Objective:

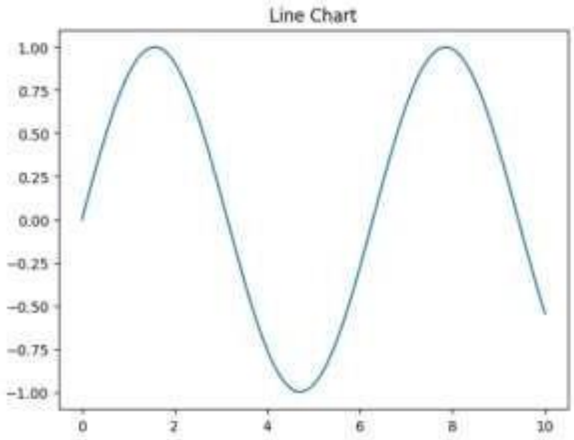
1. **Demonstrate all the basic plots using Matplotlib package and python programming.**

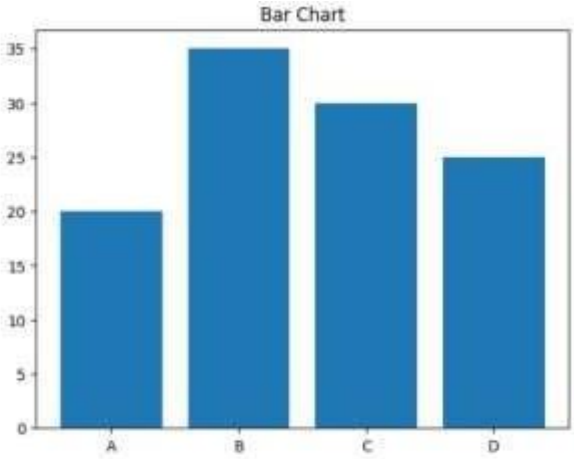
Procedure:

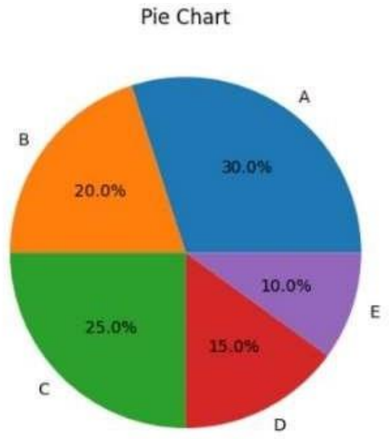
```
import matplotlib.pyplot as plt     #Used to create plots and visualizations.
import numpy as np          #A library for numerical computations, used here to generate data

# Generate some data for plotting
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.figure()        # Creates a new figure
plt.plot(x,y)       # Plots a line graph with `x` and `y`
plt.title("Line Chart")
categories=['A','B','C','D']
values=[20,35,30,25] plt.figure()
plt.bar(categories,values)
plt.title("Bar Chart")

x=np.random.randn(100)
y=np.random.randn(100)
colors=np.random.rand(100)
sizes=100*np.random.rand(100) plt.figure()      # Random sizes for each point
plt.scatter(x,y,c=colors, s=sizes, alpha=0.5)   #c assigns colors, s determines sizes, and alpha adjusts
                                                transparency.
plt.title("Scatter Plot")

sizes = [30, 20, 25, 15, 10]
labels = ['A', 'B', 'C', 'D', 'E']
plt.figure()

plt.pie(sizes, labels=labels, autopct="%1.1f%%") plt.title("Pie Chart")
plt.show()
                    #autopct shows the percentage values on the slices.
```

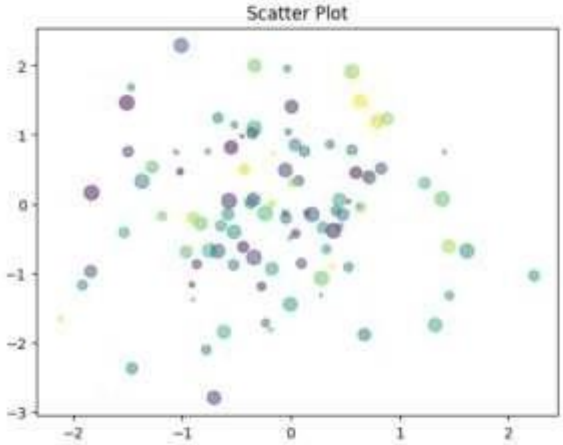
1)Line Chart
2)Bar Chart
3)Scatter Plot
4)Pie Chart

