4112	9140.4
min	30	90	101	195.1
max	450	130	175	479.0

In [39]:
```
#group the dataset as per 'Duration‘ column
print('\ngrouping\n',CALORIES_DATA.groupby('Duration'))
#display the number (count) of values for each ‘Duration’
print('\nnumber (count) of values for each ‘Duration\n',CALORIES_DATA.groupby('Duration').count())
#display the sum of all the values for each ‘Duration’
print('\nsum of all the values for each ‘Duration’\n',CALORIES_DATA.groupby('Duration').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',CALORIES_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each ‘Duration’\n',CALORIES_DATA.groupby('Duration').agg(['sum', 'mean', 'median', 'min', 'max', 'p
```

```
grouping
 <pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000014F69D71850>

number (count) of values for each ‘Duration
           Date  Pulse  Maxpulse  Calories
Duration
30            1      1         1         1
45            5      6         6         5
60           24     24        24        23
450           1      1         1         1

sum of all the values for each ‘Duration’
           Pulse  Maxpulse  Calories
Duration
30           109       133     195.1
45           618       811    1459.4
60          2481      3034    7232.6
450          104       134     253.3

various Data Aggregation functions
              Duration         Pulse      Maxpulse      Calories
sum     2.190000e+03  3.312000e+03  4.112000e+03  9.140400e+03
mean    6.843750e+01  1.035000e+02  1.285000e+02  3.046800e+02
median  6.000000e+01  1.025000e+02  1.275000e+02  2.912000e+02
min     3.000000e+01  9.000000e+01  1.010000e+02  1.951000e+02
max     4.500000e+02  1.300000e+02  1.750000e+02  4.790000e+02
prod    3.664804e+18  8.222562e+18  8.700497e+18  1.701164e+74

sum of all the values for each ‘Duration’
           Pulse                                          Maxpulse              \
             sum     mean median  min  max          prod      sum        mean
Duration
30           109  109.000  109.0  109  109  1.090000e+02      133  133.000000
45           618  103.000  102.5   90  117  1.169004e+12      811  135.166667
60          2481  103.375  102.0   92  130  2.084879e+48     3034  126.416667
450          104  104.000  104.0  104  104  1.040000e+02      134  134.000000

                                      Calories                          \
          median  min  max          prod      sum       mean median    min
Duration
30         133.0  133  133  1.330000e+02    195.1  195.10000  195.1  195.1
45         128.5  112  175  5.695721e+12   1459.4  291.88000  282.0  243.0
60         126.5  101  147  2.591241e+50   7232.6  314.46087  300.0  215.2
450        134.0  134  134  1.340000e+02    253.3  253.30000  253.3  253.3


             max          prod
Duration
30         195.1  1.951000e+02
45         406.0  1.932775e+12
60         479.0  1.781036e+57
450        253.3  2.533000e+02
```

In [41]:
```
#DATASET 2 FROM HERE
```

In [2]:
```
import pandas as pd
#Loading the Dataset
DIABETES_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\diabetes.csv")
```

In [3]:
```
#Implement descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
#count
print('Count is\n',DIABETES_DATA.count(),'\n')
#mean
print('mean is\n',DIABETES_DATA.mean(),'\n')
#max
print('max is\n',DIABETES_DATA.max(),'\n')
#min
print('min is\n',DIABETES_DATA.min(),'\n')
#standard deviation
print('standard deviation is\n',DIABETES_DATA.std(),'\n')

#print(Calories.describe()) #can also use to get the required output
```

```
Count is
 Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64

mean is
 Pregnancies                 3.845052
Glucose                   120.894531
BloodPressure              69.105469
SkinThickness              20.536458
Insulin                    79.799479
BMI                        31.992578
DiabetesPedigreeFunction    0.471876
Age                        33.240885
Outcome                     0.348958
dtype: float64

max is
 Pregnancies                17.00
Glucose                   199.00
BloodPressure             122.00
SkinThickness              99.00
Insulin                   846.00
BMI                        67.10
DiabetesPedigreeFunction    2.42
Age                        81.00
Outcome                     1.00
dtype: float64

min is
 Pregnancies                 0.000
Glucose                     0.000
BloodPressure               0.000
SkinThickness               0.000
Insulin                     0.000
BMI                         0.000
DiabetesPedigreeFunction    0.078
Age                        21.000
Outcome                     0.000
dtype: float64

standard deviation is
 Pregnancies                 3.369578
Glucose                    31.972618
BloodPressure              19.355807
SkinThickness              15.952218
Insulin                   115.244002
BMI                         7.884160
DiabetesPedigreeFunction    0.331329
Age                        11.760232
Outcome                     0.476951
dtype: float64
```

In [46]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',DIABETES_DATA.isnull(),'\n')
DIABETES_DATA['Pregnancies'].fillna('100',inplace=True)
DIABETES_DATA['Glucose'].fillna('60',inplace=True)
DIABETES_DATA['SkinThickness'].fillna('50',inplace=True)
DIABETES_DATA['Insulin'].fillna('100',inplace=True)
DIABETES_DATA['BMI'].fillna('30',inplace=True)
DIABETES_DATA['DiabetesPedigreeFunction'].fillna('0.351',inplace=True)
DIABETES_DATA['Age'].fillna('30',inplace=True)
DIABETES_DATA['Outcome'].fillna('1',inplace=True)
print('updated dataset',DIABETES_DATA)
```

