PROGRAM:

(i) Creation of different types of Numpy arrays and displaying basic information

```
# Importing numpy
import numpy as np

# Defining 1D array
mylDArray = np.array([l , 8, 27, 64]

print(myl DArray)

# Defining and printing 2D array
my2DArray = np .array([[l , 2, 3, 4], [2, 4, 9, 16], [4, 8, 18, 32]])
print(my2DArray)

#Defining and printing 3D array
my3Darray = np.array([[[ 1, 2 , 3 , 4],[ 5 , 6 , 7 ,8]], [[ 1, 2, 3, 4],[ 9, 10, 11,
 12]]])
print(my3Darray)
# Print out memory address
print(my2DArray.data)
# Print the shape of array
print(my2DArray .shape)

# Print out the data type of the array
print(my2DArray .dtype)

# Print the stride of the array.
print(my2DArray .strides)
```

(ii) Creation of an array using built-in NumPy functions

```
# Array of ones
ones = np.ones((3 ,4))
print( ones)

# Array of zeros
zeros = np.zeros((2 ,3,4),dtype=np.int16)
print(zeros)

# Array with random values
np.random.random((2,2))
```

```
# Empty array
emptyArray = np.empty((3 ,2))
print( emptyArray)

# Full array
fullArray = np.full((2 ,2),7)
print(fullArray)
 # Array of evenly-spaced values
evenSpacedArray = np .arange( l0,25,5)
print( evenSpacedArray)

# Array of evenly-spaced values
evenSpacedArray2 = np.linspace(0 ,2,9)
print( evenSpacedArray2)
```

(iii) Performing file operations with NumPy arrays

```
import numpy as np

#initialize an array
arr = np.array([[[ll , 11, 9, 9], [11, 0, 2, O]], [[10, 14, 9, 14], [O, 1, 11, 11]]])

# open a binary file in write mode
file = open("arr", "wb")

# save array to the file
np .save(file, arr)

# close the file
file.close

# open the file in read binary mode
file = open("arr", "rb")

#read the file to numpy array
arr l = np.load(file)
#close the file
print( arr1)
```

OUTPUT:

(i) Creation of different types of Numpy arrays and displaying basic information

```
[ 1  8 27  64]

[[ 1  2  3  4]

 [ 2  4  9  16]

 [ 4  8  18 32]]

[[[ 1  2  3  4]

  [ 5  6  7  8]]

 [[ 1  2  3  4]

  [ 9  10 11  12]]]

<memory at Ox0000024 7AE2AOAOO >

(3, 4)

int32

(16, 4)
```

(ii) Creation of an array using built-in NumPy functions

```
[[l. 1. 1. l.]
 [l. 1. 1. l.]
 [1. 1. 1. 1.]]

[[[O 00 O]
  [O 0 0 O]
 [O 00 O]]
 [O 0 0 O J
  [O0 0 O J
  [O0 0 O J ] ]

[[O. O.]

 [O. O.]
 [O. O.]]

[[7 7]

 [7 7]]

[10 15 20]

[O.  0.25 0.5  0.75 1.   1.25 1.5  1.75 2.  ]
```

(iii) Performing ftle operations with NumPy arrays

```
[[[11 11 9 9]

  [11 0 2 O J ]


 [[10 14 9 14]

  [ O 1 11 11]]]
```

PROGRAM:

```
import numpy as np
a = np .arange(9 , dtype = np.float_).reshape(3 ,3)

print ('First array:')
print (a)
print ('\n')

print ('Second array:')
b = np .array([l0,10, 10])
print (b )
print ('\n')

print ('Add the two arrays:')
print (np.add(a ,b))
print ('\n')

print ('Subtract the two arrays:')
print (np.subtract(a ,b))
print ('\n')

print ('Multiply the two arrays:')
print (np.multiply(a ,b))
print ('\n')

print ('Divide the two arrays:')
print (np.divide(a,b))
```