Output:


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


Bar Chart
35
30
25
20
15
10
5
0
A
B
C
D


Pie Chart
A
B
C
D
E
30.0%
20.0%
25.0%
15.0%
10.0%


Scatter Plot
2
1
0
−1
−2
−3
−2
−1
0
1
2

Objective:

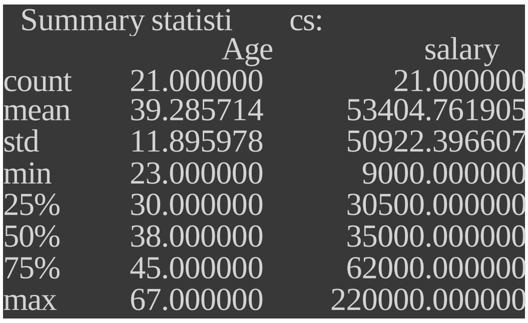
2. **Implement a python program to perform File Operations on Excel Dataset.**

Procedure:

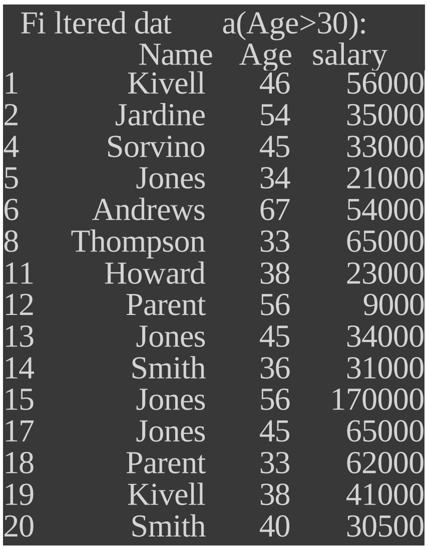
```
import pandas as pd
df = pd.read_excel('data.xlsx')

# Display the first few rows of the DataFrame
print("First few rows")
print(df.head())

# Display summary statistics of the DataFrame
print("\nSummary statistics:")
print(df.describe())

filtered_data = df[df['Age'] > 30]
print("\nFiltered data (Age > 30):")
print(filtered_data)

# Sort data by 'salary' in descending order
sorted_data = df.sort_values(by='salary', ascending=False)
print("\nSorted data (by Salary):")
print(sorted_data)

# Add a new column 'Bonus' which is 10% of the salary
df['Bonus'] = df['salary'] * 0.1
print("\nData with new column (Bonus):")
print(df)

# Write the updated DataFrame to a new Excel file
df.to_excel('Output.xlsx', index=False)
print("\nData written to Output.xlsx")
```

Output :

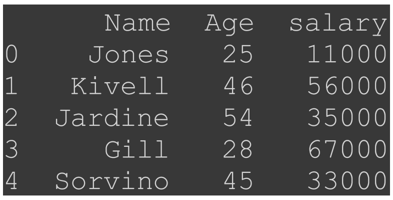
```
First few rows
```



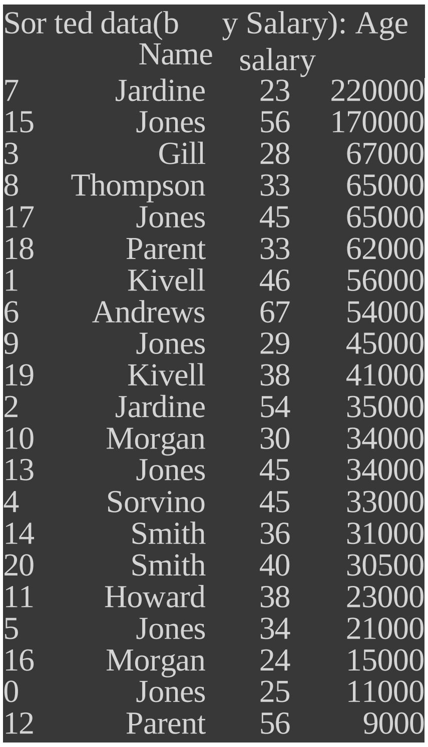
```
 Summary statistics:
              Age             salary
count   21.000000          21.000000
mean    39.285714       53404.761905
std     11.895978       50922.396607
min     23.000000        9000.000000
25%     30.000000       30500.000000
50%     38.000000       35000.000000
75%     45.000000       62000.000000
max     67.000000      220000.000000
```