```
checking null values
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0           False    False          False          False    False  False
1           False    False          False          False    False  False
2           False    False          False          False    False  False
3           False    False          False          False    False  False
4           False    False          False          False    False  False
..            ...      ...            ...            ...      ...    ...
763         False    False          False          False    False  False
764         False    False          False          False    False  False
765         False    False          False          False    False  False
766         False    False          False          False    False  False
767         False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        ...    ...      ...
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]

updated dataset      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0               6      148             72             35        0  33.6
1               1       85             66             29        0  26.6
2               8      183             64              0        0  23.3
3               1       89             66             23       94  28.1
4               0      137             40             35      168  43.1
..            ...      ...            ...            ...      ...   ...
763            10      101             76             48      180  32.9
764             2      122             70             27        0  36.8
765             5      121             72             23      112  26.2
766             1      126             60              0        0  30.1
767             1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

In [48]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',DIABETES_DATA.duplicated())
print()
DIABETES_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(DIABETES_DATA)
```

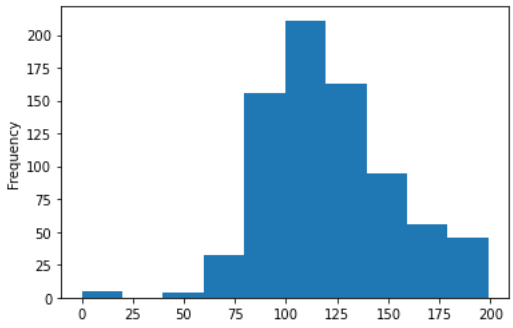
```
checking duplicate values
 0      False
1       False
2       False
3       False
4       False
        ...
763     False
764     False
765     False
766     False
767     False
Length: 768, dtype: bool

dataset after removing duplicate values

     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

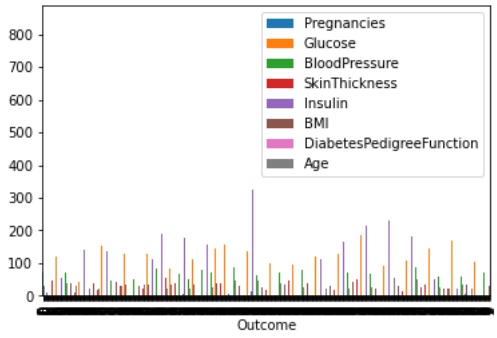
In [49]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
DIABETES_DATA.corr()
```

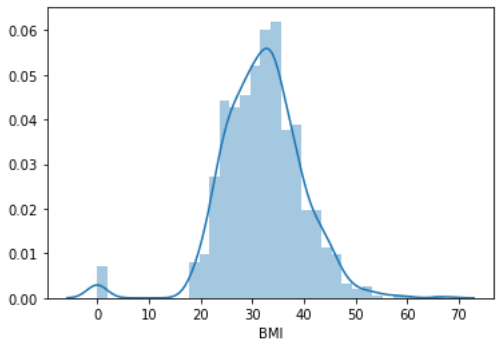
Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

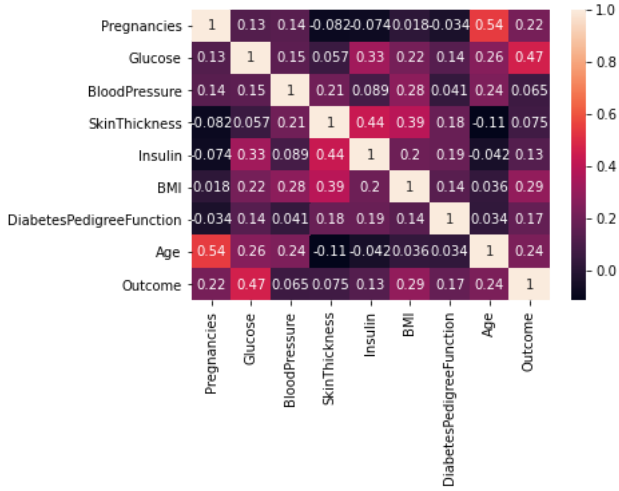
In [54]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
DIABETES_DATA['Glucose'].plot(kind = 'hist')
plt.show()
#bar
DIABETES_DATA.plot(kind = 'bar', x='Outcome')
plt.show()
#distplot
import seaborn as sns
sns.distplot(DIABETES_DATA['BMI'])
plt.show()
```







In [55]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(DIABETES_DATA.corr(),annot=True)
plt.show()
```



In [57]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
DIABETES_DATA.agg(['sum', 'min', 'max'])
```

Out[57]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
sum	2953	92847	53073	15772	61286	24570.3	362.401	25529	268
min	0	0	0	0	0	0.0	0.078	21	0
max	17	199	122	99	846	67.1	2.420	81	1

