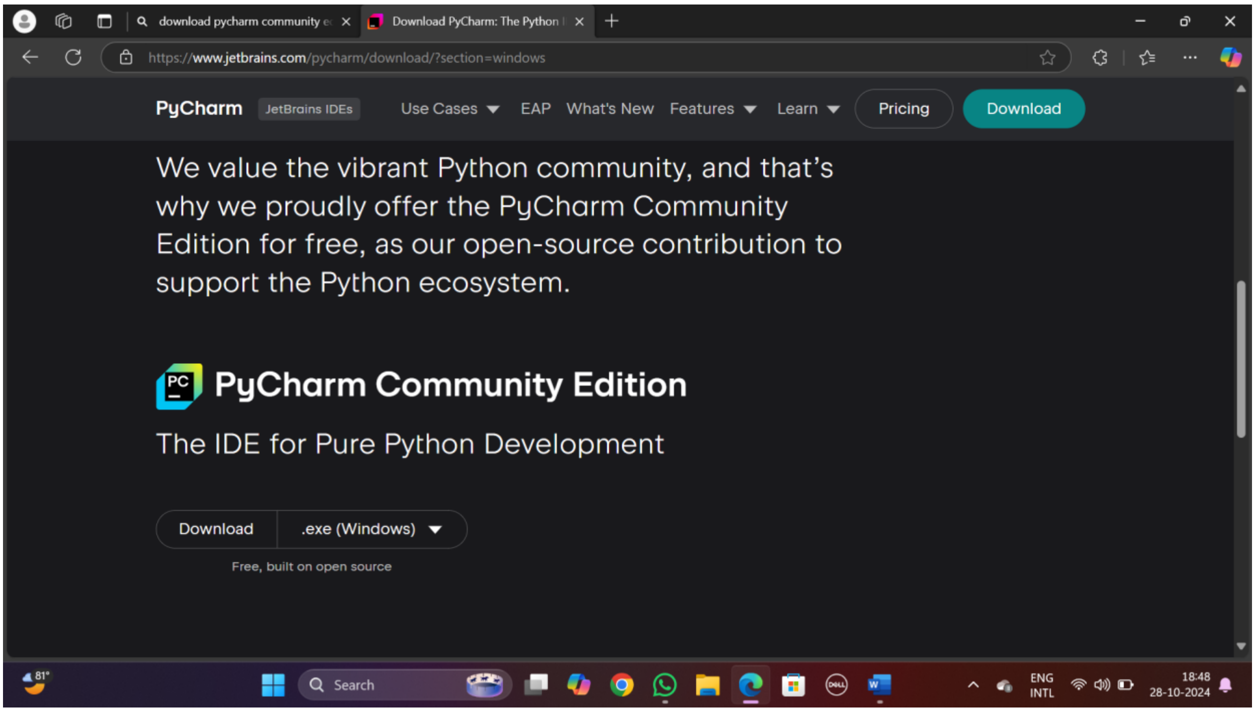
## EX.NO:1(A)

## Installation of PyCharm in Windows:



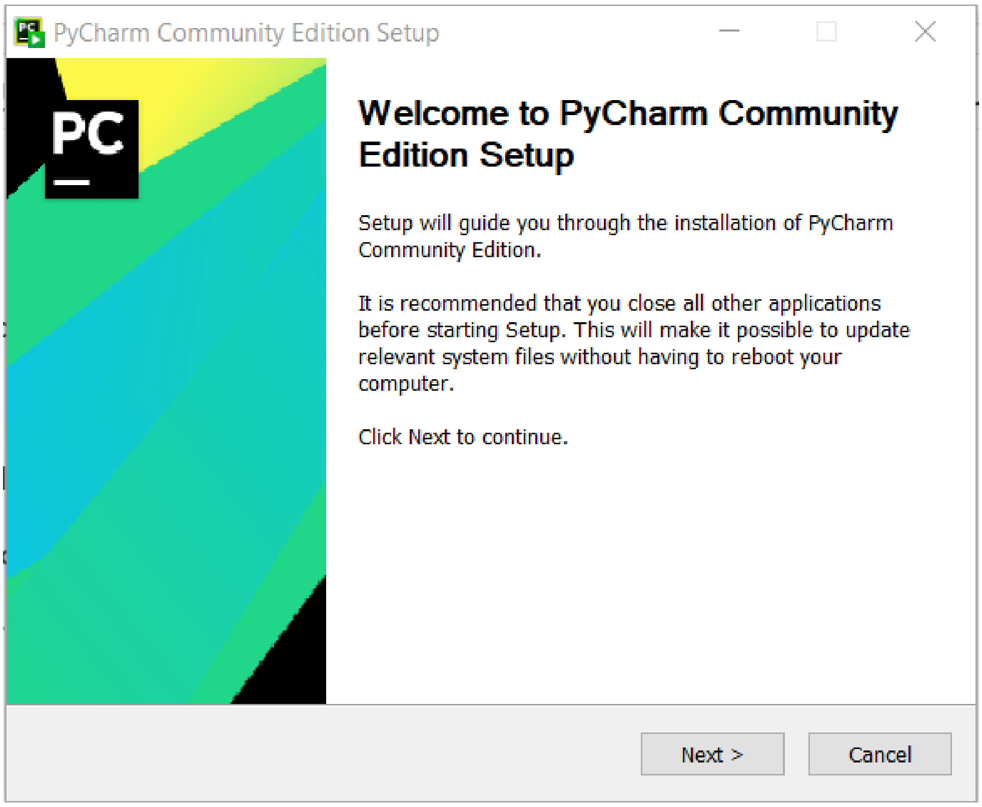
We value the vibrant Python community, and that's why we proudly offer the PyCharm Community Edition for free, as our open-source contribution to support the Python ecosystem.

PyCharm Community Edition

The IDE for Pure Python Development

Free, built on open source

## Run the Installer:



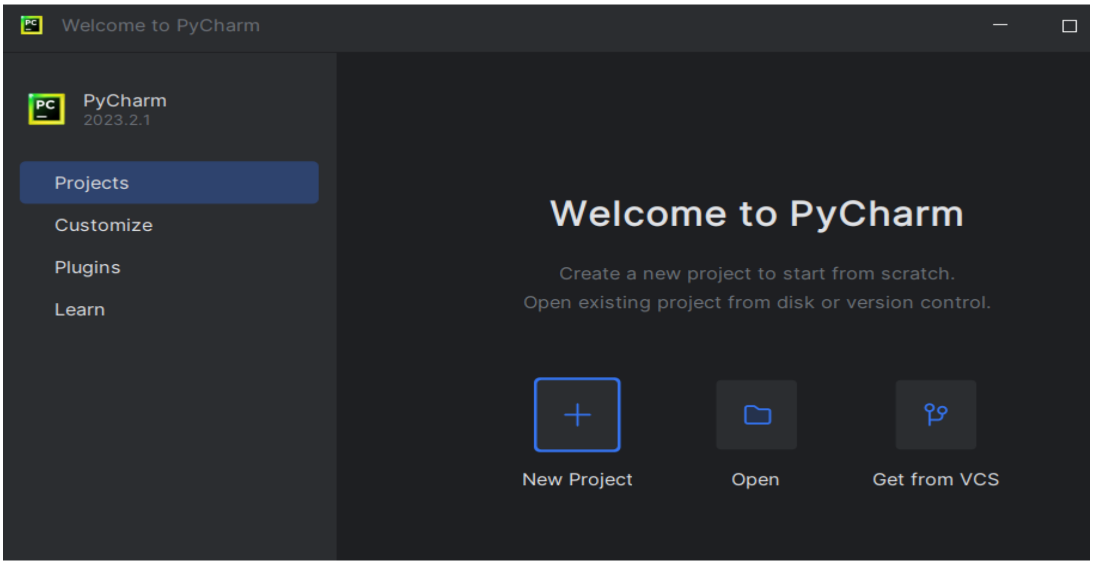
Welcome to PyCharm Community Edition Setup

Setup will guide you through the installation of PyCharm Community Edition.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

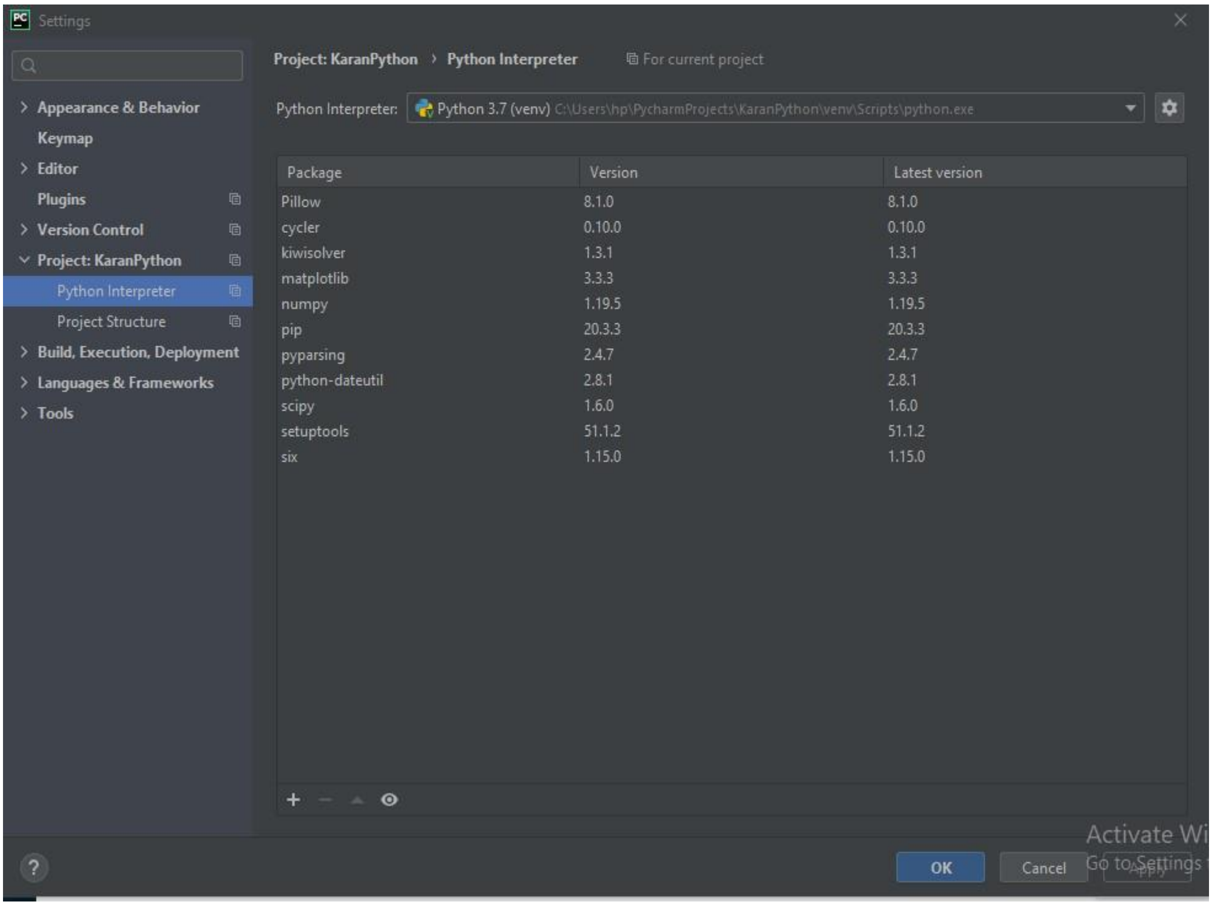
## Launch PyCharm:



Welcome to PyCharm

Create a new project to start from scratch.
Open existing project from disk or version control.

## Python Package Installation:



Python Interpreter: Python 3.7 (venv) C:\Users\hp\PycharmProjects\KaranPython\venv\Scripts\python.exe

| Package | Version | Latest version |
|---|---|---|
| Pillow | 8.1.0 | 8.1.0 |
| cycler | 0.10.0 | 0.10.0 |
| kiwisolver | 1.3.1 | 1.3.1 |
| matplotlib | 3.3.3 | 3.3.3 |
| numpy | 1.19.5 | 1.19.5 |
| pip | 20.3.3 | 20.3.3 |
| pyparsing | 2.4.7 | 2.4.7 |
| python-dateutil | 2.8.1 | 2.8.1 |
| scipy | 1.6.0 | 1.6.0 |
| setuptools | 51.1.2 | 51.1.2 |
| six | 1.15.0 | 1.15.0 |

**EX.NO:1(B)**

```
from scipy import special
import pandas as pd
import statsmodels.api as sm
from patsy import dmatrices
a, b, c = 5, 6, 7
s = (a + b + c) / 2
area = (s * (s - a) * (s - b) * (s - c)) ** 0.5
print('The area of the triangle is %0.2f' % area)
a = special.exp10(3)
print(a)
b = special.exp2(3)
print(b)
c = special.sindg(90)
print(c)
d = special.cosdg(45)
print(d)
data = pd.DataFrame({
    "x1": ["y", "x", "y", "x", "x", "y"],
    "x2": range(16, 22),
    "x3": range(1, 7),
    "x4": ["a", "b", "c", "d", "e", "f"],
    "x5": range(30, 24, -1)
})
print(data)
s1 = pd.Series([1, 3, 4, 5, 6, 2, 9])
s2 = pd.Series([1.1, 3.5, 4.7, 5.8, 2.9, 9.3])
s3 = pd.Series(['a', 'b', 'c', 'd', 'e'])
Data = {'first': s1, 'second': s2, 'third': s3}
dfseries = pd.DataFrame(Data)
print(dfseries)
```

```
df = sm.datasets.get_rdataset("Guerry", "HistData").data
vars = ['Department', 'Lottery', 'Literacy', 'Wealth', 'Region']
df = df[vars]
print(df.tail())
```

**OUTPUT:**

```
The area of the triangle is 14.70
1000.0
8.0
1.0
0.7071067811865475                          First  Second Third
x1  x2  x3 x4  x5                           0     1     1.1     a
0  y  16   1  a  30                         1     3     3.5     b
1  x  17   2  b  29                         2     4     4.7     c
2  y  18   3  c  28                         3     5     5.8     d
3  x  19   4  d  27                         4     6     2.9     e
4  x  20   5  e  26                         5     2     9.3   NaN
5  y  21   6  f  25                         6     9     NaN   NaN

     Department        Lottery  Literacy  Wealth   Region
379   L'Yonne              0.4     0.799     29.0      Centre
380   Vaucluse             0.4     0.799     29.0      Sud
381   Cantal               0.4     0.799     29.0      Sud
382   Tarn-et-Garonne      0.4     0.799     29.0      Sud
383   Meuse                0.4     0.799     29.0      Est
```

**EX.NO:2(A)**

```
import numpy as np
arr=np.array([[1,2,3],[4,2,5]])
print("array type is:",type(arr))
print("no of dimensions:",arr.ndim)
print("shape of array:",arr.shape)
print("size of array:",arr.size)
print("array store element of type:",arr.dtype)
```

**OUTPUT:**

```
array type is: <class 'numpy.ndarray'>
no of dimensions: 2
shape of array: (2, 3)
size of array: 6
array store element of type: int64
```

**EX.NO:2(B)**

```
import numpy as np
a = np.array([[1, 2, 3], [5, 8, 7]], dtype='float')
print("array created using passed list:", a)
b = np.array((1, 2, 3))
print("array created using passed tuple:", b)
c = np.zeros((3, 4))
print("an array is created with zero", c)
d = np.full((3, 3), 6, dtype='complex')
print("an array initialized with all 6s")
print("array type is complex:", d)
e = np.random.random((2, 2))
print("a random array:", e)
f = np.arange(0, 30, 5)
print("A sequential array with steps of 5:\n", f)
g = np.linspace(0, 5, 10)
print("A sequential array with 10 values between 0 and 5:\n", g)
arr = np.array([[1, 2, 3, 4], [5, 2, 4, 2], [1, 2, 0, 1]])
newarr = arr.reshape(2, 2, 3)
print("Original array:\n", arr)
print("Reshaped array:\n", newarr)
arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print("Original array:\n", arr)
print("Flattened array:\n", flarr)
```

**OUTPUT:**

```
array created using passed list: [[1. 2. 3.]
 [5. 8. 7.]]
array created using passed tuple: [1 2 3]
an array is created with zero [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
an array initialized with all 6s
array type is complex: [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]
a random array: [[0.12345678 0.87654321]
 [0.23456789 0.76543210]]
A sequential array with steps of 5:
 [ 0  5 10 15 20 25]
A sequential array with 10 values between 0 and 5:
 [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
 3.33333333 3.88888889 4.44444444 5.        ]
Original array:
 [[1 2 3 4]
 [5 2 4 2]
 [1 2 0 1]]
Reshaped array:
 [[[1 2 3]
  [4 5 2]]
 [[4 2 1]
  [2 0 1]]]
Original array:
 [[1 2 3]
 [4 5 6]]
Flattened array:
 [1 2 3 4 5 6]
```

**EX. NO:2(C)**

```
import numpy as np
array1 = np.array([[1, 2, 3], [4, 5, 6]])
array2 = np.array([[7, 8, 9], [10, 11, 12]])
print("Addition:")
print(array1 + array2)
print("Subtraction:")
print(array1 - array2)
print("Multiplication:")
print(array1 * array2)
print("Division:")
print(array2 / array1)
print("-" * 40)
print("Square:", array1 ** array2)
a = np.array([1, 2, 5, 3])
print("add 1 to every element:", a + 1)
print("sub 3 to every element:", a - 3)
print("multi 10 to every element:", a * 10)
print("Square each element:")
print(a ** 2)
a *= 2
print("Doubled each element of original array:", a)
a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])
print("Original array:\n", a)
print("Transpose of array:\n", a.T)
```

**OUTPUT:**

```
Addition:
[[ 8 10 12]
 [14 16 18]]
Subtraction:
[[-6 -6 -6]
 [-6 -6 -6]]
Multiplication:
[[ 7 16 27]
 [40 55 72]]
Division:
[[ 7.        4.        3.       ]
 [ 2.5       2.2       2.       ]]
----------------------------------------
Square: [[  1   4  27]
 [ 81 625 1296]]
add 1 to every element: [2 3 6 4]
sub 3 to every element: [-2 -1  2  0]
multi 10 to every element: [10 20 50 30]
Square each element:
[ 1  4 25  9]
Doubled each element of original array: [ 2  4 10  6]
Original array:
 [[1 2 3]
 [3 4 5]
 [9 6 0]]
Transpose of array:
 [[1 3 9]
 [2 4 6]
 [3 5 0]]
```

**EX. NO: 2(D)**

```
import numpy as np

a = np.array([[1, 4, 2], [3, 4, 6], [0, -1, 5]])

print("array element in sorted array:\n", np.sort(a, axis=None))

print("row-wise sorted array:\n", np.sort(a, axis=1))

print("Column wise sort by applying merge sort:\n", np.sort(a, axis=0, kind='mergesort'))

dtypes = [('name', 'U10'), ('grade&year', int), ('cgpa', float)]

values = [('Hrithick', 2009, 8.5), ('Ajay', 2008, 8.7), ('Pankaj', 2008, 7.9), ('Aakash', 2009, 9.0)]

arr = np.array(values, dtype=dtypes)

print("Array sorted by names:\n", np.sort(arr, order='name'))

print("Array sorted by graduation year and then cgpa:\n", np.sort(arr, order=['grade&year', 'cgpa']))
```