OUTPUT:

```
First array :
[[ 0.  1. 2.]
 [ 3. 4.  5.]
 [ 6. 7.  8.]]
Second array:
[10  10  10]

Add the two arrays:
[[  10.  11.  12.]
 [  13.  14.  15.]
 [  16.  17.  18.]]
Subtract the two arrays:
[[-10. -9. -8.]
 [ -7. -6. -5 .]
 [ -4. -3.  -2.]]

Multiply the two arrays:
[[ 0.  10. 20.]
 [ 30.  40.  50.]
 [ 60.  70 .  80.]]
Divide the two arrays:
[[ 0.  0.1 0.2]
 [ 0.3  0.4  0.5]
 [ 0.6  0.7  0.8]]
```

PROGRAM:

(i) CREATION OF A DATAFRAME FROM A SERIES

```
import numpy as np
import pandas as pd
print("Pandas Version :", pd ._version_)
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows' , 500)
series = pd.Series([2 , 3, 7, 11, 13, 17, 19, 23])
print( series)
series_df = pd .DataFrame( {
'A': range(l , 5),
'B': pd.Timestamp('20190526') ,
'C': pd .Series(5 , index=list(range(4)) , dtype='float64') ,
'D': np.array([3] * 4, dtype='int64'),
'E': pd.Categorical(["Depression" , "Social Anxiety", "Bipolar Disorder", "Eating
Disorder"]),
'F': 'Mental health',
'G': 'is challenging'
 } )
print( series_df)
```

(ii) CREATION OF A DATAFRAME FROM DICTIONARY

```
import numpy as np
import pandas as pd

dict_df = [{'A': 'Apple', 'B': 'Ball'},{'A': 'Aeroplane', 'B':'Bat', 'C': 'Cat'}]
diet_df = pd .DataFrame( diet_df)
print( diet_df)
```

(iii) CREATION OF A DATAFRAME FROM N-DIMENSIONAL ARRAYS

```
import numpy as np
import pandas as pd

sdf = {'County':['Ostfold', 'Hordaland' , 'Oslo', 'Hedmark', 'Oppland', 'Buskemd'] ,
'ISO-Code' :[l,2,3,4,5,6],
'Area': [4180.69, 4917.94 , 454.07 , 27397.76 , 25192 .10, 14910.94],
'Administrative centre' : ["Sarpsborg", "Oslo", "City of Oslo", "Hamar",
"Lillehammer", "Drammen"]}
sdf = pd.DataFrame(sdf)
print(sdf)
```

(iv) LOADING A DATASET FROM AN EXTERNAL SOURCE INTO A PANDASDATAFRAME

```
    import numpy as np
    import pandas as pd
    columns=['age' , 'workclass', 'fnlwgt', 'education', 'education_num' ,
'marital_status' , 'occupation', 'relationship', 'ethnicity' , 'gender', 'capital_gain' ,
'capital_loss','hours_per_week','country_of_origin','income']
df=pd.read_csv('http ://archive.ics.uci.edu/ml/machine-leaming-
databases/adult/adult.data' ,names=columns)
df.head(lO)
```

OUTPUT:

(i) Creation of a dataframe from a series

```
Pandas Version : 1.3.4
0    2
1    3
2    7
3   11
4   13
5   17
6   19
7   23
dtype: int64
A       B   C D             E            F            G
0  1 2019-05-26   5.0  3      Depression   Mental health   is challenging
1  2 2019-05-26   5.0  3   Social Anxiety   Mental health   is challenging
2  3 2019-05-26   5.0  3  Bipolar Disorder  Mental health   is challenging
3 4 2019-05-26    5.0  3   Eating Disorder  Mental health   is challenging
```

(ii) Creation of a dataframe from a dictionary

```
            A        B        C

0   Apple       Ball    NaN
1   Aeroplane  Bat   Cat
```

(iii) Creation of a dataframe from n-dimensional array

```
County  ISO-Code      Area          Administrative  centre

0            Ostfold        1   4180.69           Sarpsborg

1          Hordaland       2   4917.94               Oslo

2              Oslo          3    454.07          City of Oslo

3           Hedmark       4  27397.76              Hamar

4           Oppland      5    25192.10         Lillehamme

5           Buskerud     6    14910.94           Drammen
```

PROGRAM:

DATA INPUT AND OUTPUT

This notebook is the reference code for getting input and output, pandas can read a variety of file types using its pd.read_ methods. Let's take a look at the most common data types:

```
import numpy
as np import
pandas as pd
```

CSV

CSV INPUT:

```
df =
pd.read_csv('exam
ple') df
```

	a	**b**	**c**	**d**
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

CSV OUTPUT:

```
df.to_csv('example',index=False)
```

EXCEL

Pandas can read and write excel files, keep in mind, this only imports data. Not formulas or images, having images or macros may cause this read_excel method to crash.