In [4]:
```
#group the dataset as per 'glucose' column
print('\ngrouping\n',DIABETES_DATA.groupby('Glucose'))
#display the number (count) of values for each 'Duration'
print('\nnumber (count) of values for each "Glucose"\n',DIABETES_DATA.groupby('Glucose').count())
#display the sum of all the values for each 'Duration'
print('\nsum of all the values for each 'Duration'\n',DIABETES_DATA.groupby('Glucose').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',DIABETES_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each 'Duration'\n',DIABETES_DATA.groupby('Glucose').agg(['sum', 'mean', 'median', 'min', 'max', 'pr
```

```
61               3  3.000000    3.0   3   3    3           82  82.000000
...            ...       ...    ...  ..  ..  ...          ...        ...
195             13  6.500000    6.5   6   7   42          140  70.000000
196             16  5.333333    7.0   1   8   56          242  80.666667
197             16  4.000000    3.0   2   8  128          284  71.000000
198              0  0.000000    0.0   0   0    0           66  66.000000
199              1  1.000000    1.0   1   1    1           76  76.000000

                  ...    Age                    Outcome                  \
        median min  ... median min max        prod     sum  mean median min
Glucose             ...
0         68.0  48  ...   22.0  21  41  15418788.0       2  0.40    0.0   0
44        62.0  62  ...   36.0  36  36        36.0       0  0.00    0.0   0
56        56.0  56  ...   22.0  22  22        22.0       0  0.00    0.0   0
57        70.0  60  ...   54.0  41  67      2747.0       0  0.00    0.0   0
61        82.0  82  ...   46.0  46  46        46.0       0  0.00    0.0   0
...        ...  ..  ...    ...  ..  ..         ...     ...   ...    ...  ..
195       70.0  70  ...   43.0  31  55      1705.0       2  1.00    1.0   1
196       76.0  76  ...   41.0  29  57     67773.0       3  1.00    1.0   1
197       70.0  70  ...   46.0  31  62   3972774.0       3  0.75    1.0   0
```

In [5]:
```
#DATASET 2 FROM HERE
```

In [10]:
```
import pandas as pd
#Loading the Dataset
HOUSING_DATA = pd.read_csv(r"C:\Users\MSI\Downloads\HousingPrice-DataSet.csv")
```

70

In [11]:
```
#Implement descriptive and summary statistics ( to calculate Count, Mean, Max and Min,Percentile, Variance and Standard Deviation)
#count
print('Count is\n',HOUSING_DATA.count(),'\n')
#mean
print('mean is\n',HOUSING_DATA.mean(),'\n')
#max
print('max is\n',HOUSING_DATA.max(),'\n')
#min
print('min is\n',HOUSING_DATA.min(),'\n')
#standard deviation
print('standard deviation is\n',HOUSING_DATA.std(),'\n')

#print(HOUSING_DATA.describe()) #can also use to get the required output
```

```
Count is
 CRIM       486
ZN         486
INDUS      486
CHAS       486
NOX        506
RM         506
AGE        486
DIS        506
RAD        506
TAX        506
PTRATIO    506
B          506
LSTAT      486
MEDV       506
dtype: int64

mean is
 CRIM         3.611874
ZN           11.211934
INDUS        11.083992
CHAS          0.069959
NOX           0.554695
RM            6.284634
AGE          68.518519
DIS           3.795043
RAD           9.549407
TAX         408.237154
PTRATIO      18.455534
B           356.674032
LSTAT        12.715432
MEDV         22.532806
dtype: float64

max is
 CRIM        88.9762
ZN         100.0000
INDUS       27.7400
CHAS         1.0000
NOX          0.8710
RM           8.7800
AGE        100.0000
DIS         12.1265
RAD         24.0000
TAX        711.0000
PTRATIO     22.0000
B          396.9000
LSTAT       37.9700
MEDV        50.0000
dtype: float64

min is
 CRIM         0.00632
ZN           0.00000
INDUS        0.46000
CHAS         0.00000
NOX          0.38500
RM           3.56100
AGE          2.90000
DIS          1.12960
RAD          1.00000
TAX        187.00000
PTRATIO     12.60000
B            0.32000
LSTAT        1.73000
MEDV         5.00000
dtype: float64

standard deviation is
 CRIM         8.720192
ZN          23.388876
INDUS        6.835896
CHAS         0.255340
NOX          0.115878
RM           0.702617
AGE         27.999513
DIS          2.105710
RAD          8.707259
TAX        168.537116
PTRATIO      2.164946
B           91.294864
LSTAT        7.155871
MEDV         9.197104
dtype: float64
```