**OUTPUT:**

```
array element in sorted array:
 [-1  0  1  2  3  4  4  5  6]
row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column wise sort by applying merge sort:
 [[ 0  4  2]
 [ 1  4  5]
 [ 3  6  6]]
Array sorted by names:
 [('Aakash', 2009, 9.0) ('Ajay', 2008, 8.7) ('Hrithick', 2009, 8.5)
 ('Pankaj', 2008, 7.9)]
Array sorted by graduation year and then cgpa:
 [('Ajay', 2008, 8.7) ('Pankaj', 2008, 7.9) ('Aakash', 2009, 9.0)
 ('Hrithick', 2009, 8.5)]
```

**EX.NO:3(A)**

```
import pandas as pd
print("Empty dataframe")
a = pd.DataFrame()
print(a)
print("Dataframe creation using list")
lst = ['Geeks', 'For', 'Geeks', 'is', 'portal', 'for', 'Geeks']
df = pd.DataFrame(lst)
print(df)
data = {'Name': ['Tom', 'Nick', 'Krish'], 'Age': [20, 30, 40]}
a = pd.DataFrame(data)
print(a)
print("Create Dataframe from dictionary of lists")
data_dict = {'name': ["aparna", "pankaj", "sudhir", "Geeku"],
             'Degree': ["MBA", "BCA", "M.Tech", "MBA"],
             'Score': ["90", "40", "80", "98"]}
df = pd.DataFrame(data_dict)
print(df)
for i, j in df.iterrows():
    print(i, j)
    print()
```

**OUTPUT:**

```
Empty DataFrame
Columns: []
Index: []
Dataframe creation using list
       0
0  Geeks
1    For
2  Geeks
```

```
3     is
4  portal
5    for
6  Geeks
    Name  Age
0    Tom   20
1   Nick   30
2  Krish   40
Create Dataframe from dictionary of lists
     name  Degree Score
0  aparna     MBA    90
1  pankaj     BCA    40
2  sudhir  M.Tech    80
3   Geeku     MBA    98
0 name      aparna
  Degree       MBA
  Score         90
Name: 0, dtype: object
1 name      pankaj
  Degree       BCA
  Score         40
Name: 1, dtype: object
2 name      sudhir
  Degree    M.Tech
  Score         80
3 name      Geeku
  Degree       MBA
  Score         98
Name: 3, dtype: object
```

**EX.NO:3(B)**

import pandas as pd

url = 'https://github.com/chris1610/pbpython/blob/master/data/2018_Sales_Total_v2.xlsx?raw=True'

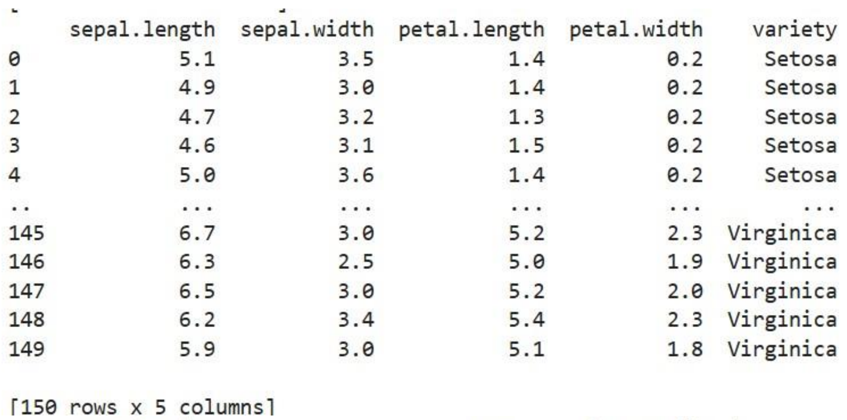
df = pd.read_excel(url)

print(df)

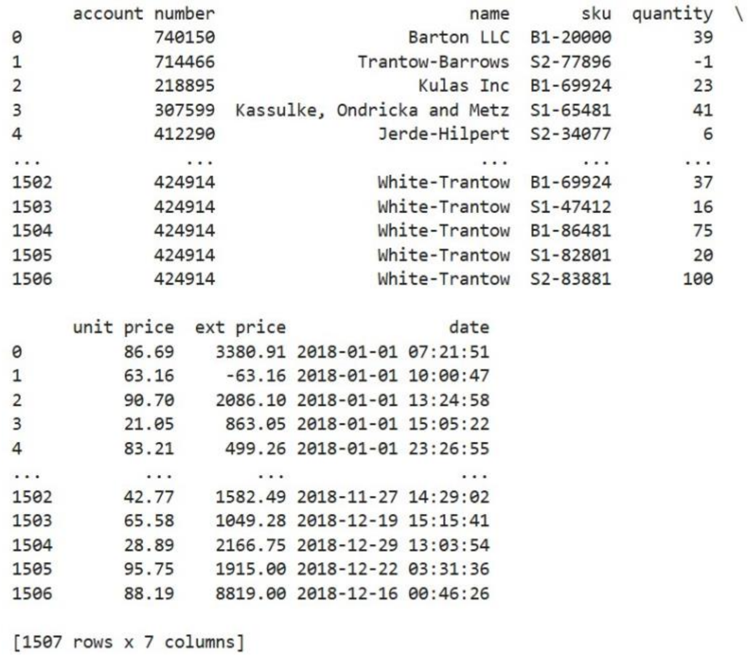
data = pd.read_csv(r'C:\Users\HI\Downloads\PythonDataScience Handbook- master notebooks\data\iris.csv')

df = pd.DataFrame(data)

**OUTPUT:**



```
     sepal.length  sepal.width  petal.length  petal.width    variety
0             5.1          3.5           1.4          0.2     Setosa
1             4.9          3.0           1.4          0.2     Setosa
2             4.7          3.2           1.3          0.2     Setosa
3             4.6          3.1           1.5          0.2     Setosa
4             5.0          3.6           1.4          0.2     Setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

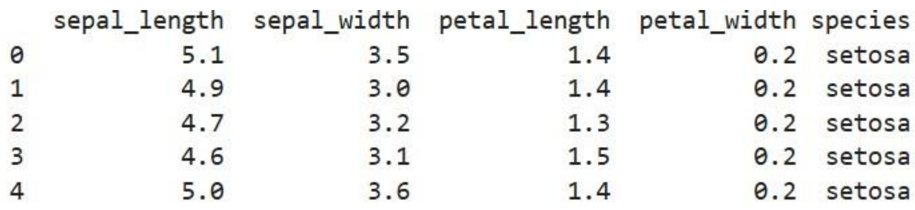


```
      account number                         name        sku  quantity  \
0             740150                   Barton LLC  B1-20000        39
1             714466              Trantow-Barrows  S2-77896        -1
2             218895                    Kulas Inc  B1-69924        23
3             307599  Kassulke, Ondricka and Metz  S1-65481        41
4             412290                Jerde-Hilpert  S2-34077         6
...              ...                          ...       ...       ...
1502          424914                White-Trantow  B1-69924        37
1503          424914                White-Trantow  S1-47412        16
1504          424914                White-Trantow  B1-86481        75
1505          424914                White-Trantow  S1-82801        20
1506          424914                White-Trantow  S2-83881       100

      unit price  ext price                date
0          86.69    3380.91 2018-01-01 07:21:51
1          63.16     -63.16 2018-01-01 10:00:47
2          90.70    2086.10 2018-01-01 13:24:58
3          21.05     863.05 2018-01-01 15:05:22
4          83.21     499.26 2018-01-01 23:26:55
...          ...        ...                 ...
1502       42.77    1582.49 2018-11-27 14:29:02
1503       65.58    1049.28 2018-12-19 15:15:41
1504       28.89    2166.75 2018-12-29 13:03:54
1505       95.75    1915.00 2018-12-22 03:31:36
1506       88.19    8819.00 2018-12-16 00:46:26

[1507 rows x 7 columns]
```

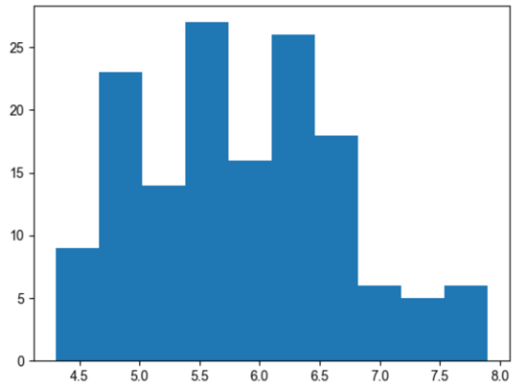
## EX.NO:4(A)

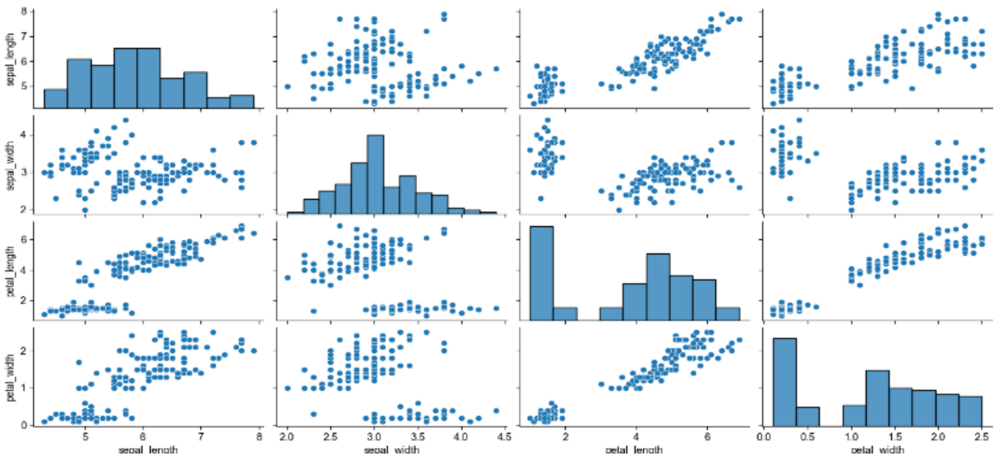
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset("iris")
my_data_frame = pd.DataFrame(iris)
print(my_data_frame.head())
plt.hist(my_data_frame.sepal_length)
sns.pairplot(my_data_frame)
sns.pairplot(iris, hue="species")
plt.show()
```

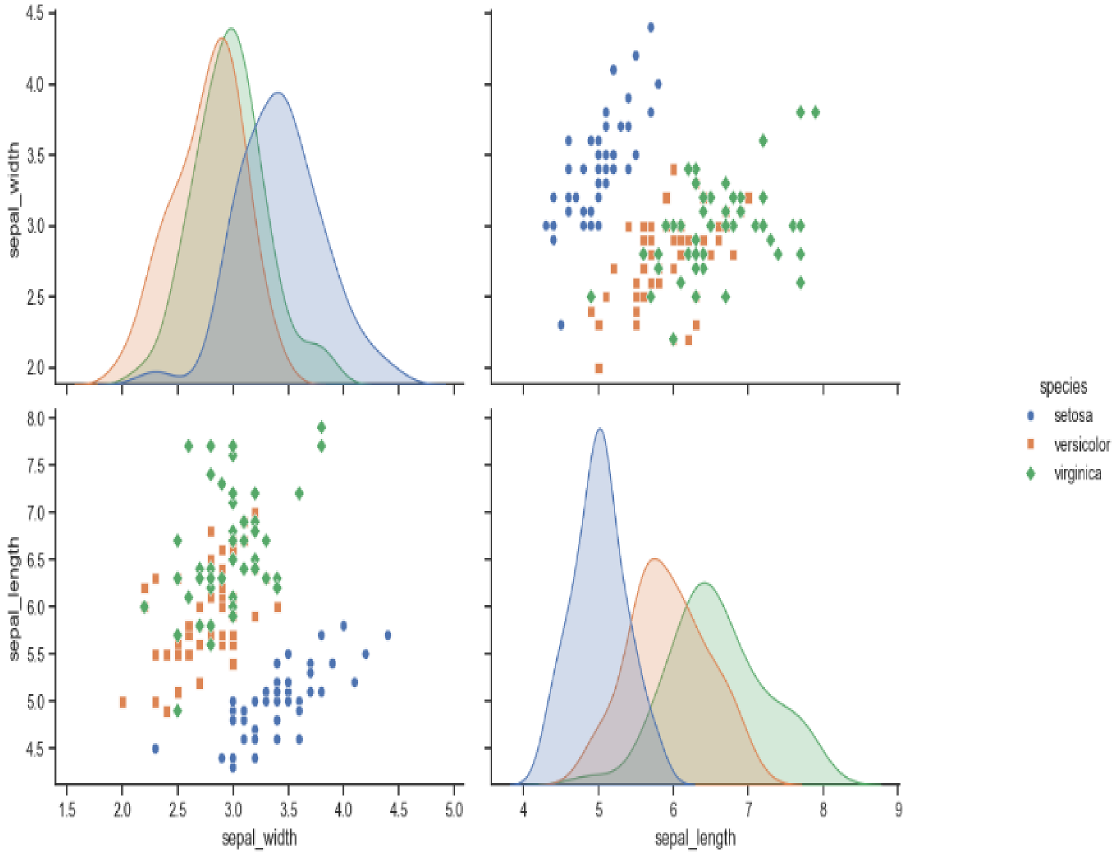
## OUTPUT:

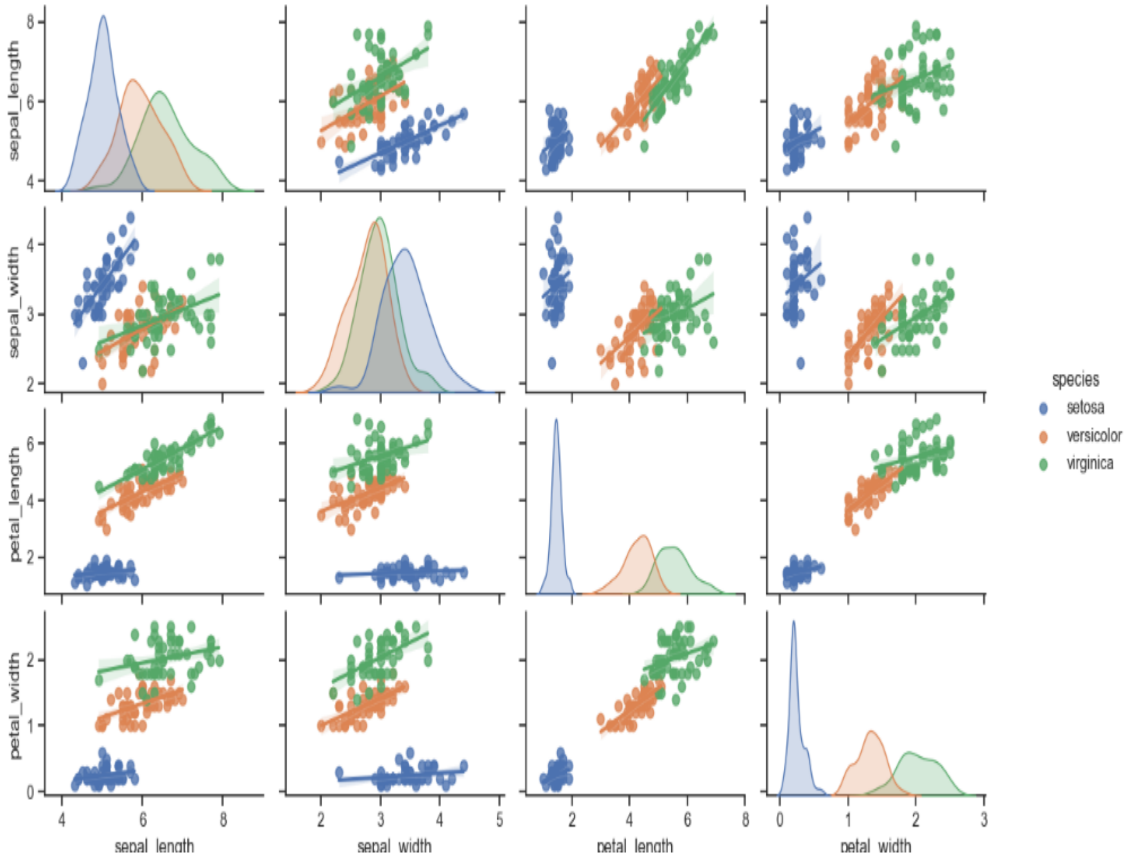


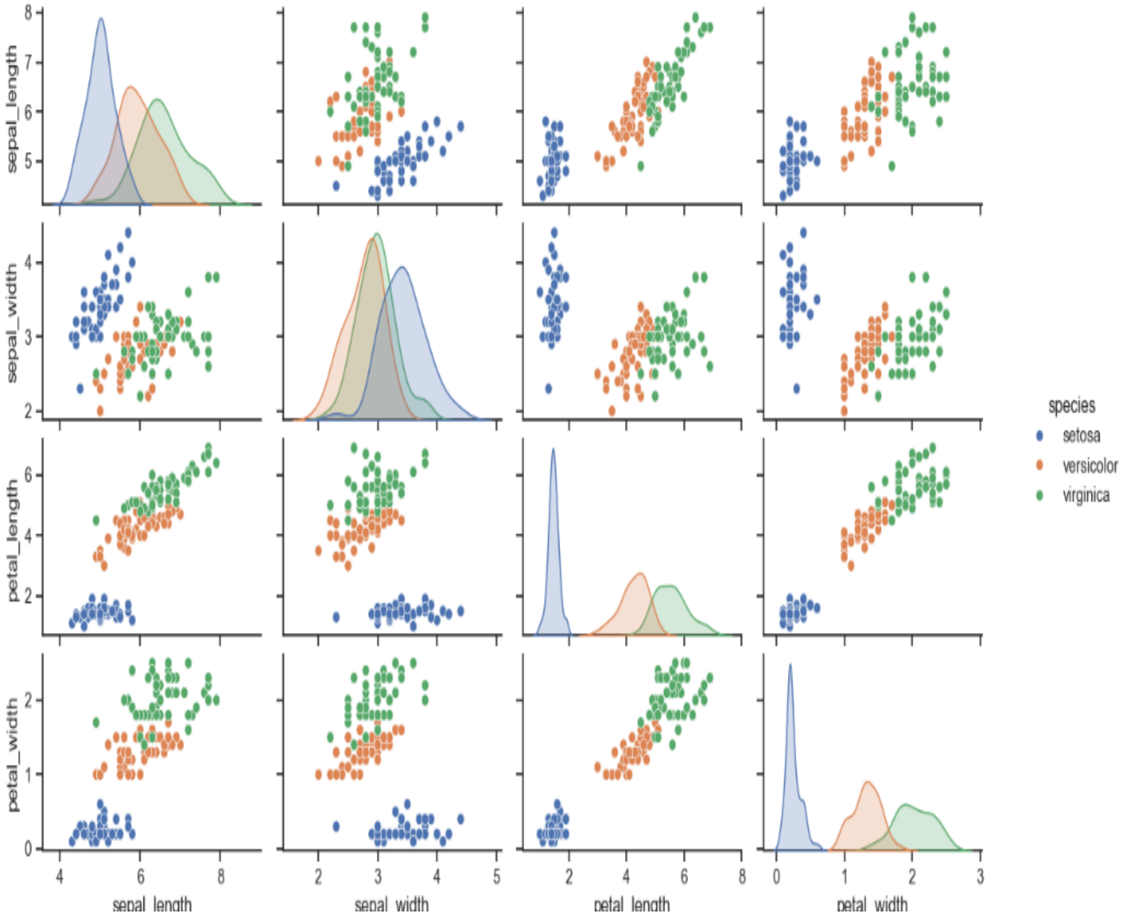
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