EXCEL INPUT :

```
pd.read_excel('Excel_Sample.xlsx'   ,sheetname='Sheet  1')
```

```
     a    b    c    d
0    0    1    2    3
1    4    5    6    7
2    8    9   10   11
3   12   13   14   15
```

EXCEL OUTPUT :

```
df.to_excel('Excel_Sample.xlsx'  ,sheet_name='Sheet   1')
```

HTML

You may need to install htmllib5, lxml, and Beautifu1Soup4. In your terminal/command prompt run:

```
pip install lxml
pip install
html5lib== 1.1
pip install
Beautifu1Soup
4
```

Then restart Jupyter Notebook. (or use conda

install) Pandas can read table tabs off of html.

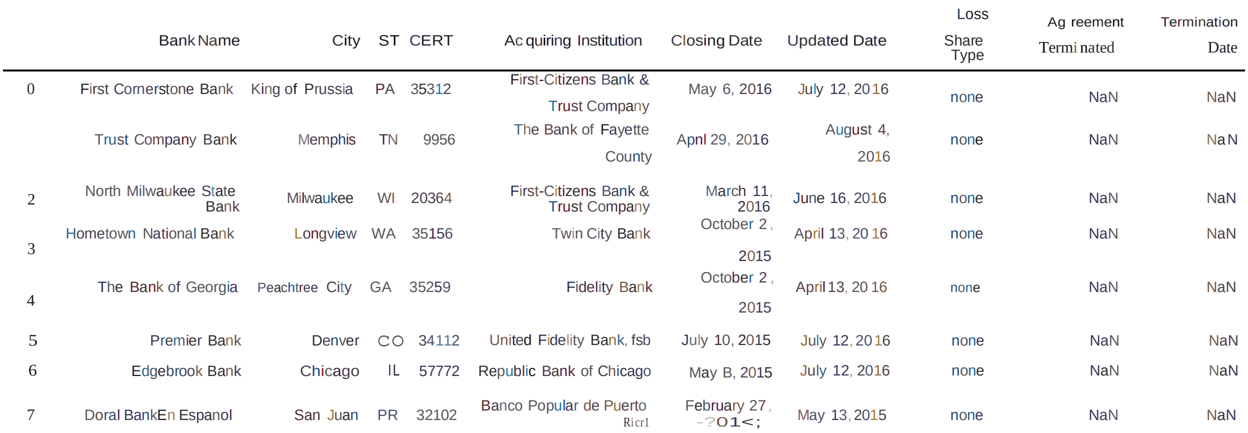
For example:

HTML INPUT

Pandas read_html function will read tables off of a webpage and return a list of DataFrame objects:

```
url = https ://www.fdic.gov/resources/resolutions/bank-

failures/failed-bank-list df = pd.read_html(url)

df[O]

match = "Metcalf Bank"

df_list = pd.read_html(url,

match=match) df_list[O]
```

HTML OUTPUT:



	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date	Loss Share Type	Agreement Terminated	Termination Date
0	First Cornerstone Bank	King of Prussia	PA	35312	First-Citizens Bank & Trust Company	May 6, 2016	July 12, 2016	none	NaN	NaN
	Trust Company Bank	Memphis	TN	9956	The Bank of Fayette County	Apnl 29, 2016	August 4, 2016	none	NaN	NaN
2	North Milwaukee State Bank	Milwaukee	WI	20364	First-Citizens Bank & Trust Company	March 11, 2016	June 16, 2016	none	NaN	NaN
3	Hometown National Bank	Longview	WA	35156	Twin City Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
4	The Bank of Georgia	Peachtree City	GA	35259	Fidelity Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
5	Premier Bank	Denver	CO	34112	United Fidelity Bank, fsb	July 10, 2015	July 12, 2016	none	NaN	NaN
6	Edgebrook Bank	Chicago	IL	57772	Republic Bank of Chicago	May B, 2015	July 12, 2016	none	NaN	NaN
7	Doral BankEn Espanol	San Juan	PR	32102	Banco Popular de Puerto Ricr1	February 27, -?O1<;	May 13, 2015	none	NaN	NaN