In [14]:
```
#Check for the presence of missing values in the dataset and replace them with some valid numeric value
print('checking null values\n',HOUSING_DATA.isnull(),'\n')
HOUSING_DATA['CRIM'].fillna('0.654',inplace=True)
HOUSING_DATA['ZN'].fillna('10',inplace=True)
HOUSING_DATA['INDUS'].fillna('7.87',inplace=True)
HOUSING_DATA['CHAS'].fillna('0',inplace=True)
HOUSING_DATA['NOX'].fillna('0.524',inplace=True)
HOUSING_DATA['RM'].fillna('6.575',inplace=True)
HOUSING_DATA['AGE'].fillna('30',inplace=True)
print('updated dataset',HOUSING_DATA)
```

```
checking null values
       CRIM     ZN  INDUS   CHAS    NOX     RM    AGE    DIS    RAD    TAX  \
0    False  False  False  False  False  False  False  False  False  False
1    False  False  False  False  False  False  False  False  False  False
2    False  False  False  False  False  False  False  False  False  False
3    False  False  False  False  False  False  False  False  False  False
4    False  False  False  False  False  False  False  False  False  False
..     ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
501  False  False  False  False  False  False  False  False  False  False
502  False  False  False  False  False  False  False  False  False  False
503  False  False  False  False  False  False  False  False  False  False
504  False  False  False  False  False  False  False  False  False  False
505  False  False  False  False  False  False  False  False  False  False

     PTRATIO      B  LSTAT   MEDV
0      False  False  False  False
1      False  False  False  False
2      False  False  False  False
3      False  False  False  False
4      False  False   True  False
..       ...    ...    ...    ...
501    False  False   True  False
502    False  False  False  False
503    False  False  False  False
504    False  False  False  False
505    False  False  False  False

[506 rows x 14 columns]

updated dataset         CRIM  ZN  INDUS CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0    0.00632  18   2.31    0  0.538  6.575  65.2  4.0900    1  296     15.3
1    0.02731   0   7.07    0  0.469  6.421  78.9  4.9671    2  242     17.8
2    0.02729   0   7.07    0  0.469  7.185  61.1  4.9671    2  242     17.8
3    0.03237   0   2.18    0  0.458  6.998  45.8  6.0622    3  222     18.7
4    0.06905   0   2.18    0  0.458  7.147  54.2  6.0622    3  222     18.7
..       ...  ..    ...  ...    ...    ...   ...     ...  ...  ...      ...
501  0.06263   0  11.93    0  0.573  6.593  69.1  2.4786    1  273     21.0
502  0.04527   0  11.93    0  0.573  6.120  76.7  2.2875    1  273     21.0
503  0.06076   0  11.93    0  0.573  6.976    91  2.1675    1  273     21.0
504  0.10959   0  11.93    0  0.573  6.794  89.3  2.3889    1  273     21.0
505  0.04741   0  11.93    0  0.573  6.030    30  2.5050    1  273     21.0

          B  LSTAT  MEDV
0    396.90   4.98  24.0
1    396.90   9.14  21.6
2    392.83   4.03  34.7
3    394.63   2.94  33.4
4    396.90    NaN  36.2
..      ...    ...   ...
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9

[506 rows x 14 columns]
```

In [15]:
```
#Find and remove duplicate records (if any) in the dataset.
print('checking duplicate values\n',HOUSING_DATA.duplicated())
print()
HOUSING_DATA.drop_duplicates(inplace=True)
print('dataset after removing duplicate values\n')
print(HOUSING_DATA)
```

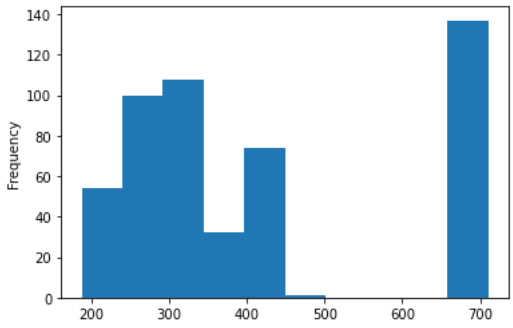
```
checking duplicate values
 0      False
1       False
2       False
3       False
4       False
        ...
501     False
502     False
503     False
504     False
505     False
Length: 506, dtype: bool

dataset after removing duplicate values

        CRIM  ZN  INDUS CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0    0.00632  18   2.31    0  0.538  6.575  65.2  4.0900    1  296     15.3
1    0.02731   0   7.07    0  0.469  6.421  78.9  4.9671    2  242     17.8
2    0.02729   0   7.07    0  0.469  7.185  61.1  4.9671    2  242     17.8
3    0.03237   0   2.18    0  0.458  6.998  45.8  6.0622    3  222     18.7
4    0.06905   0   2.18    0  0.458  7.147  54.2  6.0622    3  222     18.7
..       ...  ..    ...  ...    ...    ...   ...     ...  ...  ...      ...
501  0.06263   0  11.93    0  0.573  6.593  69.1  2.4786    1  273     21.0
502  0.04527   0  11.93    0  0.573  6.120  76.7  2.2875    1  273     21.0
503  0.06076   0  11.93    0  0.573  6.976    91  2.1675    1  273     21.0
504  0.10959   0  11.93    0  0.573  6.794  89.3  2.3889    1  273     21.0
505  0.04741   0  11.93    0  0.573  6.030    30  2.5050    1  273     21.0

          B  LSTAT  MEDV
0    396.90   4.98  24.0
1    396.90   9.14  21.6
2    392.83   4.03  34.7
3    394.63   2.94  33.4
4    396.90    NaN  36.2
..      ...    ...   ...
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9

[506 rows x 14 columns]
```

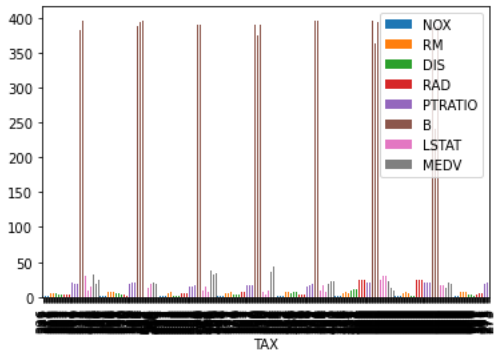
In [16]:
```
#Determine the correlation matrix for different columns (attributes) in a given dataset
HOUSING_DATA.corr()
```