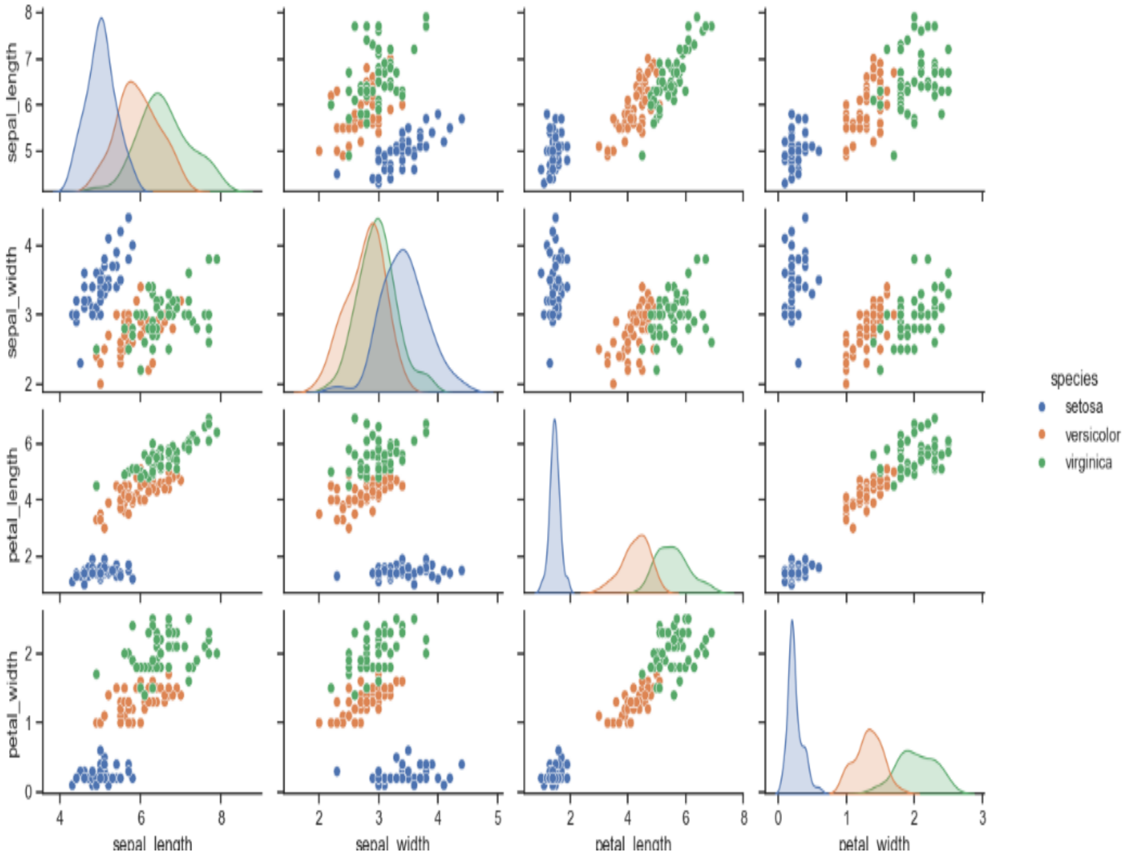





sepal_width
4.5
4.0
3.5
3.0
2.5
2.0
sepal_length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
sepal_width
4
5
6
7
8
9
sepal_length
species
setosa
versicolor
virginica


sepal_length
8
6
4
sepal_width
4
3
2
petal_length
6
4
2
petal_width
2
1
0
4
6
8
sepal_length
2
3
4
5
sepal_width
2
4
6
8
petal_length
0
1
2
3
petal_width
species
setosa
versicolor
virginica

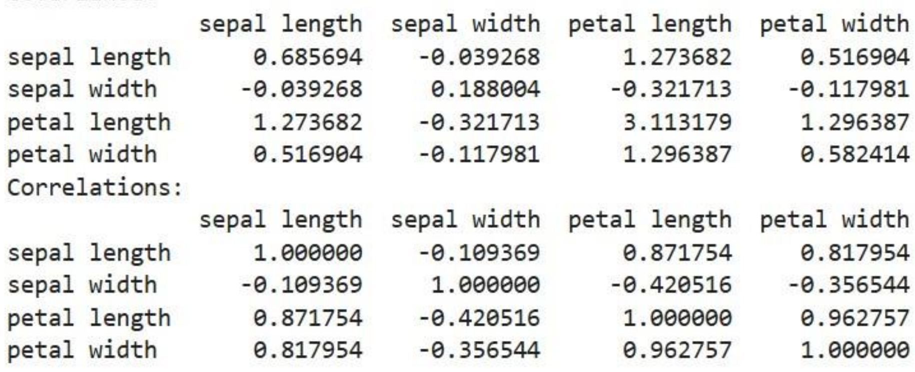

sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica
sepal_length
sepal_width
petal_length
petal_width


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica
sepal_length
sepal_width
petal_length
petal_width

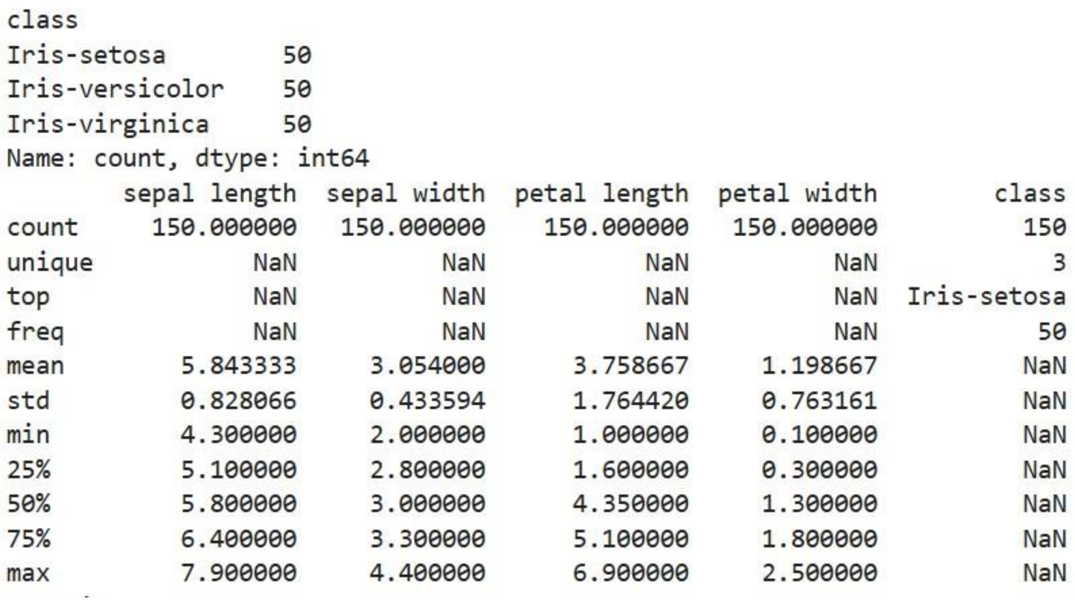
**EX.NO:4(B)**

```
import pandas as pd
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
data = pd.read_csv(url, header=None)
data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'class']
print(data.head())
from pandas.api.types import is_numeric_dtype
for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\tMean = %.2f' % data[col].mean())
        print('\tStandard deviation = %.2f' % data[col].std())
        print('\tMinimum = %.2f' % data[col].min())
        print('\tMaximum = %.2f' % data[col].max())
print(data['class'].value_counts())
print(data.describe(include='all'))
numeric_data = data.drop(columns=['class'])
print("Covariance:")
print(numeric_data.cov())
print('Correlations:')
print(numeric_data.corr())
```

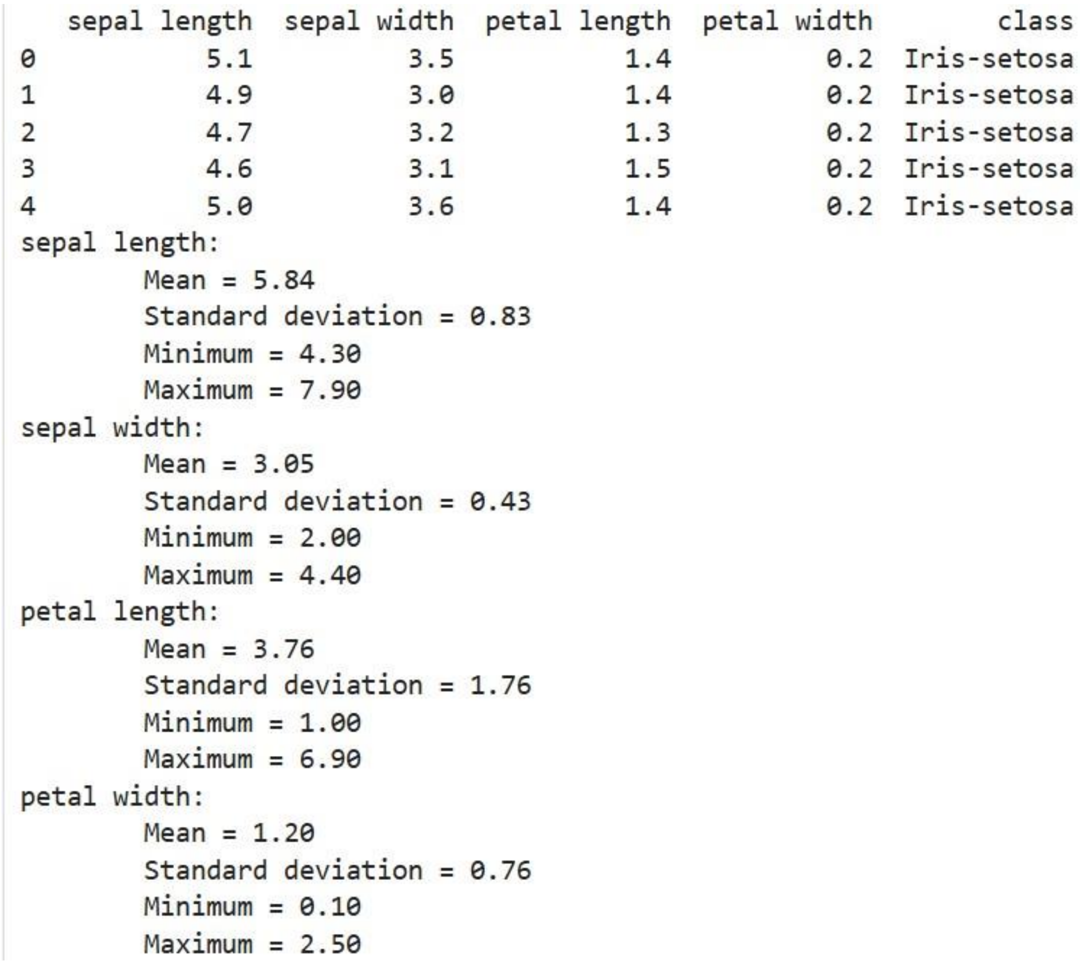
**OUTPUT:**



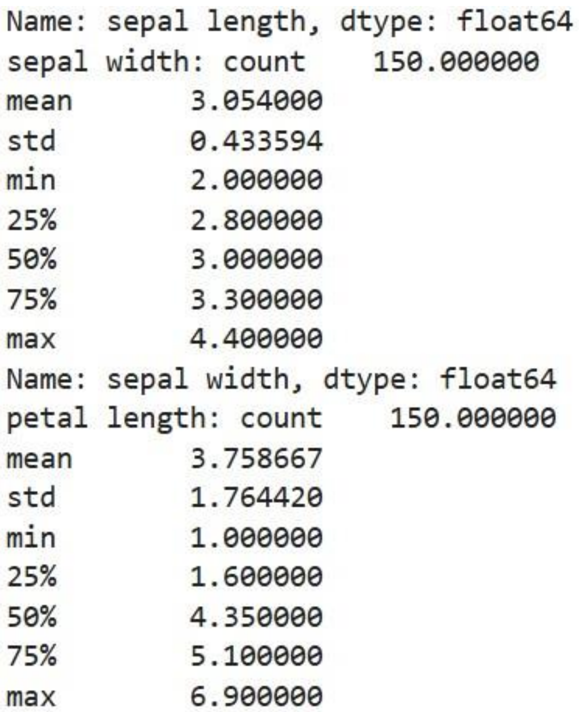
```
Covariance:
              sepal length  sepal width  petal length  petal width
sepal length      0.685694    -0.039268      1.273682     0.516904
sepal width      -0.039268     0.188004     -0.321713    -0.117981
petal length      1.273682    -0.321713      3.113179     1.296387
petal width       0.516904    -0.117981      1.296387     0.582414
Correlations:
              sepal length  sepal width  petal length  petal width
sepal length      1.000000    -0.109369      0.871754     0.817954
sepal width      -0.109369     1.000000     -0.420516    -0.356544
petal length      0.871754    -0.420516      1.000000     0.962757
petal width       0.817954    -0.356544      0.962757     1.000000
```



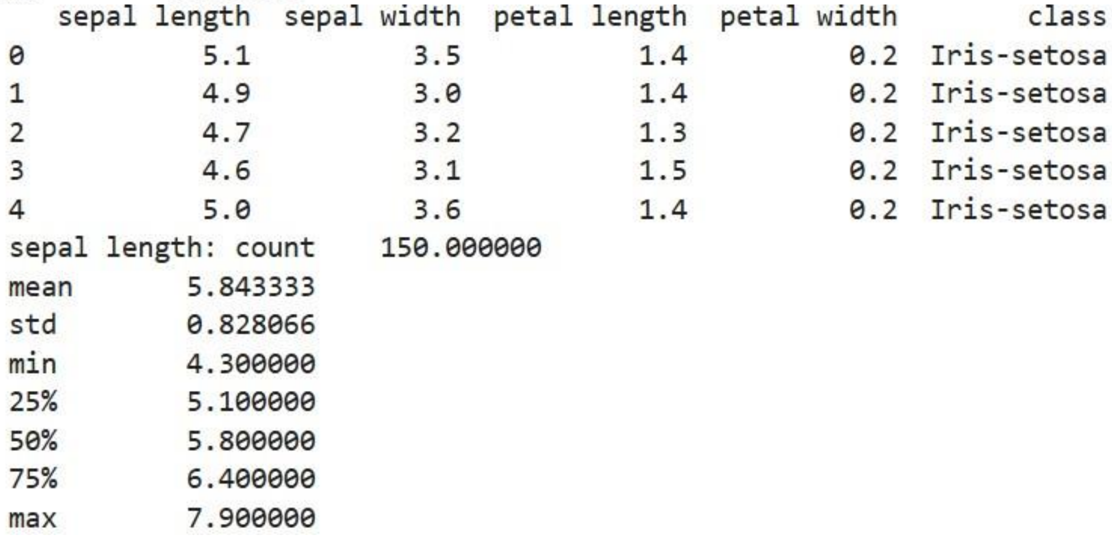
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
        sepal length  sepal width  petal length  petal width        class
count     150.000000   150.000000    150.000000   150.000000          150
unique           NaN          NaN           NaN          NaN            3
top              NaN          NaN           NaN          NaN  Iris-setosa
freq             NaN          NaN           NaN          NaN           50
mean        5.843333     3.054000      3.758667     1.198667          NaN
std         0.828066     0.433594      1.764420     0.763161          NaN
min         4.300000     2.000000      1.000000     0.100000          NaN
25%         5.100000     2.800000      1.600000     0.300000          NaN
50%         5.800000     3.000000      4.350000     1.300000          NaN
75%         6.400000     3.300000      5.100000     1.800000          NaN
max         7.900000     4.400000      6.900000     2.500000          NaN
```



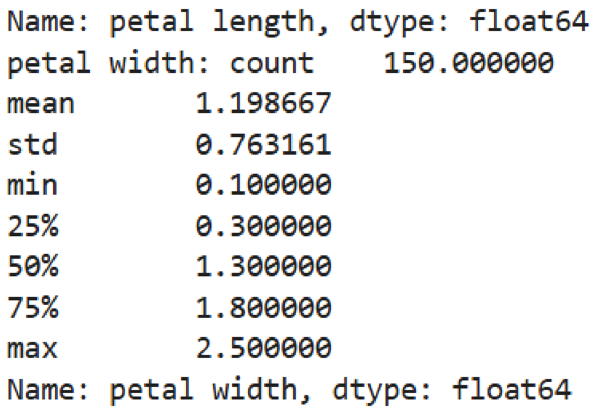
```
   sepal length  sepal width  petal length  petal width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
sepal length:
        Mean = 5.84
        Standard deviation = 0.83
        Minimum = 4.30
        Maximum = 7.90
sepal width:
        Mean = 3.05
        Standard deviation = 0.43
        Minimum = 2.00
        Maximum = 4.40
petal length:
        Mean = 3.76
        Standard deviation = 1.76
        Minimum = 1.00
        Maximum = 6.90
petal width:
        Mean = 1.20
        Standard deviation = 0.76
        Minimum = 0.10
        Maximum = 2.50
```



```
Name: sepal length, dtype: float64
sepal width: count    150.000000
mean          3.054000
std           0.433594
min           2.000000
25%           2.800000
50%           3.000000
75%           3.300000
max           4.400000
Name: sepal width, dtype: float64
petal length: count    150.000000
mean          3.758667
std           1.764420
min           1.000000
25%           1.600000
50%           4.350000
75%           5.100000
max           6.900000
```