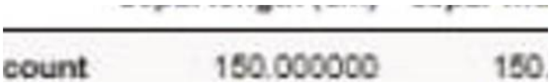
PROGRAM:

```
import pandas as pd

from pandas import DataFrame

from skleam.datasets import load_iris

# skleam.datasetsincludes common example datasets

# A function to load in the iris dataset

iris_obj = load_iris()

# Dataset preview

iris_obj.data

iris = DataFrame(iris_obj.data, columns=iris_obj.feature_names,index=pd.Index([i for i in
range(iris_obj.data.shape[O]) ])).join(DataFrame(iris_obj .target, columns=pd.Index([" species"]),
index=pd.Index([i for i in range(iris_obj.target.shape[O])])))

iris # prints iris data
```

Commands

```
iris_obj.feature_names

iris.count()

iris.mean()

iris.median()

iris.var()

 iris.std()

iris.max()

iris.min()

iris.describe()
```

OUTPUT:



```
count        150.000000        150.
```

m) petal width (cm) species

PROGRAM:

```
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set_style('darkgrid')
     %matplotlib inline
     from matplotlib.ticker  import FormatStrFormatter
     import warnings
     warnings .filterwarnings('ignore')

     df = pd.read_csv('C:/Users/kirub/Documents/Learning/Untitled    Folder/diabetes.csv')
     df.head()
     df.shape
     df.dtypes
     df['Outcome']=df['Outcome']  .astype('bool')
     df.dtypes['Outcome']
     df.info()
     df.describe().T

     # Frequency# finding the unique count
     dfl  =  df['Outcome'].value_counts()

     # displaying dfl
     print(dfl)
     #mean
     df.mean()
     #median
     df.median()
        #mode
      df.mode()
   #Variance
df.var()
   #standard  deviation  df.std()
   #
   #kurtosis
     df.kurtosis( axis=O,skipna=True)
     df['Outcome'].kurtosis(axis=O,skipna=  True)
   #skewness
     # skewness along the index axis
     df.skew(axis =  0,  skipna  =  True)

# skip  the  na  values
# find skewness in each row
df.skew(axis  = 1, skipna  = True)
```

```
#Pregnancy variable
preg_proportion = np.array( df['Pregnancies']. value_counts())
preg_month = np.array( df['Pregnancies']. value_counts().index)
preg_proportion_perc =
np.array(np.round(preg_proportion/ sum(preg_proportion ),3)*100,dtype=int)

preg =
pd.DataFrame( {'month':preg_month,'count_of_preg_prop' :preg_proportion,'percentage_pro
portion':preg_proportion_perc})
preg.set_index(['month'] ,inplace=True)
preg.head( 10)

sns.countplot(data=df['Outcome'])

sns.distplot(df['Pregnancies'])

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

OUTPUT:

```
''
               coun              n          st     lll1n   25%    59     75X      x
     agt     1025 0                    9072290     29.0     .0   SG.O   61 0    77 0
     Hlt     1  5.0      0.695610    0-160373      0.0    0.0     10     1D     1O
      cp     1or D       9J2   9     1.029   1      0 0    0 0    1 0    2 0    JO
 UISU!ps     1   5.0   131.61 707   17.5  67  6   9  .0    0.0  130.0   140.    00 0
    chol     1oro     2..! 000000   515925  0     1260    211 0  24 00  275 0  5    0
      bs     1     .O     .1 l9  68     .3565 7    0.0    0.0     0.              1 0
 restecg     1025.0    0 529756     0.527878      0 0    0 0    1 0    1 0    2 0
 lhlilch     1or o    1-1'31 -11 6  23 00572       7  0  132 0   1570            o"
                                                                       166 0      0
    txng     1025.0    0.336585    0 472m         0.0    0.0    0.0    1 0    10
 oldpea      1or o       0 1512     1   1ro-3     0 0    0 0    0 8    1 8     ?
   slope         .0    .385366     o. 1nss        0.0    1.0    1.0      0     0
      ca          0     0     1-1   1 0   7  s    0 0    0 0    0 0    1 0     0
    tnal     1DT D    2.323902    0 620660        0 0    2.0    2 0    30    JO
```

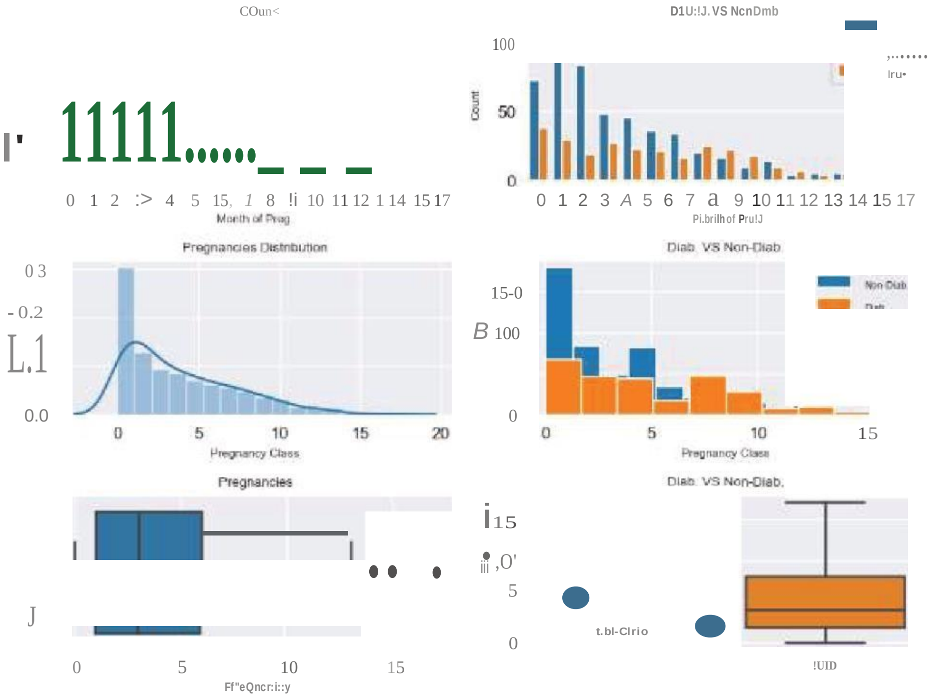
PROGRAM:

BIVARIATE ANALYSIS GENERAL PROGRAM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
%matplotlib inline
from matplotlib.ticker import FormatStrFormatter
import warnings
warnings .filterwarnings('ignore')
df = pd.read_csv('C:/Users/diabetes.csv')
df.head()
df.shape
df.dtypes
df['Outcome']=df['Outcome'] .astype('bool')
fig,axes = plt.subplots(nrows=3,ncols=2,dpi= l20,figsize = (8,6))
plotOO=sns .countplot('Pregnancies' ,data=df,ax=axes [0][0],color='green')
axes[O][0].set_title('Count',fontdict= {'fontsize':8})
axes[O] [0].set_xlabel('Month of Preg.',fontdict= {'fontsize' :7})
axes[O] [0].set_ylabel('Count',fontdict={ 'fontsize':7})
   plt.tight_layout()
plotO 1=sns.countplot('Pregnancies' ,data=df,hue='Outcome' ,ax=axes [0][1])
axes[O][l].set_title('Diab.  VS  Non-Diab.',fontdict={  'fontsize':8})
axes[O][ 1].set_xlabel('Month of Preg.',fontdict= {'fontsize' :7})
axes[O] [l].set_ylabel('Count',fontdict={  'fontsize':7})
plotO l .axes.legend(loc= 1)
plt.setp( axes[O] [1].get_legend() .get_texts(),  fontsize='6')
plt.setp( axes[O] [1].get_legend().get_title(), fontsize='6')
plt.tight_layout()
```

```
plot 10 =  sns.distplot( df['Pregnancies '],ax=axes[ 1][0])
axes[ 1][0].set_title('Pregnancies Distribution' ,fontdict= {'fontsize': 8})
axes[l]  [0].set_xlabel('Pregnancy     Class',fontdict={   'fontsize':7})
axes[l] [0].set_ylabel('Freq/Dist',fontdict={   'fontsize':7})
plt.tight_layout()

plot 11 = df[df['Outcome']==False] ['Pregnancies'] .plot.hist(ax=axes[ l][l],label='Non•
Diab.')
plot l 1_2=df[df['Outcome']==True] ['Pregnancies'] .plot.hist(ax=axes[ l][l],label='Diab. ')
axes[l][l] .set_title('Diab.  VS Non-Diab.',fontdict={ 'fontsize':8})
axes[l] [l].set_xlabel('Pregnancy Class',fontdict={ 'fontsize':7})
axes[l] [l].set_ylabel('Freq/Dist',fontdict={  'fontsize':7})
plot 11.axes.legend(loc= 1)
plt.setp(axes[l][  l].get_legend().get_texts(),  fontsize='6')  #  for  legend  text
plt.setp(axes[l][  l].get_legend().get_title(),  fontsize='6')  #  for  legend  title
plt.tight_layout()

plot20 = sns.boxplot(df['Pregnancies'] ,ax=axes [2][0],orient='v')
axes[2][0]  .set_title('Pregnancies'  ,fontdict={   'fontsize':8})
axes[2] [0].set_xlabel('Pregnancy' ,fontdict={ 'fontsize':7})
axes[2][0].set_ylabel('Five Point Summary',fontdict={  'fontsize':7})
plt.tight_layout()

plot2 l = sns.boxplot(x='Outcome' ,y='Pregnancies' ,data=df,ax=axes [2][1])
axes[2][1].set_title('Diab.  VS  Non-Diab.',fontdict={  'fontsize':8})
axes[2] [l].set_xlabel('Pregnancy' ,fontdict={ 'fontsize':7})
axes[2][1].set_ylabel('Five Point Summary',fontdict={ 'fontsize' :7})
plt.xticks( ticks=[O, 1],labels=['Non-Diab. ','Diab.'],fontsize=7)
plt.tight_layout()
plt.show()
```