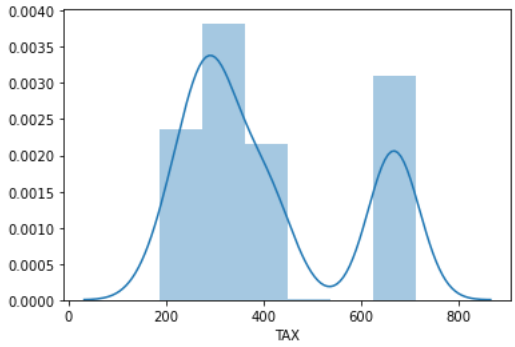
Out[16]:

	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
NOX	1.000000	-0.302188	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.582641	-0.427321
RM	-0.302188	1.000000	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.614339	0.695360
DIS	-0.769230	0.205246	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.493328	0.249929
RAD	0.611441	-0.209847	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.479541	-0.381626
TAX	0.668023	-0.292048	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.536110	-0.468536
PTRATIO	0.188933	-0.355501	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.375966	-0.507787
B	-0.380051	0.128069	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.369889	0.333461
LSTAT	0.582641	-0.614339	-0.493328	0.479541	0.536110	0.375966	-0.369889	1.000000	-0.735822
MEDV	-0.427321	0.695360	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.735822	1.000000

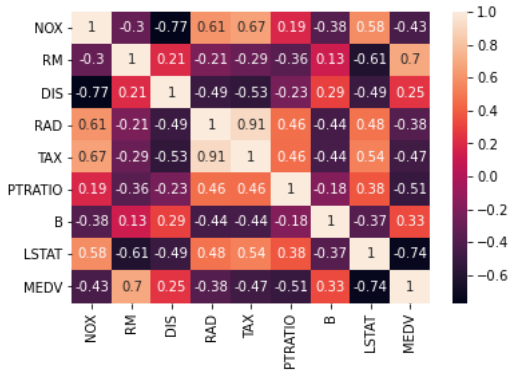
In [21]:
```
#Plot histogram, bar plot, distplot for various features attributes of the dataset
import matplotlib.pyplot as plt
#histogram
HOUSING_DATA['TAX'].plot(kind = 'hist')
plt.show()
#bar
HOUSING_DATA.plot(kind = 'bar', x='TAX')
plt.show()
#distplot
import seaborn as sns
sns.distplot(HOUSING_DATA['TAX'])
plt.show()
```







In [22]:
```
#Plot Heatmap for the correlation between different attributes in the dataset
sns.heatmap(HOUSING_DATA.corr(),annot=True)
plt.show()
```



In [23]:
```
#using agg(), aggregate function to calculate sum, min and max of each column
HOUSING_DATA.agg(['sum', 'min', 'max'])
```

Out[23]:

	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
sum	280.6757	3180.025	1920.2916	4832	206568	9338.5	180477.06	6179.70	11401.6
min	0.3850	3.561	1.1296	1	187	12.6	0.32	1.73	5.0
max	0.8710	8.780	12.1265	24	711	22.0	396.90	37.97	50.0

In [24]:
```
#group the dataset as per 'TAX‘ column
print('\ngrouping\n',HOUSING_DATA.groupby('TAX'))
#display the number (count) of values for each ‘TAX’
print('\nnumber (count) of values for each "TAX"\n',HOUSING_DATA.groupby('TAX').count())
#display the sum of all the values for each ‘TAX’
print('\nsum of all the values for each ‘TAX’\n',HOUSING_DATA.groupby('TAX').sum())
#perform various Data Aggregation functions
print('\nvarious Data Aggregation functions\n',HOUSING_DATA.agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
#perform various Data Aggregation functions for a particular column(attribute)
print('\nsum of all the values for each ‘TAX’\n',HOUSING_DATA.groupby('TAX').agg(['sum', 'mean', 'median', 'min', 'max', 'prod']))
```

```
grouping
 <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002765CE8B6D0>

number (count) of values for each "TAX"
      CRIM   ZN  INDUS  CHAS  NOX   RM  AGE  DIS  RAD  PTRATIO    B  LSTAT  \
TAX
187      1    1      1     1    1    1    1    1    1        1    1      1
188      7    7      7     7    7    7    7    7    7        7    7      7
193      8    8      8     8    8    8    8    8    8        8    8      8
198      1    1      1     1    1    1    1    1    1        1    1      1
216      5    5      5     5    5    5    5    5    5        5    5      5
..     ...  ...    ...   ...  ...  ...  ...  ...  ...      ...  ...    ...
432      9    9      9     9    9    9    9    9    9        9    9      8
437     15   15     15    15   15   15   15   15   15       15   15     15
469      1    1      1     1    1    1    1    1    1        1    1      1
666    132  132    132   132  132  132  132  132  132      132  132    126
711      5    5      5     5    5    5    5    5    5        5    5      5

     MEDV
```

LAB 7: TIME-SERIES DATA ANALYSIS:
Analysis of Time-dependent data to predict Future Trends from Past Valu
STATS MODEL- PYTHON LIBRARY