```
   sepal length  sepal width  petal length  petal width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
sepal length: count    150.000000
mean          5.843333
std           0.828066
min           4.300000
25%           5.100000
50%           5.800000
75%           6.400000
max           7.900000
```

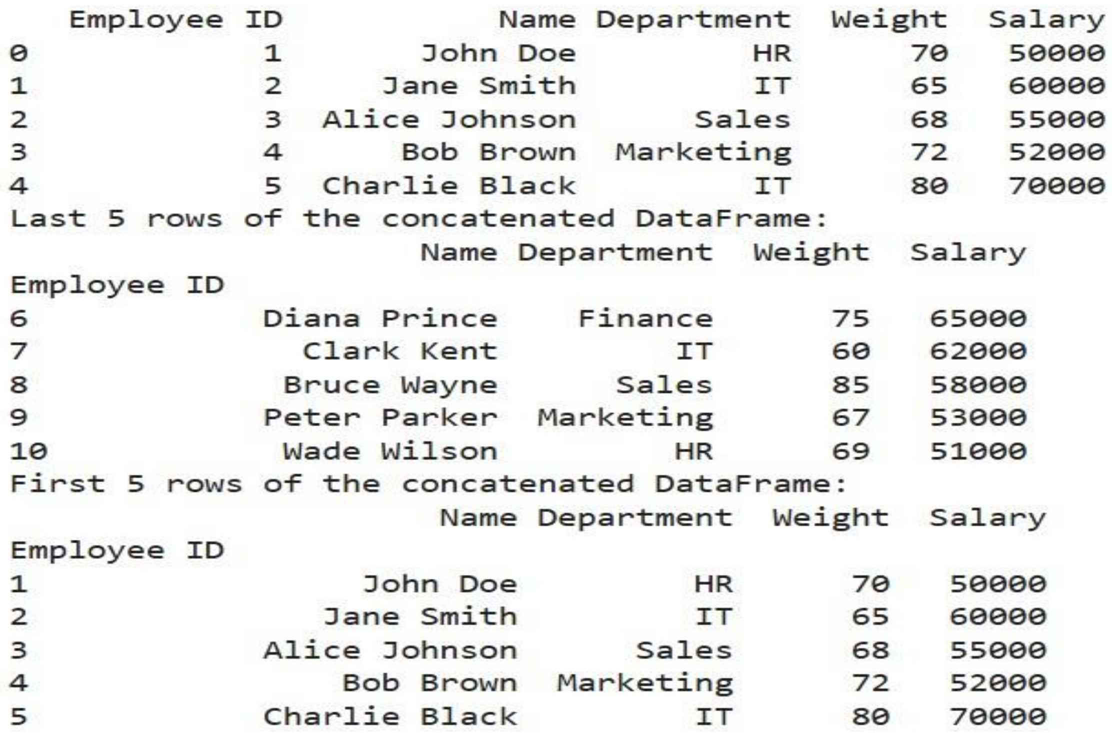


```
Name: petal length, dtype: float64
petal width: count    150.000000
mean          1.198667
std           0.763161
min           0.100000
25%           0.300000
50%           1.300000
75%           1.800000
max           2.500000
Name: petal width, dtype: float64
```

**EX. NO:4(C)**

```
import pandas as pd
file = r'C:\Users\Downloads\dept.xlsx'
df = pd.read_excel(file)
print(df)
sheet1 = pd.read_excel(file, sheet_name=0, index_col=0)
sheet2 = pd.read_excel(file, sheet_name=1, index_col=0)
newData = pd.concat([sheet1, sheet2])
print("Last 5 rows of the concatenated DataFrame:")
print(newData.tail())
print("First 5 rows of the concatenated DataFrame:")
print(newData.head())
print("Sorted column by 'Weight':")
sorted_data = newData.sort_values(['Weight'], ascending=True)
print(sorted_data.head(5))
print("Descriptive statistics of the DataFrame:")
print(newData.describe())
```

**OUTPUT:**

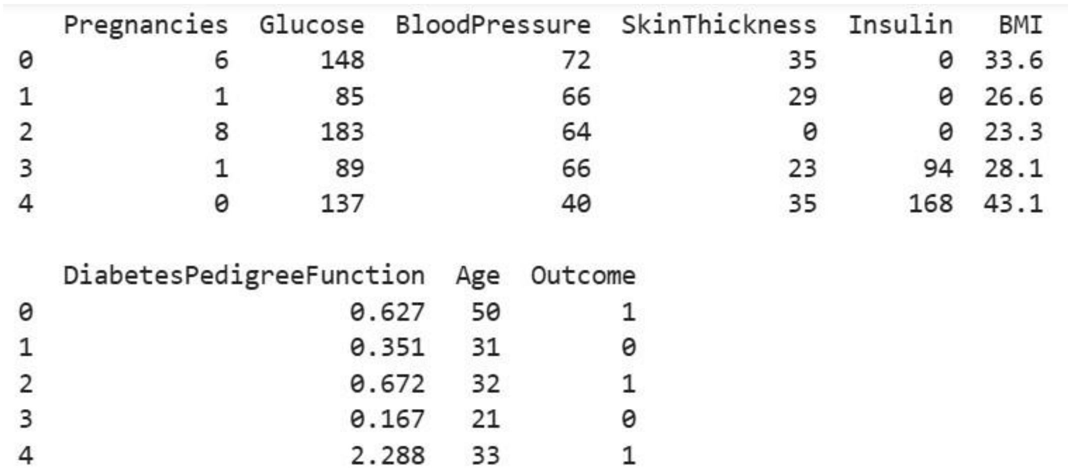


```
   Employee ID           Name Department  Weight  Salary
0            1       John Doe         HR      70   50000
1            2     Jane Smith         IT      65   60000
2            3  Alice Johnson      Sales      68   55000
3            4      Bob Brown  Marketing      72   52000
4            5  Charlie Black         IT      80   70000
Last 5 rows of the concatenated DataFrame:
                     Name Department  Weight  Salary
Employee ID
6            Diana Prince    Finance      75   65000
7              Clark Kent         IT      60   62000
8             Bruce Wayne      Sales      85   58000
9            Peter Parker  Marketing      67   53000
10            Wade Wilson         HR      69   51000
First 5 rows of the concatenated DataFrame:
                      Name Department  Weight  Salary
Employee ID
1                 John Doe         HR      70   50000
2               Jane Smith         IT      65   60000
3            Alice Johnson      Sales      68   55000
4                Bob Brown  Marketing      72   52000
5            Charlie Black         IT      80   70000
```

**EX.NO:5**

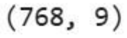
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import pickle
dataset = pd.read_csv(r"C:\Users\charl\Downloads\diabetes.csv")
print(dataset.head(), dataset.shape, dataset.describe(), dataset['Outcome'].value_counts(),
dataset.isna().sum(), sep='\n')
sns.countplot(x='Outcome', data=dataset)
plt.show(block=False)
sns.heatmap(dataset.corr(), annot=True)
plt.show(block=False)
x = dataset.drop(["Pregnancies", "BloodPressure", "SkinThickness", "Outcome"], axis=1)
y = dataset['Outcome']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
svc = SVC()
svc.fit(x_train, y_train)
pickle.dump(svc, open('classifier.pkl', 'wb'))
pickle.dump(sc, open('sc.pkl', 'wb'))

features = ["Glucose", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age"]
for feature in features:
    plt.figure(figsize=(16, 6))
    sns.histplot(dataset[feature][dataset["Outcome"] == 1], kde=True)
    plt.title(feature, fontsize=20)
    plt.show(block=False)
plt.show()
```

**OUTPUT:**

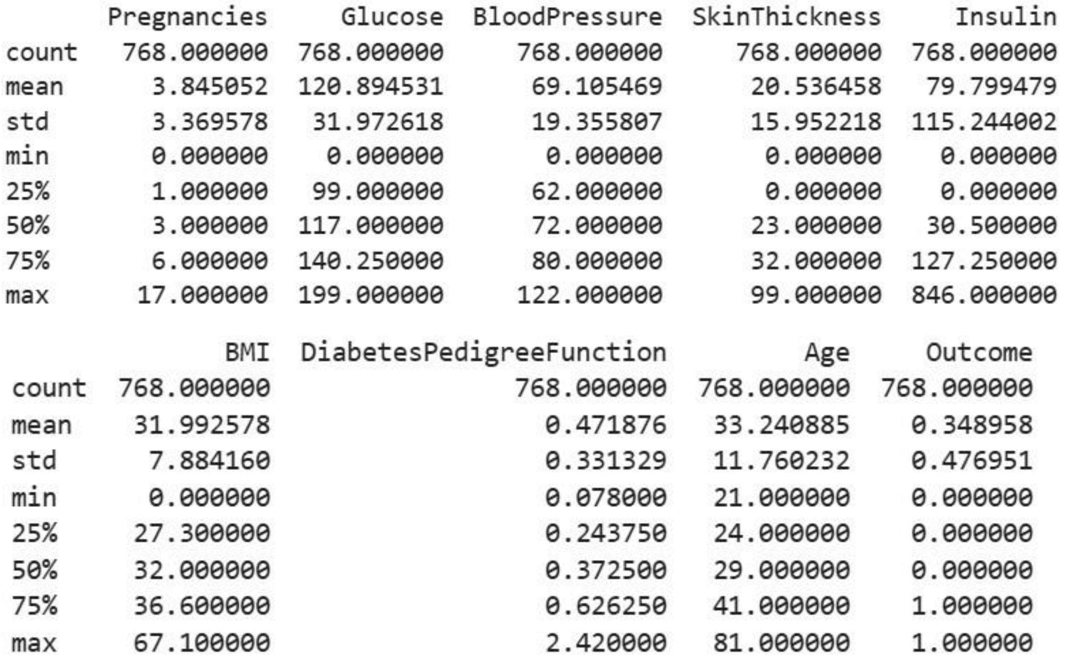


```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   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
```

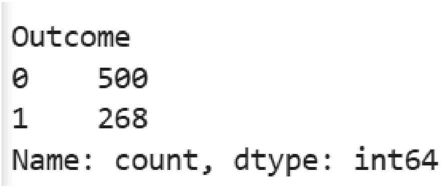
:



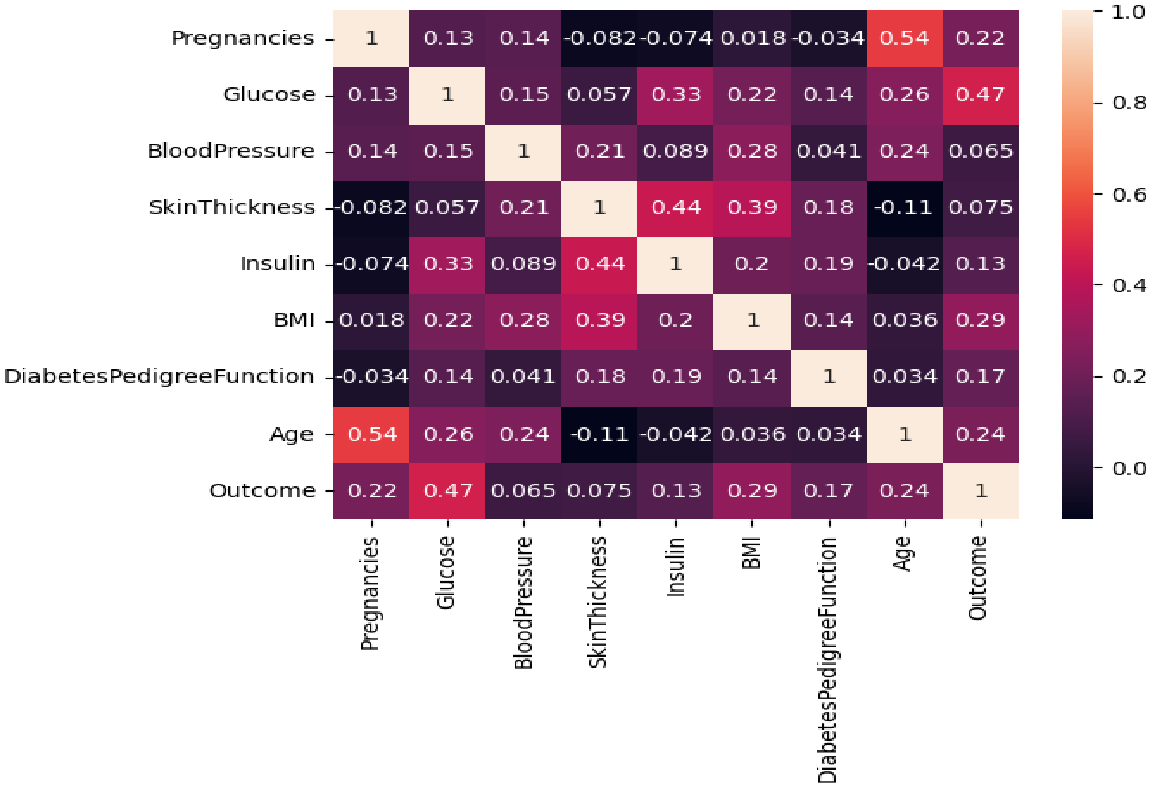
```
(768, 9)
```

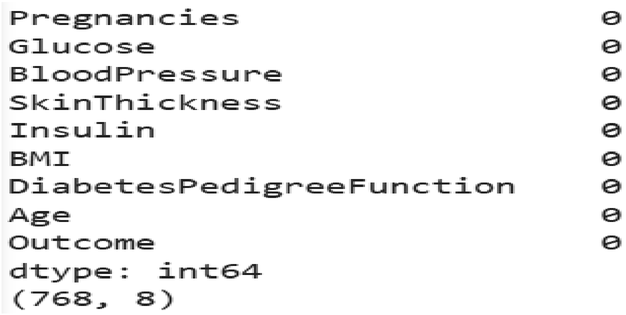


```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       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
```

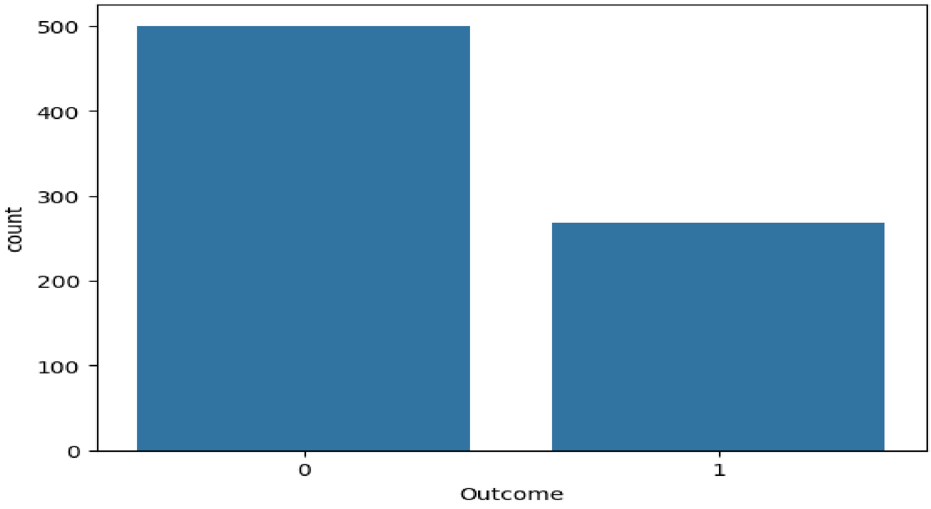


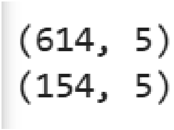
```
Outcome
0    500
1    268
Name: count, dtype: int64
```