OUTPUT:

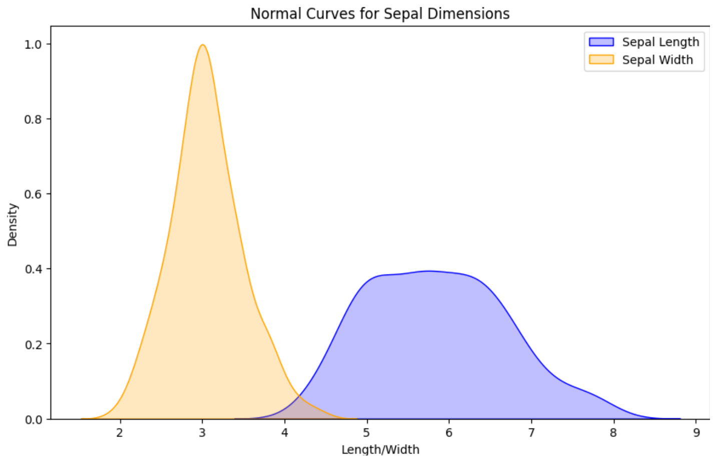


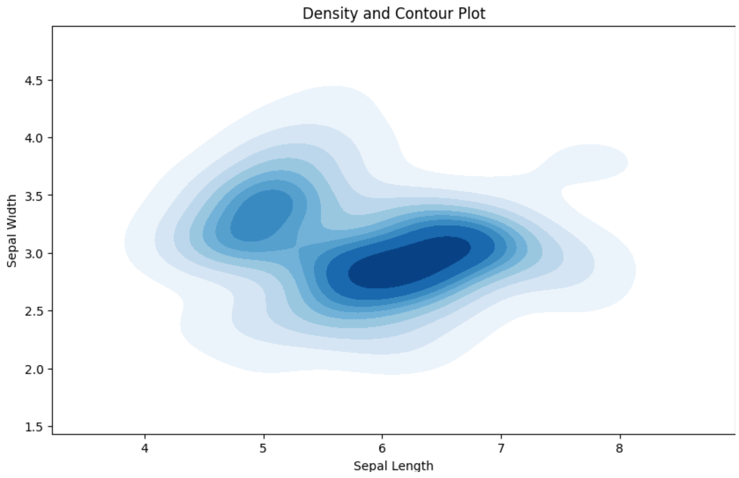
Count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17
Month of Preg
Diab. VS Non-Diab
100
50
0
Count
Non-Diab
Diab
Pregnancies Distribution
0.3
0.2
0.1
0.0
0 5 10 15 20
Pregnancy Class
150
100
0
0 5 10 15
Pregnancy Class
Pregnancies
0 5 10 15
Pregnancy
Diab. VS Non-Diab.
15
10
5
0
Non-Diab
Diab