1. Perform the following operations on the given Time-series dataset **AirPassengers.csv**

AirPassengers

	A	B
1	Month	#Passengers
2	1949-01	112
3	1949-02	118
4	1949-03	132
5	1949-04	129
6	1949-05	121
7	1949-06	135
8	1949-07	148
9	1949-08	148
10	1949-09	136
11	1949-10	119
12	1949-11	104
13	1949-12	118
14	1950-01	115
15	1950-02	126
16	1950-03	141
17	1950-04	135
18	1950-05	125
19	1950-06	149
20	1950-07	170

- Reading Time-series data into Pandas dataframe
- PLOTTING TIME-SERIES DATA using plot() function
- ETS Decomposition-MULTIPLICATIVE MODEL
    - Extracting 'TREND' component
    - Extracting 'SEASONAL' component
    - Extracting 'RESIDUAL' component
- ETS Decomposition-ADDITIVE MODEL
    - Extracting 'TREND' component
    - Extracting 'SEASONAL' component
    - Extracting 'RESIDUAL' component

2. Download **SuperStore-SalesDataSet** using the link given below:
https://www.kaggle.com/datasets/rohitsahoo/sales-forecasting
Perform similar operations on the dataset as performed on **AirPassengers.csv** dataset

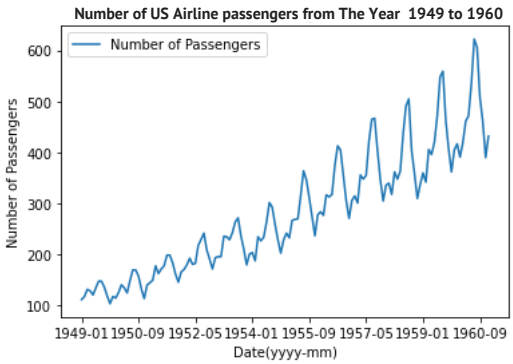
LAB 7

In [1]:
```
#Name: Harsh Arora
#Roll Number:AE-1218
#Course: BSC.(HONS.) COMPUTER SCIENCE
```

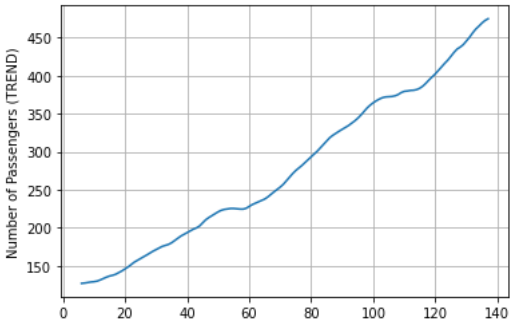
In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose as sm
```

In [3]:
```
Passenger_Data= pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
Passenger_Data.columns=['Date(yyyy-mm)','Number of The Passengers']
Passenger_Data.plot(x='Date(yyyy-mm)', y='Number of The Passengers',title= 'Number of US Airline passengers from The Year  1949 to 1960')
plt.ylabel('Number of The  Passengers')
```

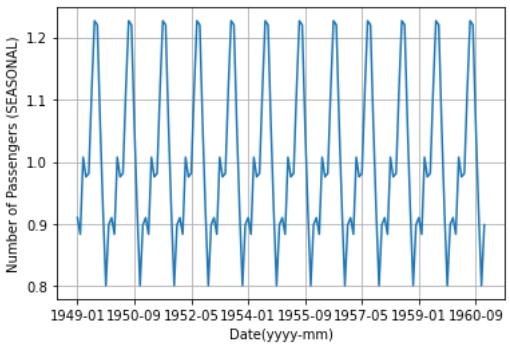
Out[3]:
```
Text(0, 0.5, 'Number of The  Passengers')
```



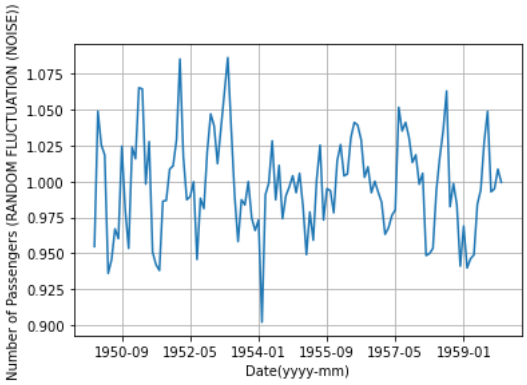
In [5]:
```
#3. Data Representation of ETS Decomposition-MULTIPLICATIVE MODEL EXTRACTING 'TREND' COMPONENT

Passenger_Data= pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
Passenger_Data.columns=['Date(yyyy-mm)','Number of The Passengers']