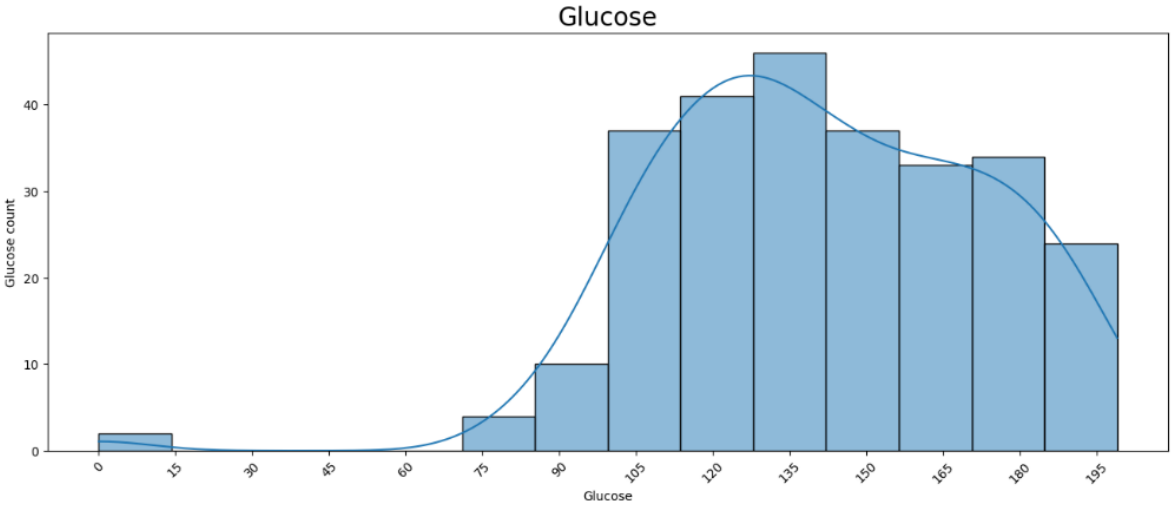


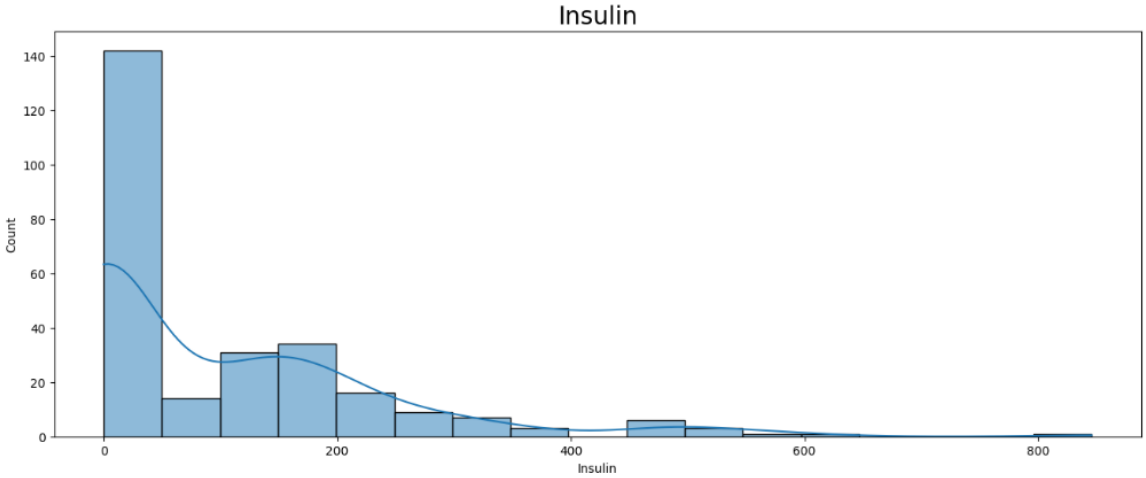
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
(768, 8)
```

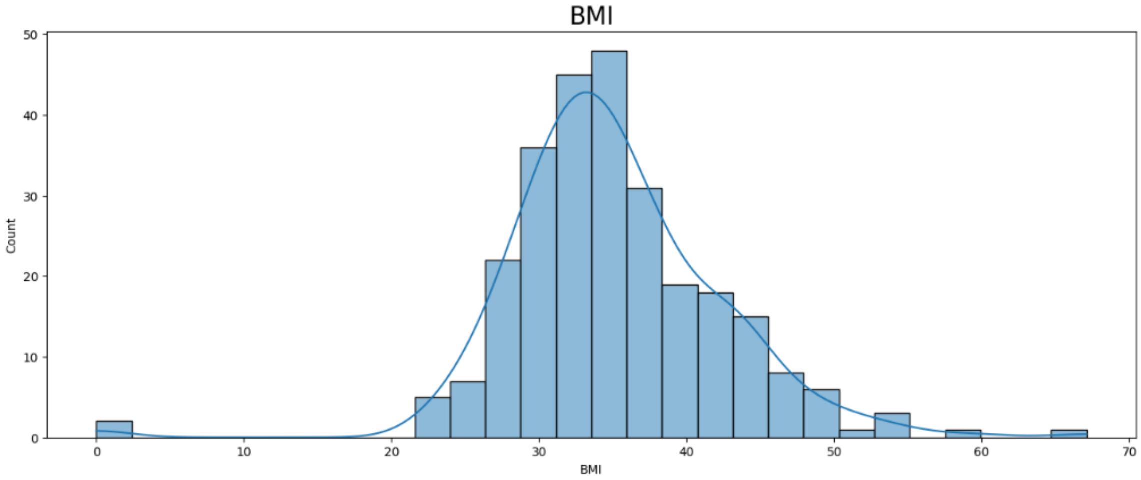


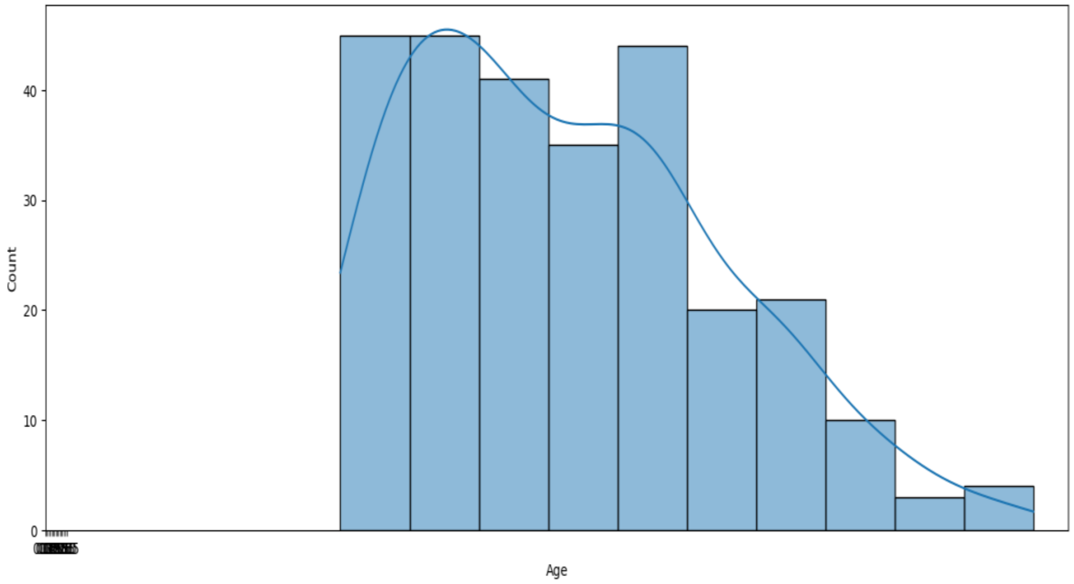


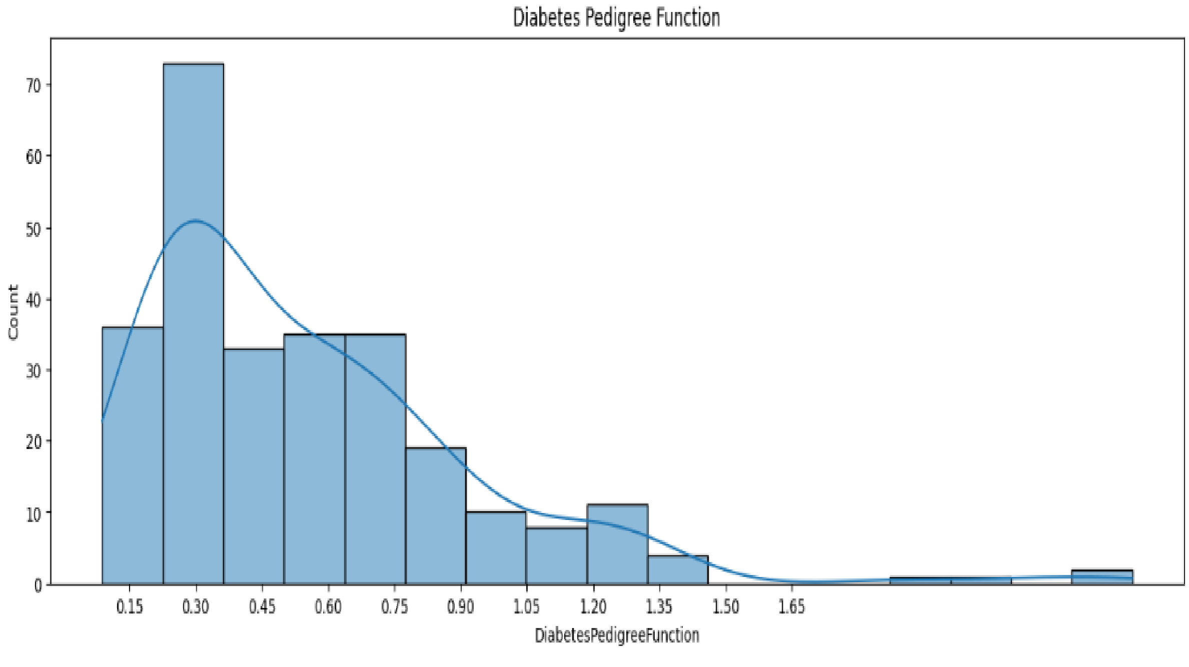
```
(614, 5)
(154, 5)
```


Glucose
Glucose count
0
10
20
30
40
0
15
30
45
60
75
90
105
120
135
150
165
180
195
Glucose


Insulin
Count
0
20
40
60
80
100
120
140
0
200
400
600
800
Insulin


BMI
Count
0
10
20
30
40
50
0
10
20
30
40
50
60
70
BMI

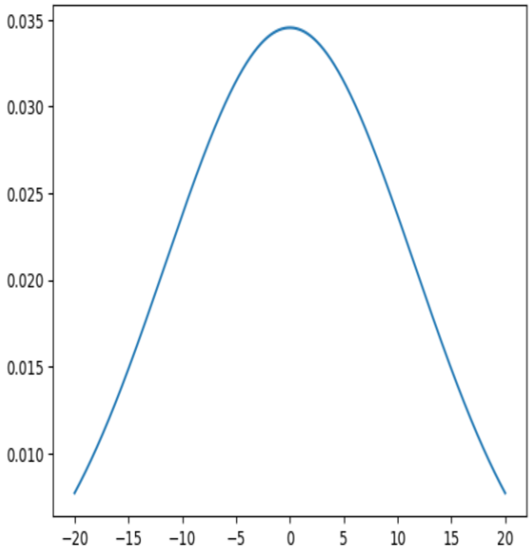


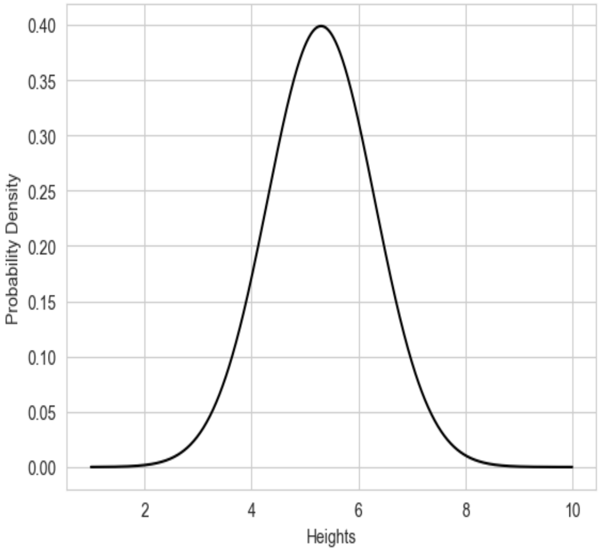

Diabetes Pedigree Function

**EX.NO:6(A)**

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from scipy.stats import norm
import statistics
x_axis = np.arange(-20, 20, 0.01)
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
data = np.arange(1, 10, 0.01)
pdf = norm.pdf(data, loc=5.3, scale=1)
sb.set_style('whitegrid')
sb.lineplot(x=data, y=pdf, color='black')
plt.xlabel('Heights')
plt.ylabel('Probability Density')
plt.show()
```

**OUTPUT:**

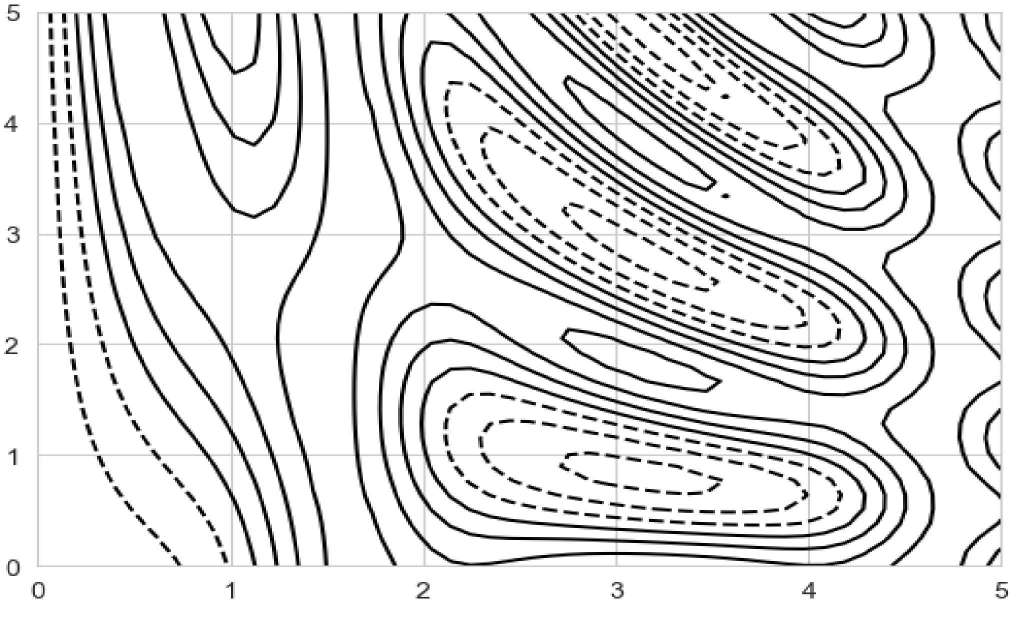


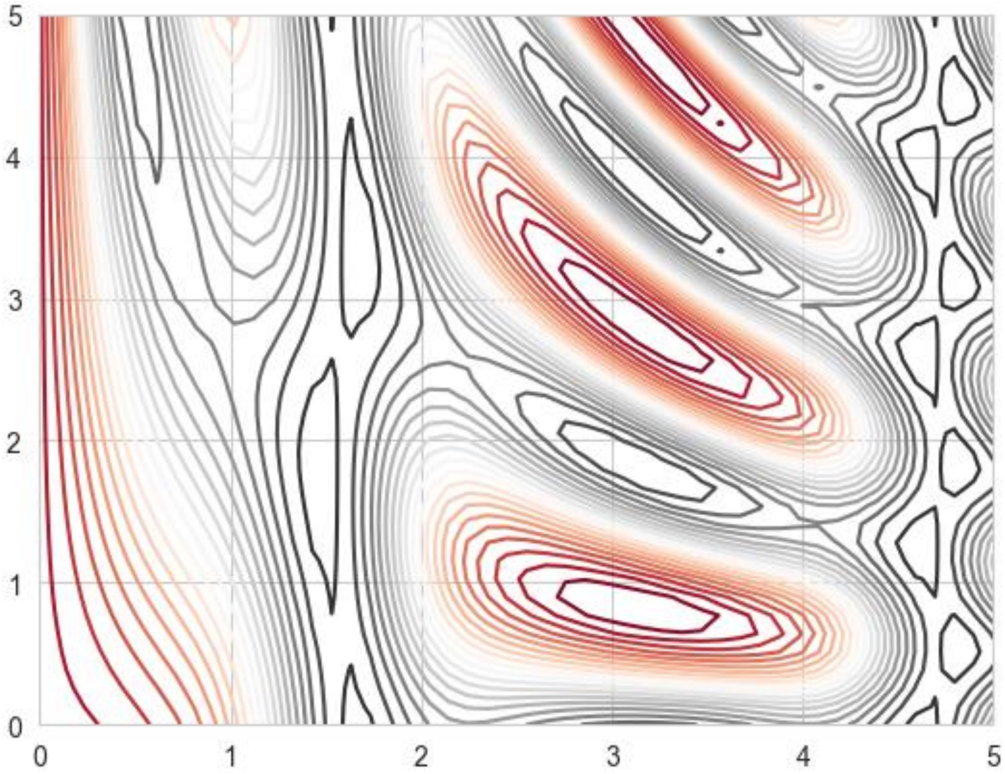


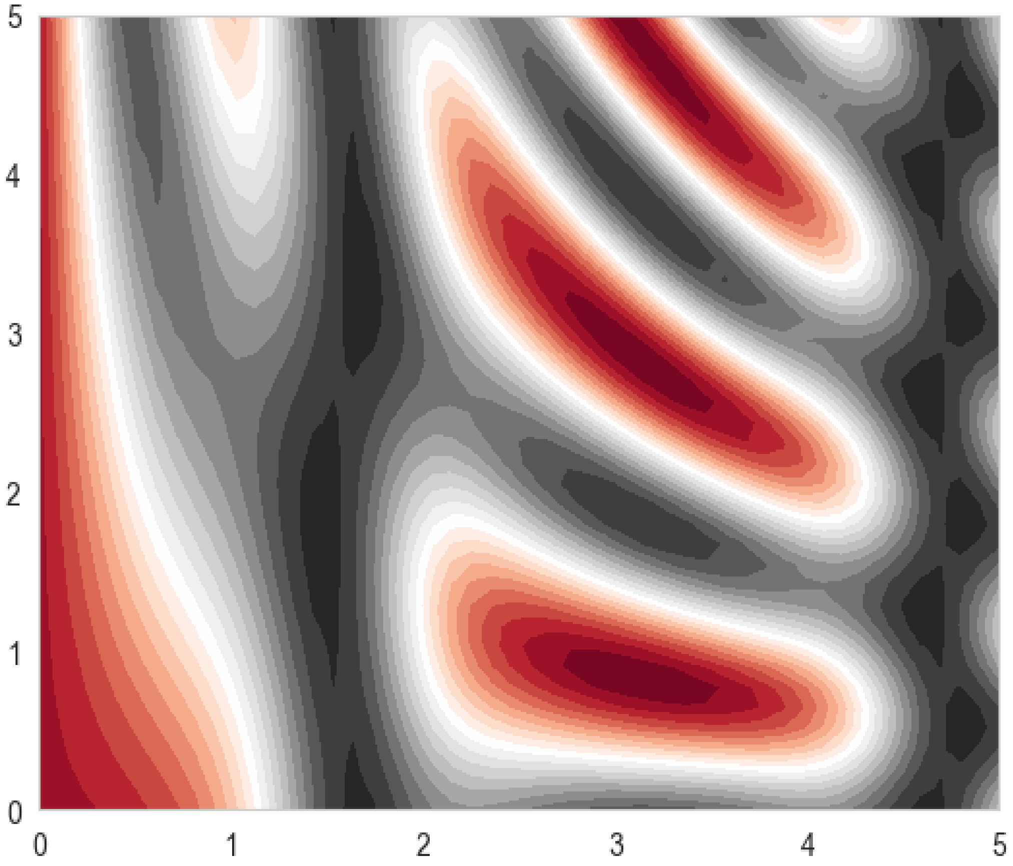
**EX.NO:6(B)**

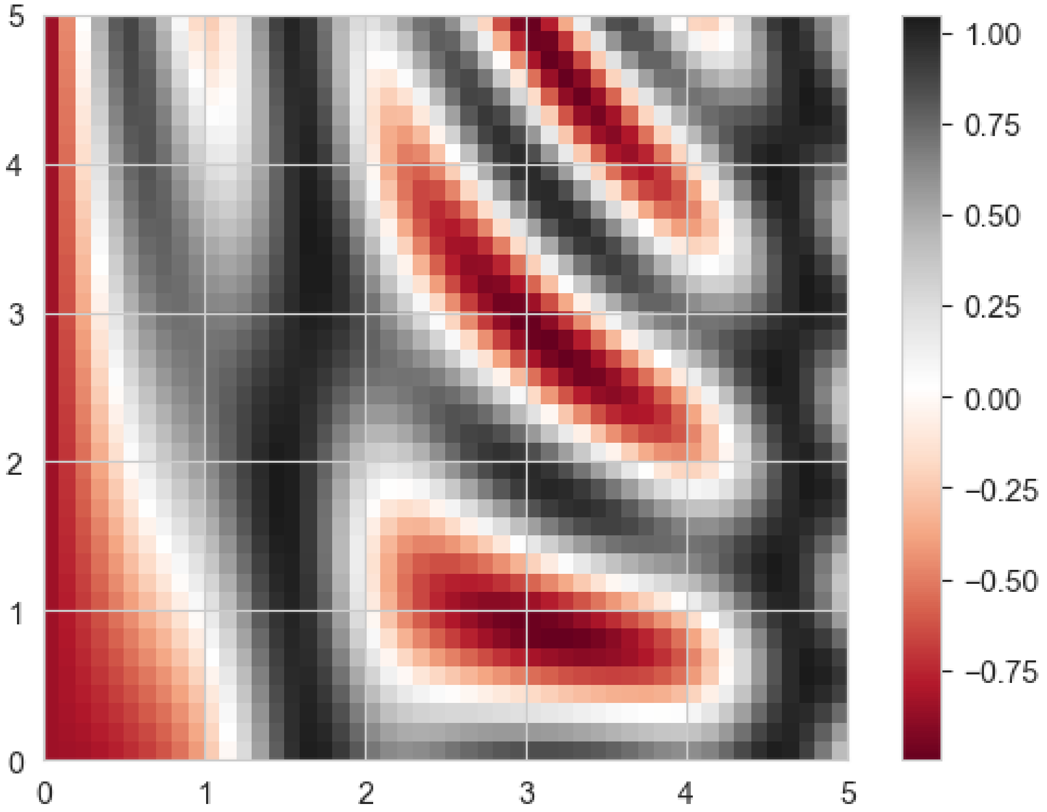
```
import matplotlib.pyplot as plt
import numpy as np
def f(x, y):
    return np.sin(x)**10 + np.cos(10 + y * x) * np.cos(x)
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
plt.contour(X, Y, Z, colors='black')
plt.contour(X, Y, Z, 20, cmap='RdGy')
plt.contourf(X, Y, Z, 20, cmap='RdGy')
plt.imshow(Z, extent=[0, 5, 0, 5], origin='lower', cmap='RdGy')
plt.colorbar()
plt.gca().set_aspect('equal')
contours = plt.contour(X, Y, Z, levels=3, colors='black')
plt.clabel(contours, inline=True, fontsize=8)
plt.imshow(Z, extent=[0, 5, 0, 5], origin='lower', cmap='RdGy', alpha=0.5)
plt.show()
```