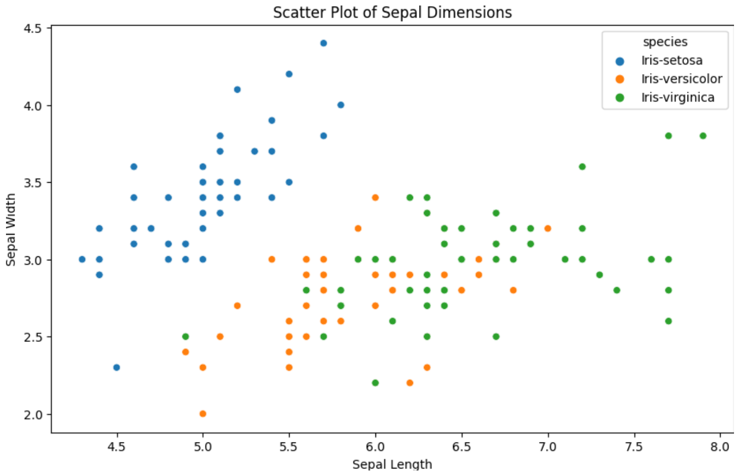
PROGRAM

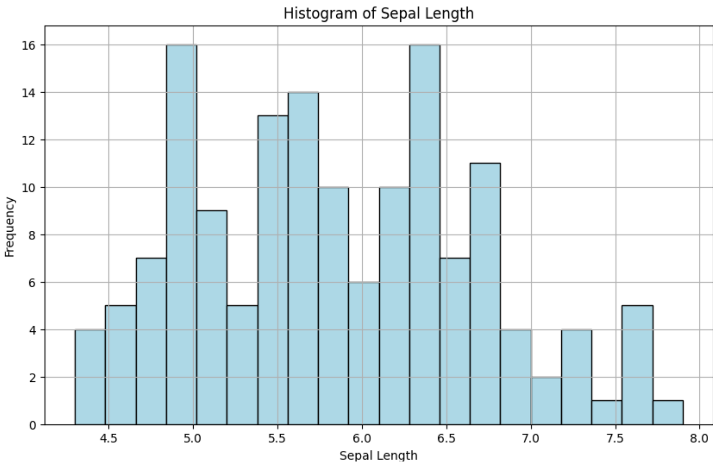
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
data = pd.read_csv(url, header=None, names=column_names)

# Normal Curves
plt.figure(figsize=(10, 6))
sns.kdeplot(data['sepal_length'], color='blue', label='Sepal Length', fill=True)
sns.kdeplot(data['sepal_width'], color='orange', label='Sepal Width', fill=True)
plt.title('Normal Curves for Sepal Dimensions')
plt.xlabel('Length/Width')
plt.ylabel('Density')
plt.legend()
plt.show()
# Density and Contour Plots
plt.figure(figsize=(10, 6))
sns.kdeplot(x=data['sepal_length'], y=data['sepal_width'], cmap='Blues', fill=True)
plt.title('Density and Contour Plot')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
# Correlation and Scatter Plots
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=data)
plt.title('Scatter Plot of Sepal Dimensions')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
# Histograms
plt.figure(figsize=(10, 6))
data['sepal_length'].hist(bins=20, color='lightblue', edgecolor='black')
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.show()
from mpl_toolkits.mplot3d import Axes3D