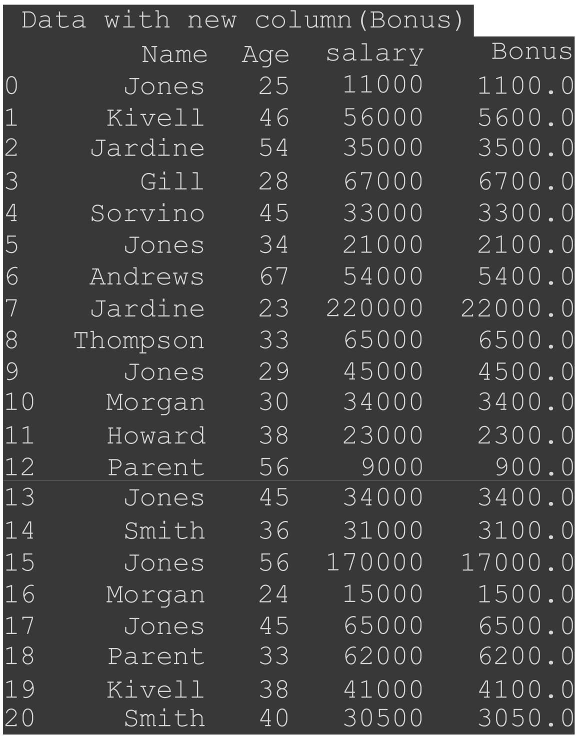
```
 Fi ltered dat     a(Age>30):
           Name   Age   salary
1        Kivell    46    56000
2       Jardine    54    35000
4       Sorvino    45    33000
5         Jones    34    21000
6       Andrews    67    54000
8      Thompson    33    65000
11       Howard    38    23000
12       Parent    56     9000
13        Jones    45    34000
14        Smith    36    31000
15        Jones    56   170000
17        Jones    45    65000
18       Parent    33    62000
19       Kivell    38    41000
20        Smith    40    30500
```



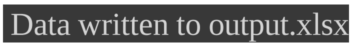
```
      Name  Age  salary
0    Jones   25   11000
1   Kivell   46   56000
2  Jardine   54   35000
3     Gill   28   67000
4  Sorvino   45   33000
```



```
Sor ted data(b     y Salary): Age
           Name   salary
7       Jardine    23   220000
15        Jones    56   170000
3          Gill    28    67000
8      Thompson    33    65000
17        Jones    45    65000
18       Parent    33    62000
1        Kivell    46    56000
6       Andrews    67    54000
9         Jones    29    45000
19       Kivell    38    41000
2       Jardine    54    35000
10       Morgan    30    34000
13        Jones    45    34000
4       Sorvino    45    33000
14        Smith    36    31000
20        Smith    40    30500
11       Howard    38    23000
5         Jones    34    21000
16       Morgan    24    15000
0         Jones    25    11000
12       Parent    56     9000
```



```
 Data with new column(Bonus)
         Name  Age  salary    Bonus
0       Jones   25   11000   1100.0
1      Kivell   46   56000   5600.0
2     Jardine   54   35000   3500.0
3        Gill   28   67000   6700.0
4     Sorvino   45   33000   3300.0
5       Jones   34   21000   2100.0
6     Andrews   67   54000   5400.0
7     Jardine   23  220000  22000.0
8    Thompson   33   65000   6500.0
9       Jones   29   45000   4500.0
10     Morgan   30   34000   3400.0
11     Howard   38   23000   2300.0
12     Parent   56    9000    900.0
13      Jones   45   34000   3400.0
14      Smith   36   31000   3100.0
15      Jones   56  170000  17000.0
16     Morgan   24   15000   1500.0
17      Jones   45   65000   6500.0
18     Parent   33   62000   6200.0
19     Kivell   38   41000   4100.0
20      Smith   40   30500   3050.0
```



```
Data written to output.xlsx
```