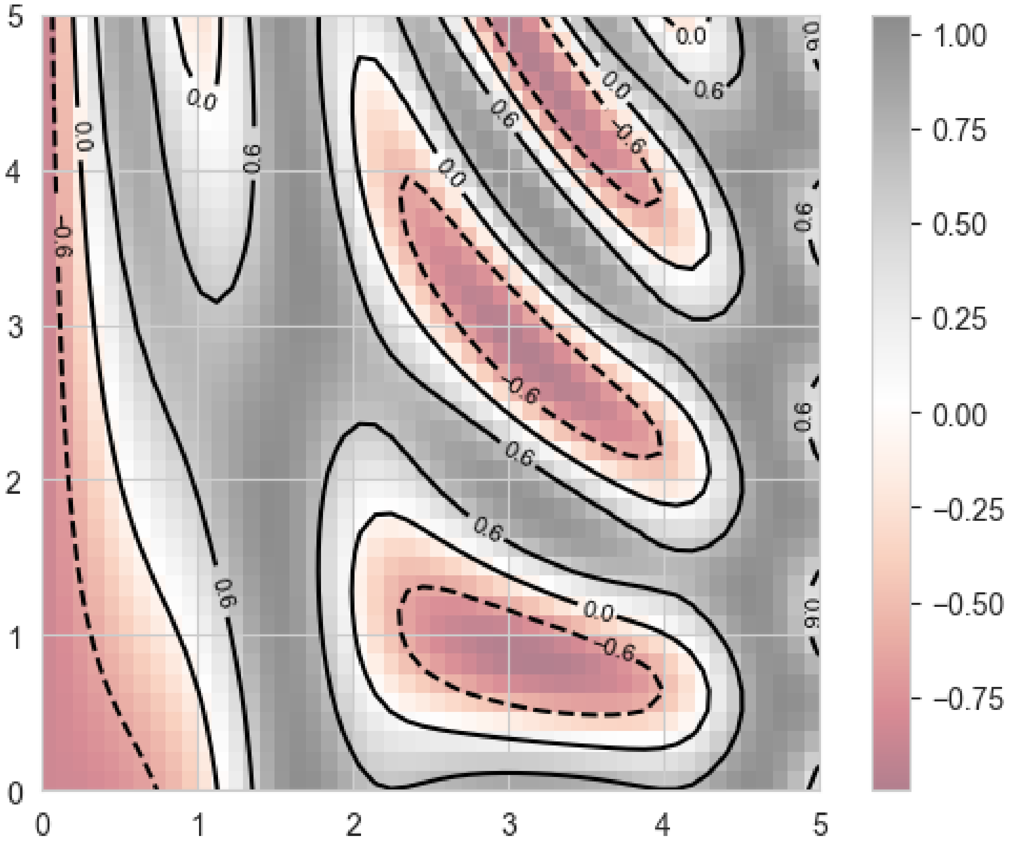
**OUTPUT:**




5
4
3
2
1
0
0
1
2
3
4
5


5
4
3
2
1
0
0
1
2
3
4
5

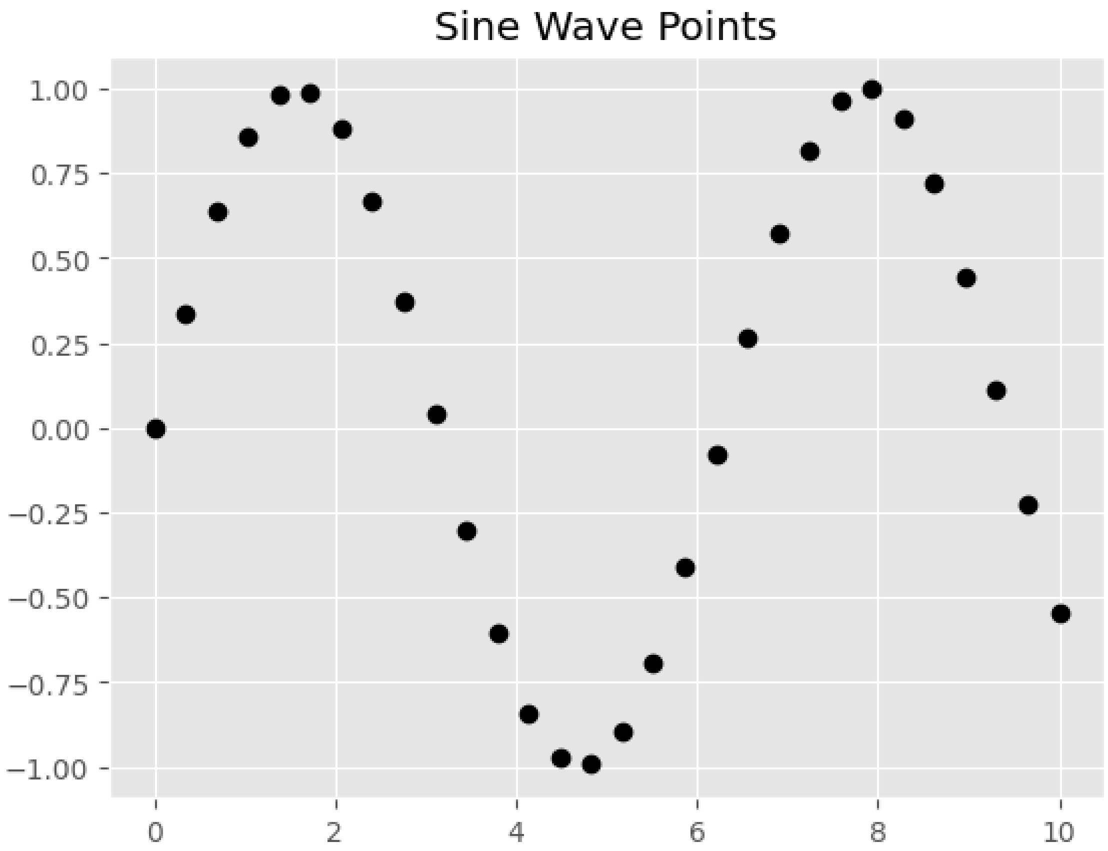

5
4
3
2
1
0
0
1
2
3
4
5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

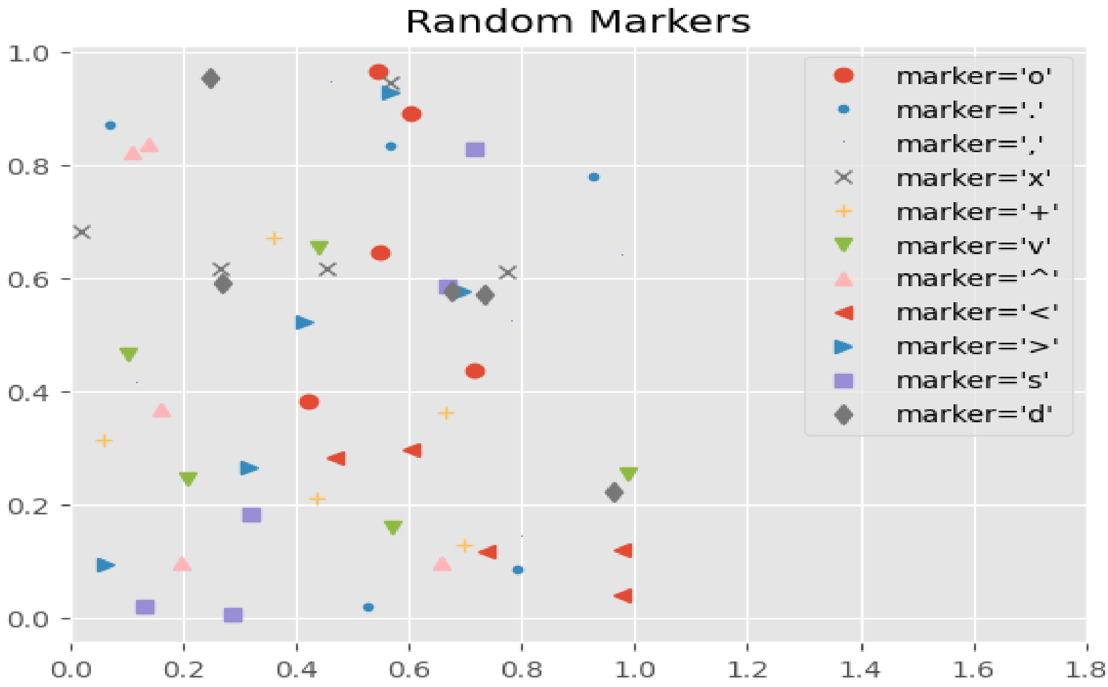

5
4
3
2
1
0
0
1
2
3
4
5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.0
0.6
-0.6
0.0
0.6
0.0
0.6
0.0
-0.6
0.0
0.6
-0.6
0.6
0.6
0.0
-0.6
0.0
0.6
0.6
0.6
0.6
0.6

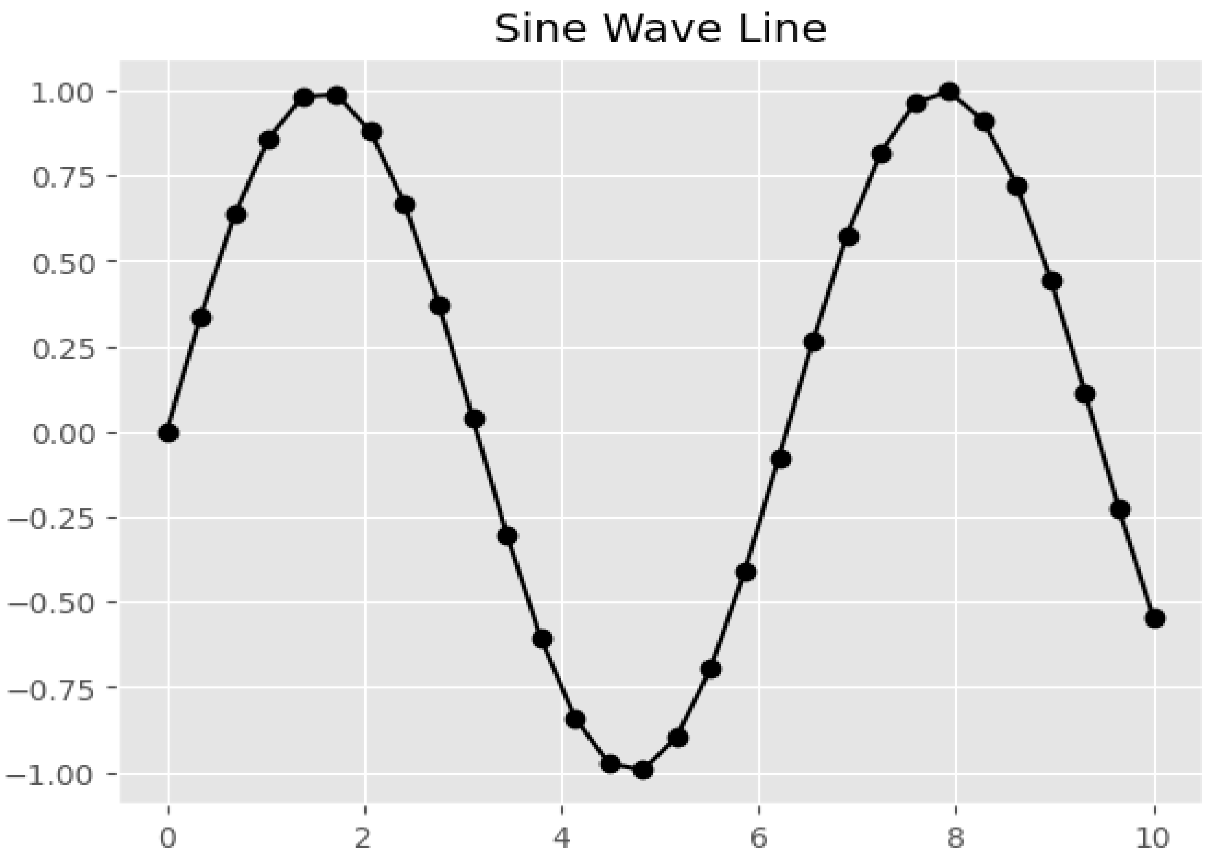
**EX.NO:6(C)**

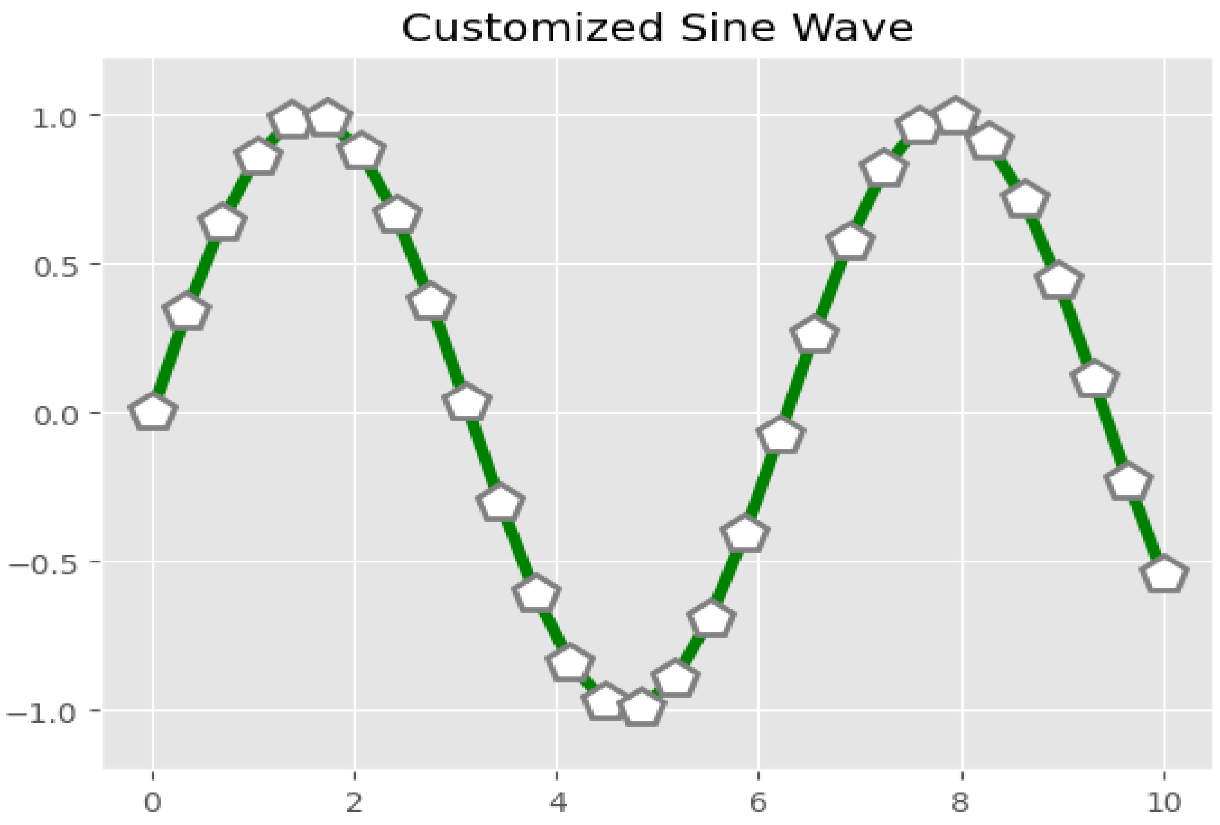
```
import matplotlib.pyplot as plt

import numpy as np

from sklearn.datasets import load_iris

plt.style.use('ggplot')

x = np.linspace(0, 10, 30)

y = np.sin(x)

plt.plot(x, y, 'o', color='black')

plt.show()

markers = ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']

for m in markers: plt.plot(np.random.rand(5), np.random.rand(5), m)

plt.xlim(0, 1.8)

plt.show()

plt.plot(x, y, '-ok')

plt.show()

plt.plot(x, y, '-p', color='green', markersize=15, markerfacecolor='white')

plt.ylim(-1.2, 1.2)

plt.show()

plt.scatter(x, y)

plt.show()

x, y, s = np.random.randn(100), np.random.randn(100), 1000 * np.random.rand(100)

plt.scatter(x, y, c=np.random.rand(100), s=s, alpha=0.3, cmap='viridis')

plt.colorbar()

plt.show()

iris = load_iris()

plt.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target, s=100 * iris.data[:, 3], cmap='viridis',
alpha=0.2)

plt.show()
```