# Multiplicative Decomposition of The DATA

multiplicative_decomposition =sm(PassengerData['Number of Passengers'],model='multiplicative', period=12)
multiplicative_decomposition.trend.plot() # TREND component of the plot indicates GROWING OR DECAYING TREND
plt.ylabel('Number of The Passengers (TREND)')
plt.grid()
```



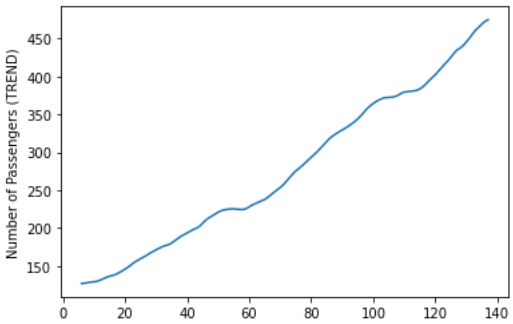
In [8]:
```
#EXTRACTING 'SEASONAL' COMPONENT- MULTIPLICATIVE MODEL
Passenger_Data= pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
Passenger_Data.columns=['Date(yyyy-mm)','Number of The Passengers']
Passenger_Data.set_index('Date(yyyy-mm)',inplace= True  )
#Graph of  Multiplicative Decomposition
multiplicative_decomposition =sm(PassengerData['Number of The  Passengers'],model='multiplicative', period=12)
multiplicative_decomposition.seasonal.plot()
plt.ylabel('Number of The Passengers (SEASONAL)')
plt.grid()
```



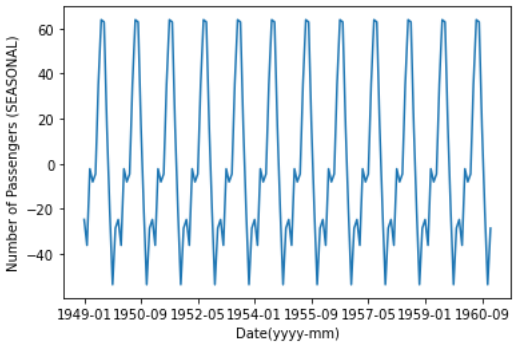
In [9]:
```
#EXTRACTING The 'RESIDUAL' COMPONENT- MULTIPLICATIVE MODEL
Passenger_Data= pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
Passenger_Data.columns=['Date(yyyy-mm)','Number of The Passengers']
Passenger_Data.set_index('Date(yyyy-mm)',inplace=True)
# Graph of Multiplicative Decomposition
multiplicative_decomposition = sm(PassengerData['Number of  Passengers], model='multiplicative', period=24)
multiplicative_decomposition.resid.plot()
plt.ylabel('Number of Passengers (RANDOM FLUCTUATION (NOISE))')
plt.grid()
```



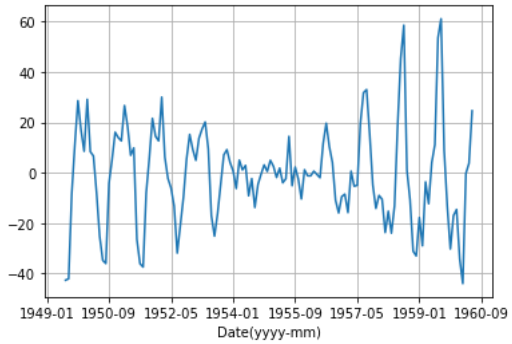
In [10]:
```
#EXTRACTING 'TREND' COMPONENT- ADDITIVE MODEL
Passenger_Data= pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
Passenger_Data.columns=['Date(yyyy-mm)','Number of Passengers']
# Graph  oF Additive Decomposition
Additive_decomposition = sm(PassengerData['Number of Passengers'],model='additive', period=12)
Additive_decomposition.trend.plot()
plt.ylabel('Number of ThePassengers (TREND)')
plt.show()
```



In [12]:
```
#EXTRACTING 'SEASONAL' COMPONENT- ADDITIVE MODEL
PassengerData = pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
PassengerData.columns=['Date(yyyy-mm)','Number of Passengers']
PassengerData.set_index('Date(yyyy-mm)',inplace=True)
# Additive Decomposition
Additive_decomposition = sm(PassengerData['Number of Passengers'],model='additive', period=12)
Additive_decomposition.seasonal.plot()
plt.ylabel('Number of Passengers (SEASONAL)')
plt.show()
```



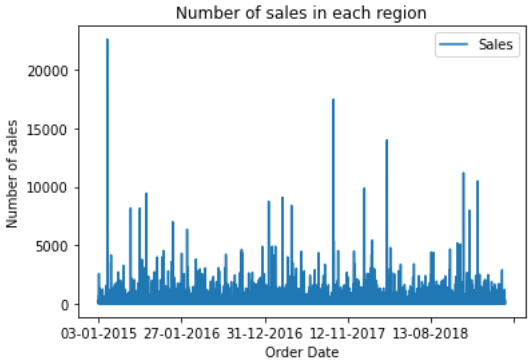
In [13]:
```
#EXTRACTING 'RESIDUAL' COMPONENT- ADDITIVE MODEL
PassengerData = pd.read_csv(r"C:\Users\MSI\Downloads\AirPassengers.csv")
PassengerData.columns=['Date(yyyy-mm)','Number of Passengers']
PassengerData.set_index('Date(yyyy-mm)',inplace=True)
# Additive Decomposition
Additive_decomposition = sm(PassengerData['Number of Passengers'], model='additive', period=12)
Additive_decomposition.resid.plot()
plt.grid()
plt.show()
```



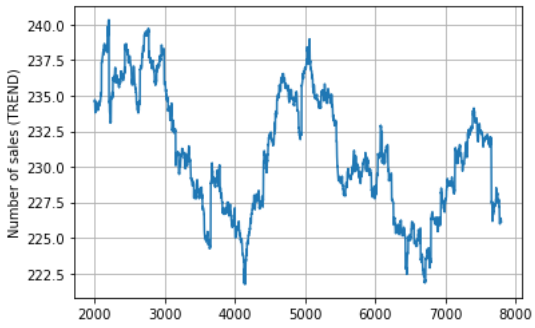
In [15]:
```
#DataSet Number 2:
```

In [17]:
```
sales_Data= pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")
salesData.plot(x='Order Date', y='Sales',title='Number of sales in each region')
plt.ylabel('Number of sales')
```

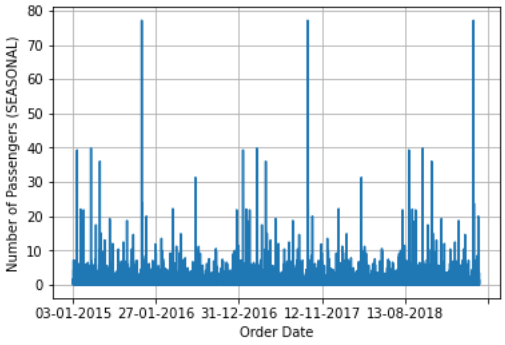
Out[17]:
```
Text(0, 0.5, 'Number of sales')
```



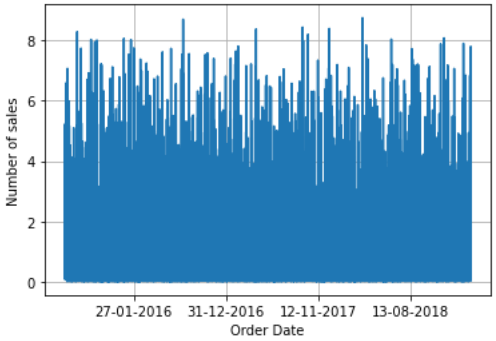
In [18]:
```
#3. ETS Decomposition-MULTIPLICATIVE MODEL EXTRACTING 'TREND' COMPONENT