# Three-Dimensional Plotting
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data['sepal_length'], data['sepal_width'], data['petal_length'], c='r', marker='o')
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')
ax.set_zlabel('Petal Length')
plt.title('3D Scatter Plot')
plt.show()
```

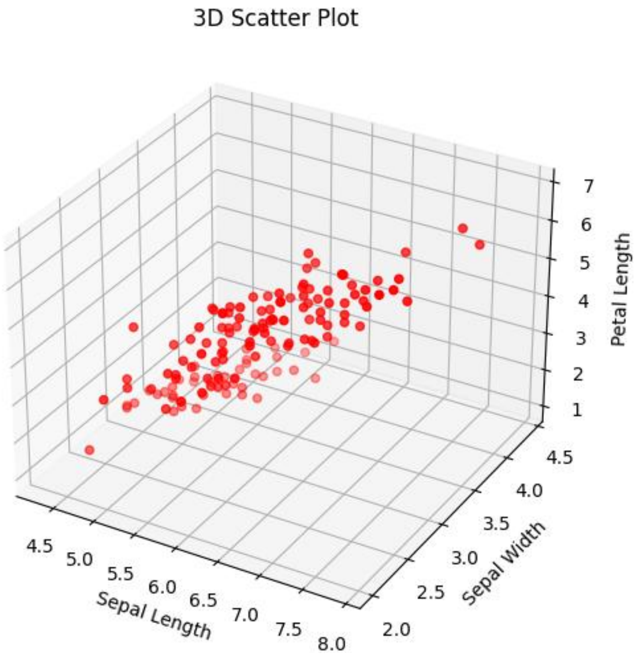
OUTPUT


Normal Curves for Sepal Dimensions


Density and Contour Plot


Scatter Plot of Sepal Dimensions

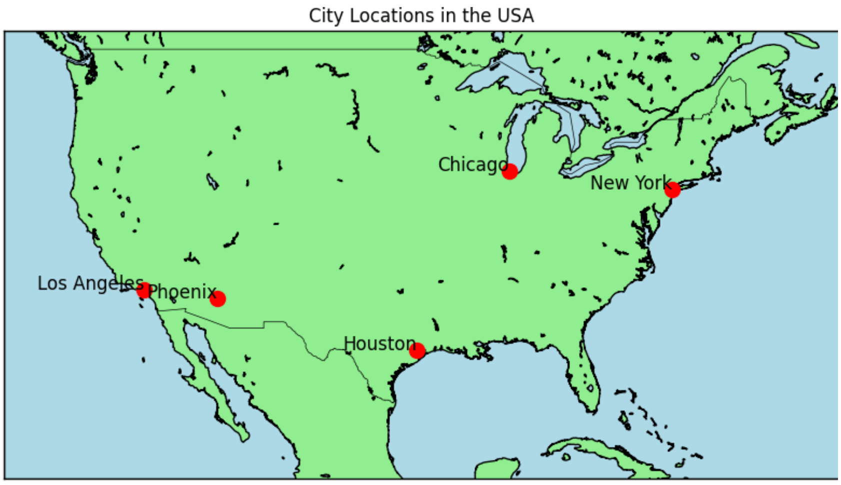

Histogram of Sepal Length


3D Scatter Plot

PROGRAM

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
# Step 1: Load Required Libraries (done above)
# Step 2: Load Geographic Data
# Example city data: list of cities with latitude and longitude
cities = {
    'New York': (40.7128, -74.0060),
    'Los Angeles': (34.0522, -118.2437),
    'Chicago': (41.8781, -87.6298),
    'Houston': (29.7604, -95.3698),
    'Phoenix': (33.4484, -112.0740)
}
# Step 3: Set Up the Base Map
plt.figure(figsize=(10, 8))
map = Basemap(projection='merc', llcrnrlat=20, urcrnrlat=50,
              llcrnrlon=-130, urcrnrlon=-60, lat_ts=20, resolution='i')
# Draw map details
map.drawcoastlines()
map.drawcountries()
map.drawmapboundary(fill_color='lightblue')
map.fillcontinents(color='lightgreen', lake_color='lightblue')
# Step 4: Plot Data Points
for city, (lat, lon) in cities.items():
    x, y = map(lon, lat)  # Convert lat/lon to x/y
    map.plot(x, y, 'ro', markersize=10)  # Plot city location
    plt.text(x, y, city, fontsize=12, ha='right')
# Step 5: Customize the Map
plt.title('City Locations in the USA')
# Step 6: Display the Map
plt.show()
```

OUTPUT


City Locations in the USA
Chicago
New York
Los Angeles
Phoenix
Houston