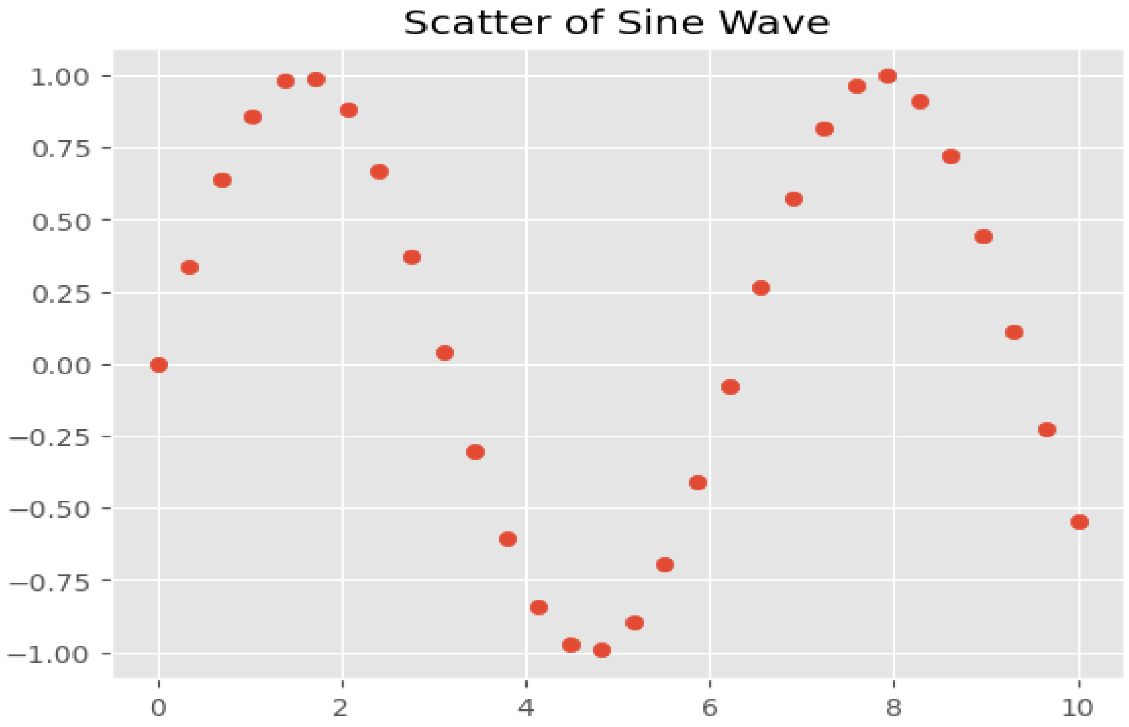
**OUTPUT:**

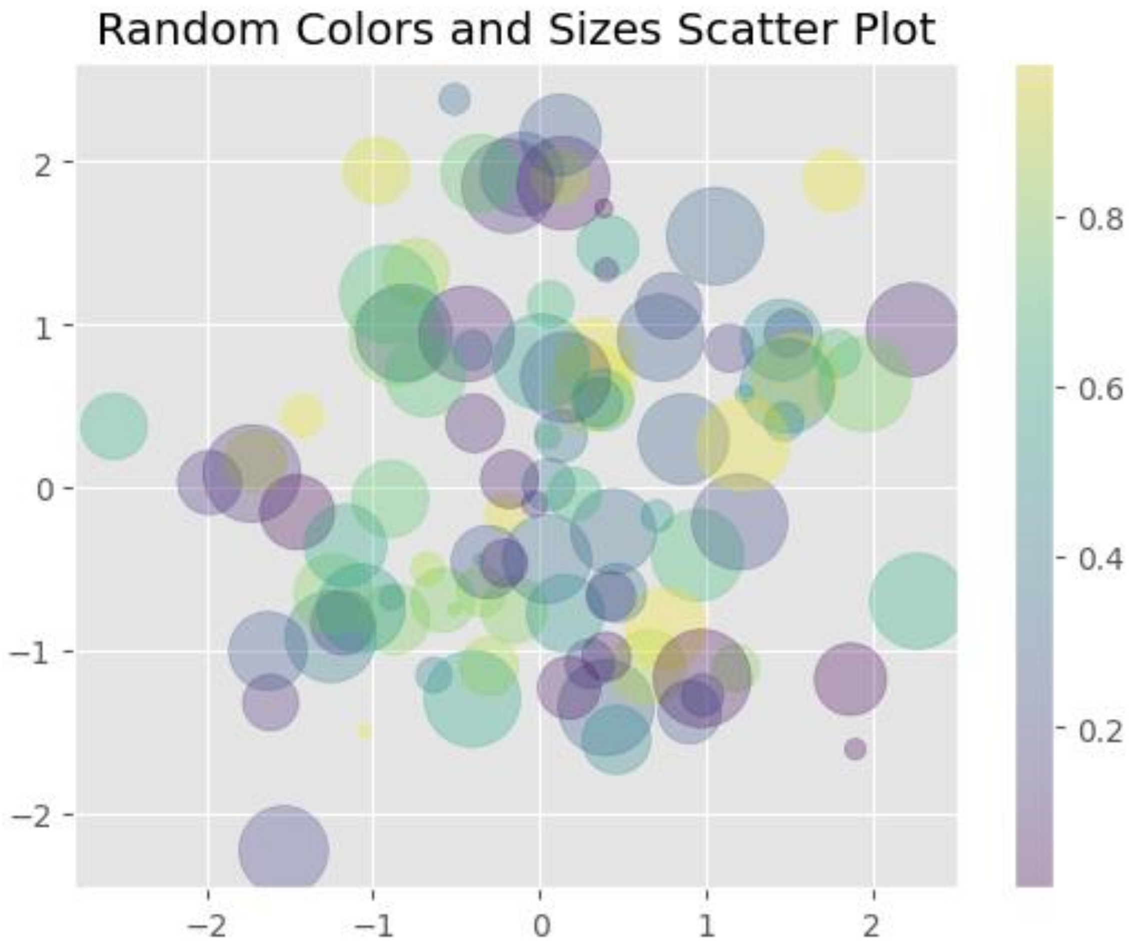

Sine Wave Points
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10

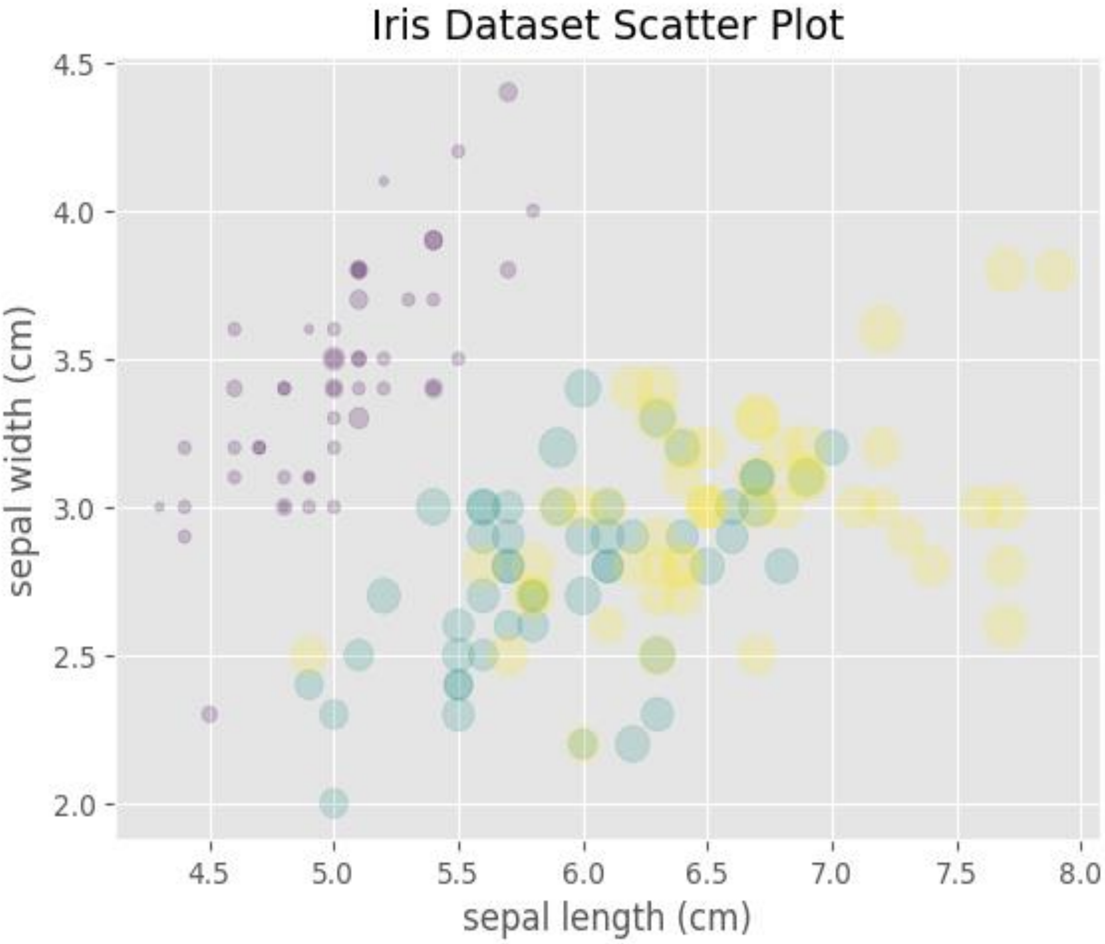

Random Markers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
marker='o'
marker='.'
marker=','
marker='x'
marker='+'
marker='v'
marker='^'
marker='<'
marker='>'
marker='s'
marker='d'


Sine Wave Line
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10


Customized Sine Wave
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10


Scatter of Sine Wave
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10

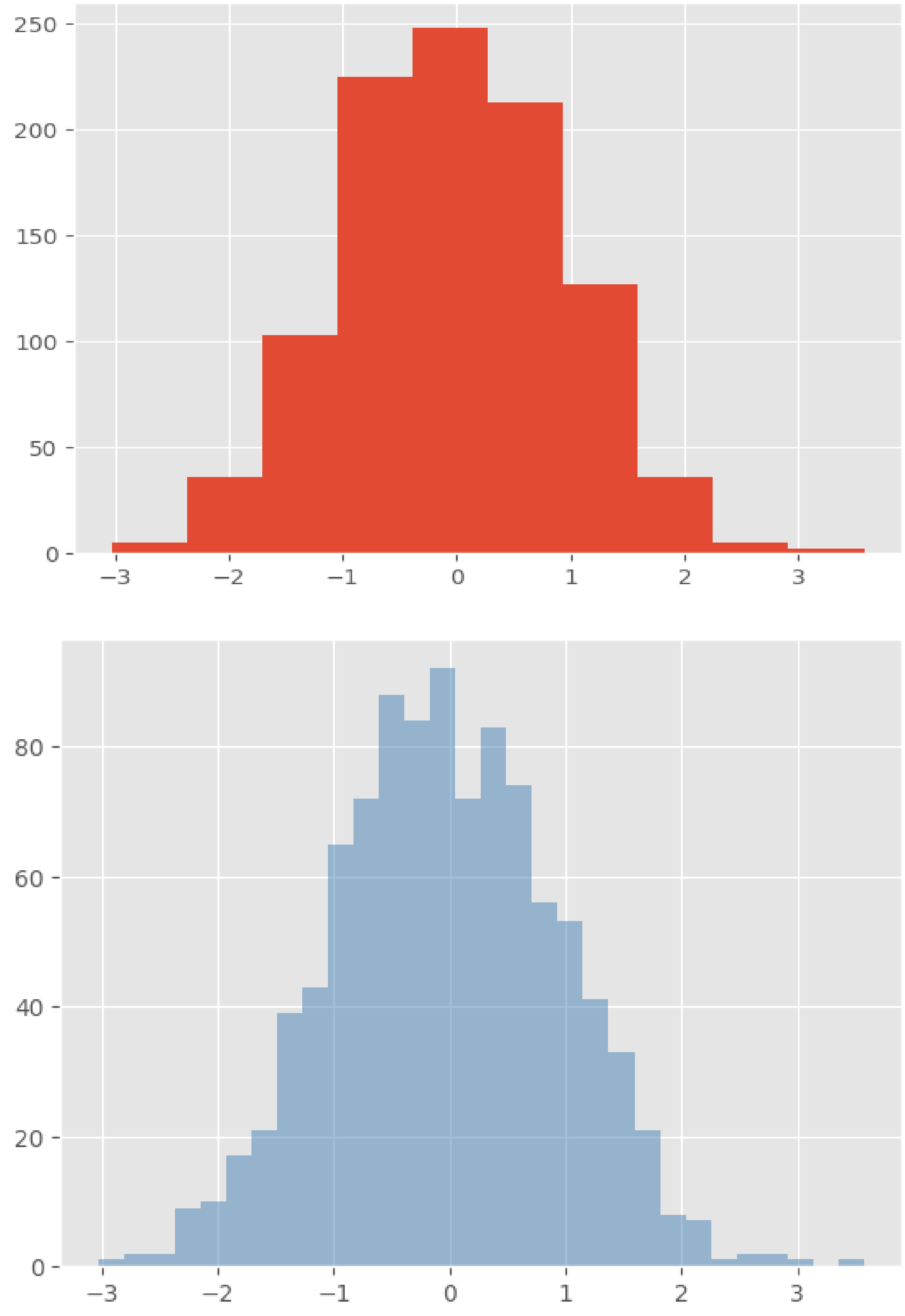

Random Colors and Sizes Scatter Plot
2
1
0
−1
−2
−2
−1
0
1
2
0.8
0.6
0.4
0.2

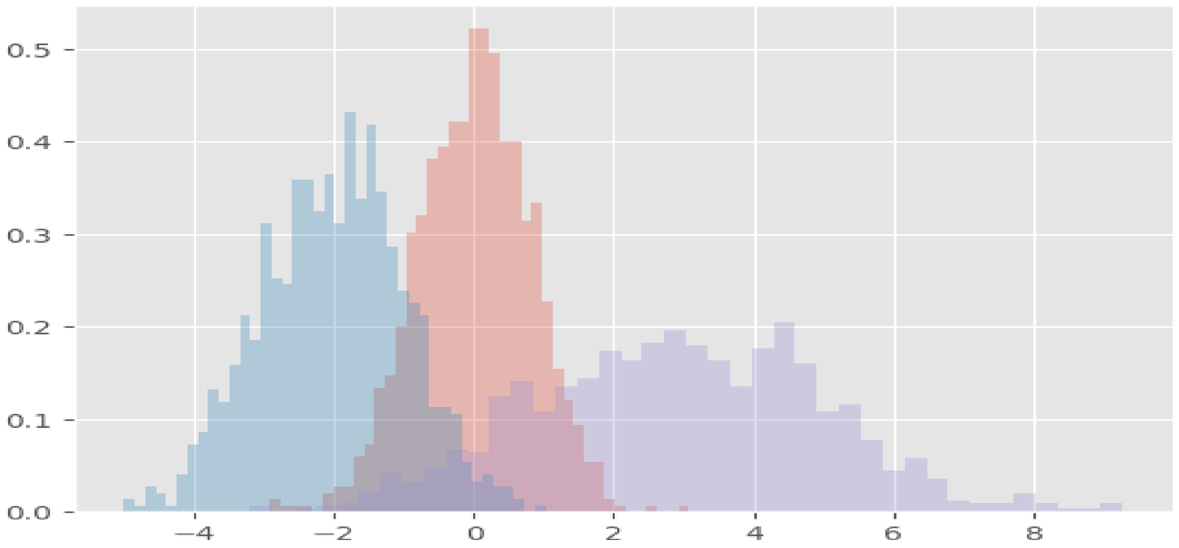

Iris Dataset Scatter Plot
sepal width (cm)
sepal length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

**EX. NO:6(D)**

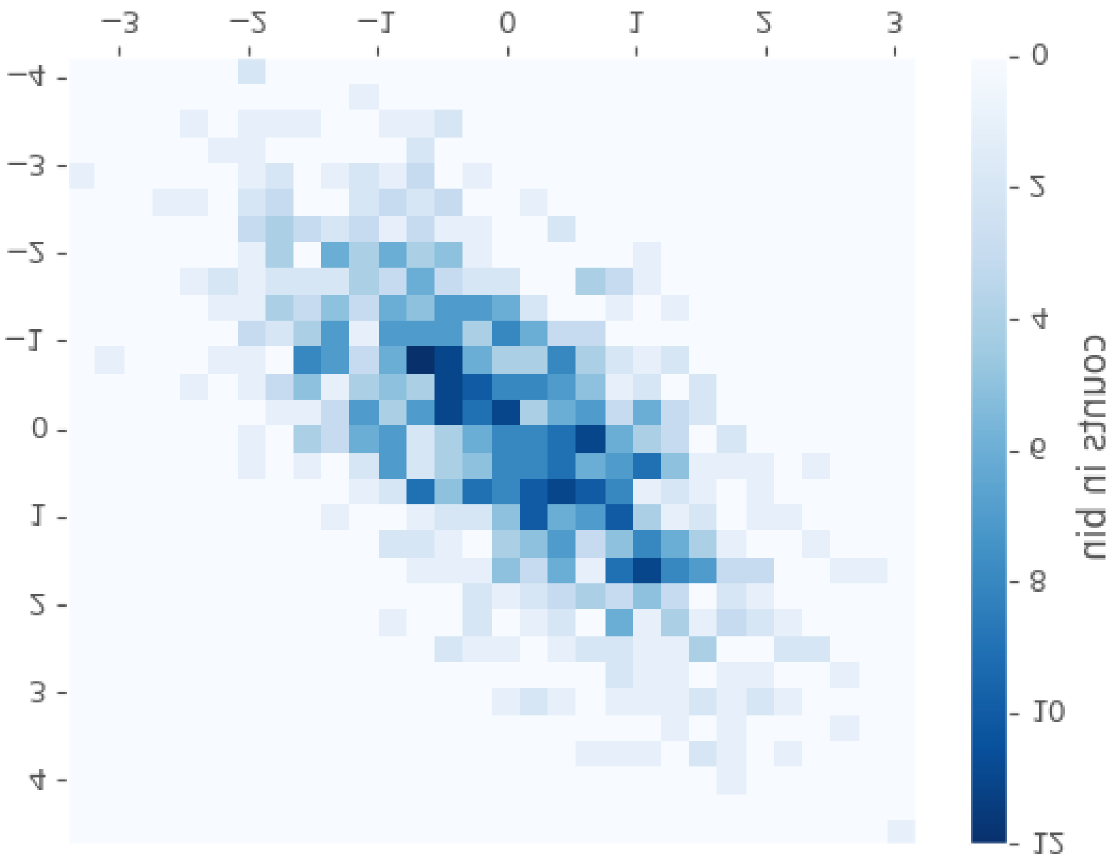
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

plt.style.use('ggplot')
data = np.random.randn(1000)

plt.hist(data, bins=30, alpha=0.5, histtype='stepfilled', color='steelblue', edgecolor='none')
plt.show()

x1, x2, x3 = np.random.normal(0, 0.8, 1000), np.random.normal(-2, 1, 1000),
np.random.normal(3, 2, 1000)
plt.hist(x1, bins=40, histtype='stepfilled', alpha=0.3, density=True)
plt.hist(x2, bins=40, histtype='stepfilled', alpha=0.3, density=True)
plt.hist(x3, bins=40, histtype='stepfilled', alpha=0.3, density=True)
plt.show()

counts, _ = np.histogram(data, bins=5)
print(counts)

mean, cov = [0, 0], [[1, 1], [1, 2]]
x, y = np.random.multivariate_normal(mean, cov, 1000).T

plt.hist2d(x, y, bins=30, cmap='Blues')
plt.colorbar(label='counts in bin')
plt.show()

plt.hexbin(x, y, gridsize=30, cmap='Blues')
plt.colorbar(label='count in bin')
plt.show()

data = np.vstack([x, y])
kde = gaussian_kde(data)
xgrid, ygrid = np.meshgrid(np.linspace(-3.5, 3.5, 40), np.linspace(-6, 6, 40))
Z = kde.evaluate(np.vstack([xgrid.ravel(), ygrid.ravel()]))