salesData = pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")

# Graph of Multiplicative Decomposition

multiplicative_decomposition =sm(salesData['Sales'],model='multiplicative', period=4000)
multiplicative_decomposition.trend.plot() # TREND component of the plot indicates GROWING OR DECAYING TREND
plt.ylabel('Number of sales (TREND)')
plt.grid()
```



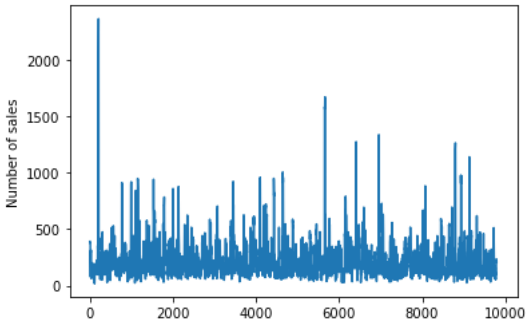
In [19]:
```
#EXTRACTING 'SEASONAL' COMPONENT- MULTIPLICATIVE MODEL
salesData = pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")
salesData.set_index('Order Date',inplace=True)
# Graph of Multiplicative Decomposition
multiplicative_decomposition =sm(salesData['Sales'],model='multiplicative', period=4000)
multiplicative_decomposition.seasonal.plot()
plt.ylabel('Number of Passengers (SEASONAL)')
plt.grid()
```



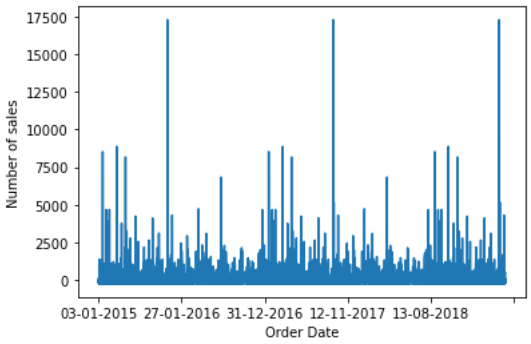
In [20]:
```
#EXTRACTING 'RESIDUAL' COMPONENT- MULTIPLICATIVE MODEL
salesData = pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")
salesData.set_index('Order Date',inplace=True)
# Graph OF Multiplicative Decomposition
multiplicative_decomposition = sm(salesData['Sales'], model='multiplicative', period=1000)
multiplicative_decomposition.resid.plot()
plt.ylabel('Number of sales')
plt.grid()
```



In [22]:
```
#EXTRACTING 'TREND' COMPONENT- ADDITIVE MODEL
salesData = pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")

# Graph of Additive Decomposition
Additive_decomposition = sm(salesData['Sales'],model='additive', period=12)
Additive_decomposition.trend.plot()
plt.ylabel('Number of sales')
plt.show()
```



In [24]:
```
#EXTRACTING 'SEASONAL' COMPONENT- ADDITIVE MODEL
sales_Data= pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")
salesData.set_index('Order Date',inplace=True)
# Additive Decomposition
Additive_decomposition = sm(salesData['Sales'],model='additive', period=4000)
Additive_decomposition.seasonal.plot()
plt.ylabel('Number of sales')
plt.show()
```



In [25]:
```
#EXTRACTING 'RESIDUAL' COMPONENT- ADDITIVE MODEL
sales_Data= pd.read_csv(r"C:\Users\MSI\Downloads\SuperStore-SalesDataSet.csv")
salesData.set_index('Order Date',inplace=True)
# Additive Decomposition
Additive_decomposition = sm(salesData['Sales'], model='additive', period=1000)
Additive_decomposition.resid.plot()
plt.grid()
plt.show()
```