plt.imshow(Z.reshape(xgrid.shape), origin='lower', aspect="auto", extent=[-3.5, 3.5, -6, 6],
cmap='Blues')
plt.colorbar(label="density")
plt.show()
```

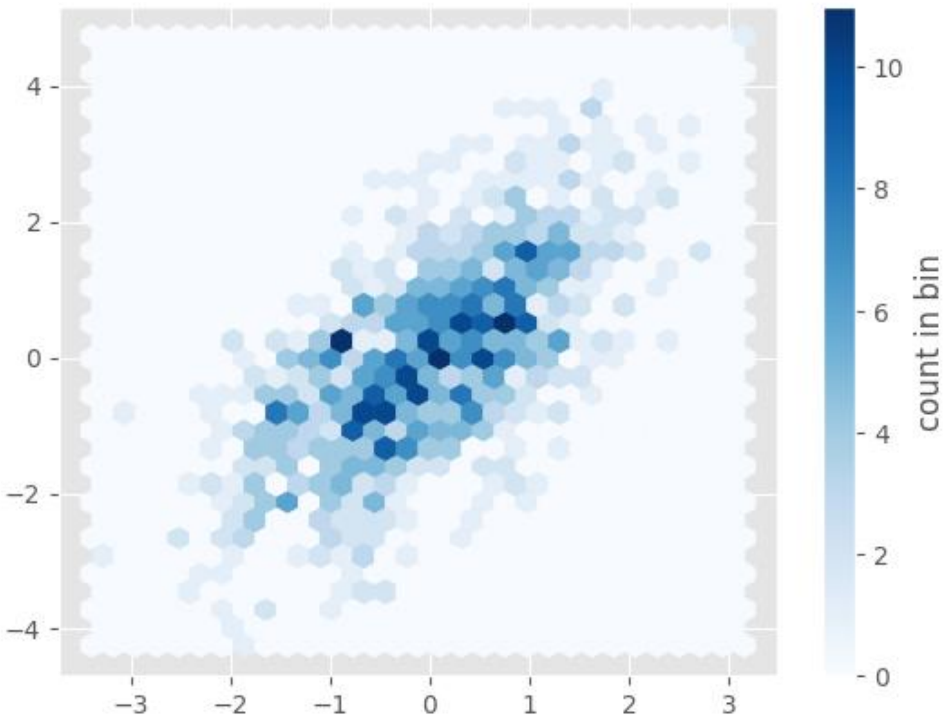
**OUTPUT:**

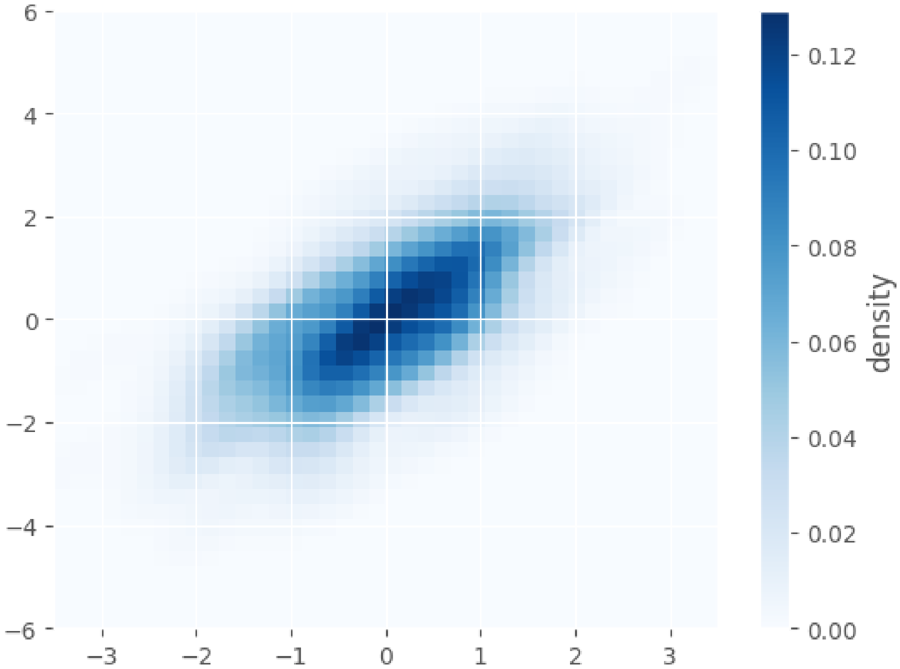




```
[ 41 328 461 163   7]
```



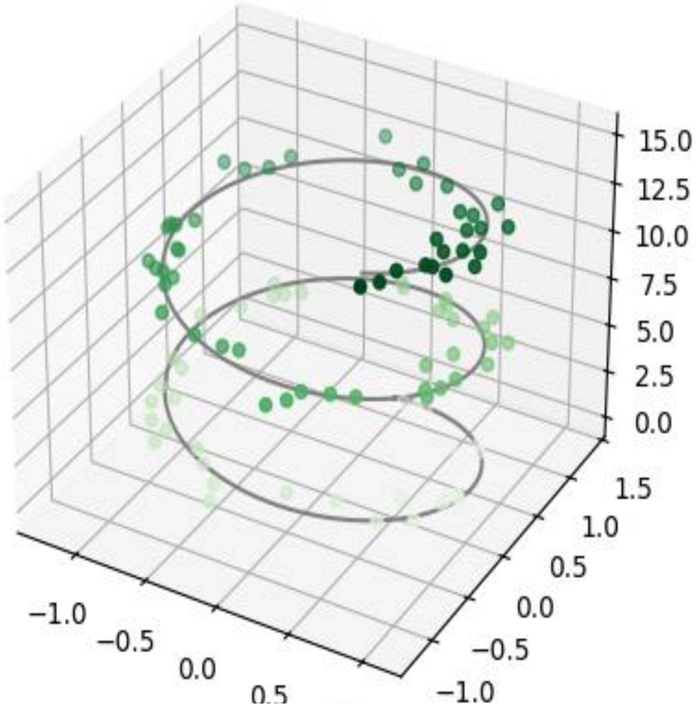

4
2
0
−2
−4
−3
−2
−1
0
1
2
3
10
8
6
4
2
0
count in bin


6
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3
0.12
0.10
0.08
0.06
0.04
0.02
0.00
density

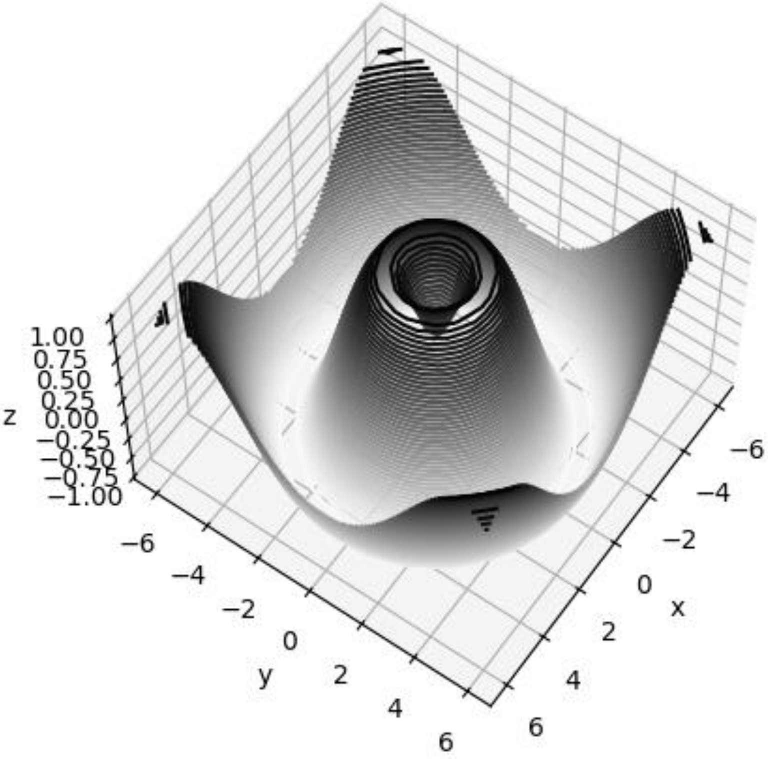
**EX.NO:6(E)**

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(12, 10))

ax1 = fig.add_subplot(231, projection='3d')
zline = np.linspace(0, 15, 1000)
ax1.plot3D(np.sin(zline), np.cos(zline), zline, 'gray')

ax2 = fig.add_subplot(232, projection='3d')
zdata = 15 * np.random.random(100)
ax2.scatter(np.sin(zdata) + 0.1 * np.random.randn(100), np.cos(zdata) + 0.1 *
np.random.randn(100), zdata, c=zdata, cmap='Greens')

def f(x, y):
    return np.sin(np.sqrt(x**2 + y**2))

x, y = np.linspace(-6, 6, 30), np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)

ax3 = fig.add_subplot(233, projection='3d')
ax3.contour3D(X, Y, Z, 50, cmap='binary')
ax3.view_init(60, 35)

ax4 = fig.add_subplot(234, projection='3d')
ax4.plot_wireframe(X, Y, Z, color='black')

ax5 = fig.add_subplot(235, projection='3d')
ax5.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis', edgecolor='none')

theta = 2 * np.pi * np.random.random(1000)
r = 6 * np.random.random(1000)
x = r * np.sin(theta)
y = r * np.cos(theta)
ax6 = fig.add_subplot(236, projection='3d')
ax6.scatter(x, y, f(x, y), c=f(x, y), cmap='viridis', linewidth=0.5)

plt.tight_layout()
plt.show()
```

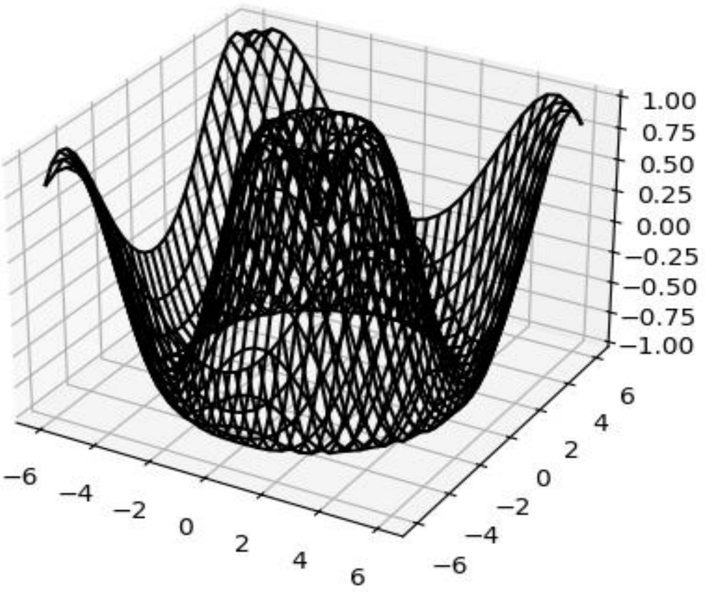
**OUTPUT:**


15.0
12.5
10.0
7.5
5.0
2.5
0.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5

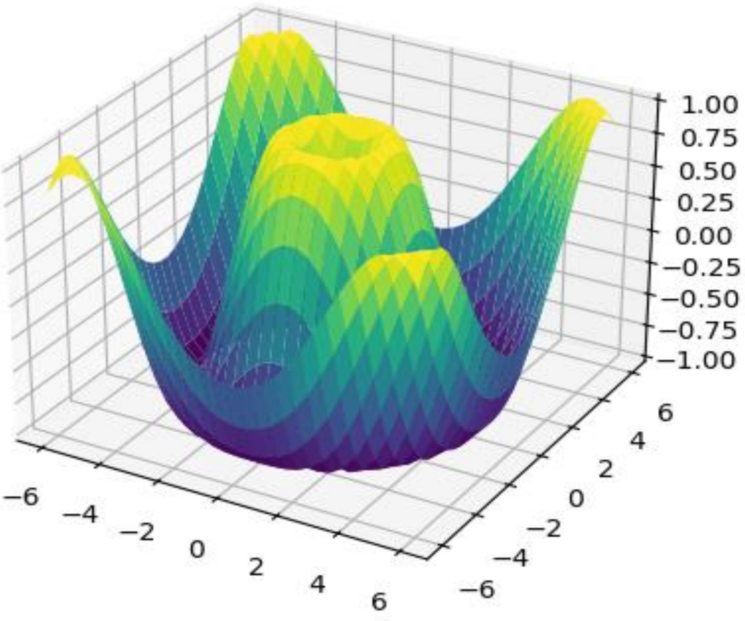
:

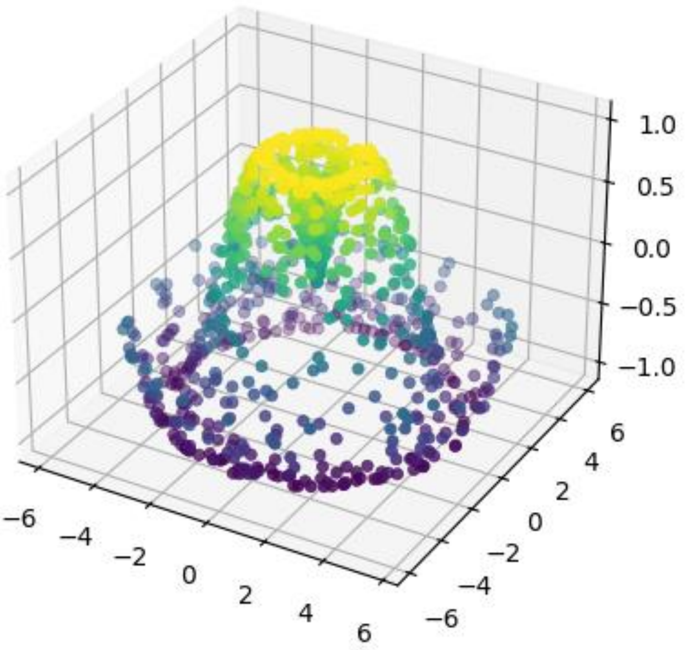

1.00
0.75
0.50
0.25
z
0.00
−0.25
−0.50
−0.75
−1.00
−6
−4
−2
0
y
2
4
6
−6
−4
−2
0
x
2
4
6

wireframe

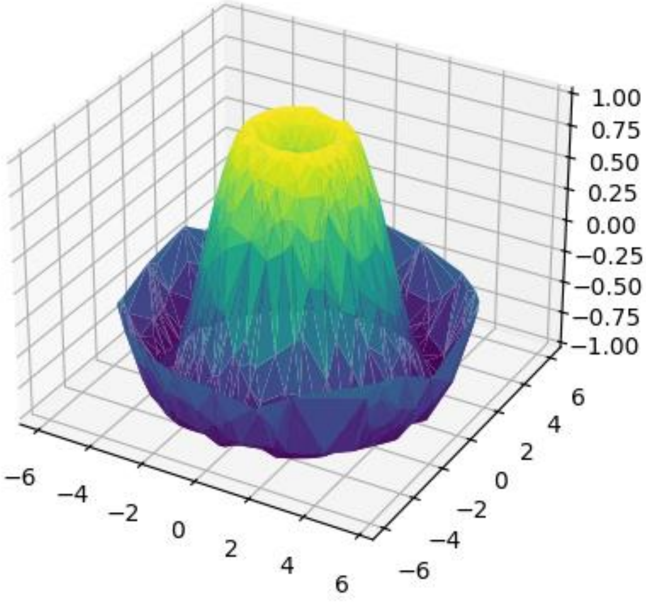


surface





1.0
0.5
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6
6
4
2
0
−2
−4
−6

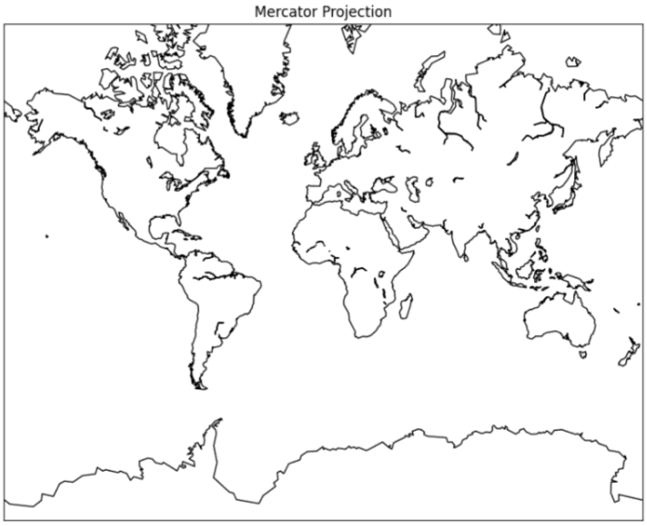


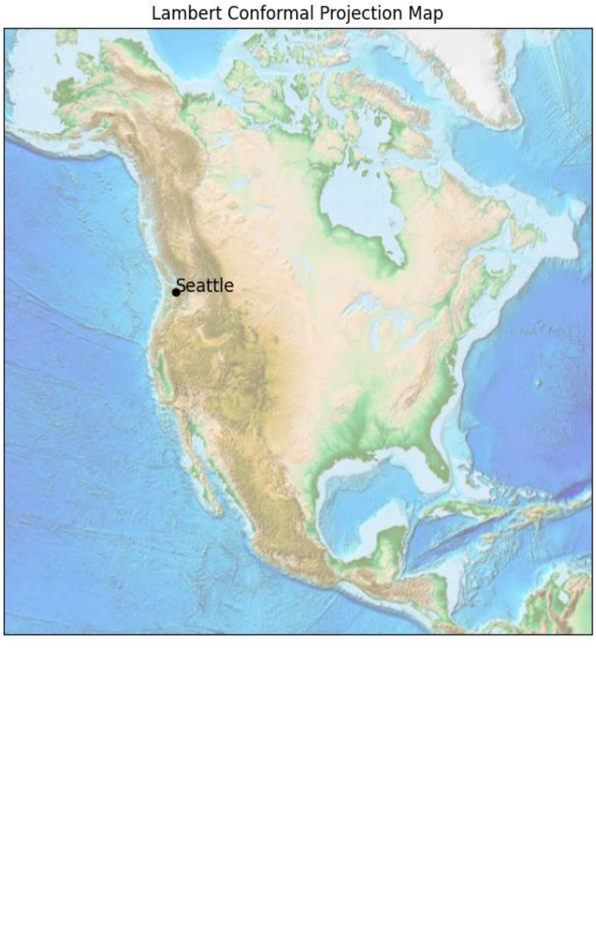
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−6
−4
−2
0
2
4
6
6
4
2
0
−2
−4
−6

**EX.NO:7**

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
import warnings

warnings.filterwarnings("ignore")

fig1 = plt.figure(figsize=(12, 8))
m1 = Basemap(projection='merc', llcrnrlat=-80, urcrnrlat=80,
            llcrnrlon=-180, urcrnrlon=180, resolution='c')
m1.drawcoastlines()
plt.title("Mercator Projection")
plt.show()

fig2 = plt.figure(figsize=(8, 8))
m2 = Basemap(projection='lcc', resolution='i',
            width=8E6, height=8E6, lat_0=45, lon_0=-100)
m2.etopo(scale=0.5, alpha=0.5)

x, y = m2(-122.3, 47.6)
plt.plot(x, y, 'ok', markersize=5)
plt.text(x, y, 'Seattle', fontsize=12)

plt.title('Lambert Conformal Projection Map')
plt.show()
```

**OUTPUT:**


Mercator Projection


Lambert Conformal Projection Map
Seattle