Objective:

3. **Write a python program to perform Array operations using the Numpy package.**

Procedure:

```
import numpy as np #
Create arrays
a = np.array([1, 2, 3, 4, 5])
b = np.array([6, 7, 8, 9, 10])


print("Array a", a)
print("Array b", b)
print("Sum of array a and b", np.add(a,b))
print("Difference of array a and b", np.subtract(a,b))
print("Product of arrays a and b", np.multiply(a,b))
print("Division of arrays a and b", np.divide(a,b))
print("Square root of array a:",np.sqrt(a))
print("Exponential of array a:",np.exp(a))
print("Minimum value of array a:",np.min(a))
print("Maximum value of array b:",np.max(b))
print("Mean of array a:",np.mean(a)) print("Standard
deviation of array b:",np.std(b)) print("Sum of
elements in array a:",np.sum(a))
c=np.array([[1,2],[3,4],[5,6]])
print("Array c:")
print(c)
print("Reshaped array c:")
print(np.reshape(c,(2,3)))

d=np.array([[1,2,3],[4,5,6]])
print("Array d:")
print(d)
print("Transposed array d:")
print(np.transpose(d))
```

Output:

```
Array a [1 2 3 4 5]

Array b [ 6  7  8  9 10]

Sum of array a and b [ 7  9 11 13 15]

Difference of array a and b [-5 -5 -5 -5 -5]

Product of arrays a and b [ 6 14 24 36 50]
Division of arrays a and b [0.16666667     0.28571429     0.44444444 0.5    ]
0.375
Square root of array a: [1.       1.41421356 1.73205081       2.23606798]
2.

Exponential of array a: [  2.71828183   7.3890561   20.08553692  54.59815003 148.4131591 ]

Minimum value of array a: 1
Maximum value of array b: 10
Mean of array a: 3.0
Standard deviation of array b:
1.4142135623730951 Sum of elements in array
a: 15
Array c:
[[1 2]
[3 4]
[5 6]]
Reshaped array c:
[[1 2 3]
[4 5 6]]
```

```
Array d:
[[1 2 3]
[4 5 6]]

Transposed array d:
[[1 4]
[2 5]
[3 6]]
```

Objective:

4. **Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.**

Procedure:

```
import numpy as np
x=np.array(([2,9],[1,9],[3,6]),dtype=float)
y=np.array(([92],[86],[89]),dtype=float)
x=x/np.amax(x,axis=0)
y=y/100

def sigmoid(x):
return 1/(1+np.exp(-x))
def derivation_sigmoid(x): return
x*(1-x)
epoch=5000 lr=0.1
inputlayer_neurons=2
hiddenlayer_neurons=3
outputlayer_neurons=1

wb=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bb=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,outputlayer_neurons))
bout=np.random.uniform(size=(1,outputlayer_neurons))
for i in range(epoch):
hinp1=np.dot(x,wb)

hinp=hinp1+bb hlayer_act=sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp=outinp1+bout
output=sigmoid(outinp)
```

```
EO=y-output outgrad=derivation_sigmoid(output)
d_output=EO*outgrad EH=d_output.dot(wout.T)
hiddengrad=derivation_sigmoid(hlayer_act)
d_hiddenlayer=EH*hiddengrad
wout+=hlayer_act.T.dot(d_output)*lr
wb+=x.T.dot(d_output)*lr


print("Inpput:\n" +str(x))
print("Actual:\n"+str(y))
print("Predicted:\n",output)
```

Output:

```
Input:
[[0.66666667  1.          ]
[0.33333333    1.         ]
[1.          0.66666667  ]]
Actual:
[[0.92
[0.86]
[0.89]]


Predicted:
[[0.89184048]
[0.88433366]
[0.89399225]]
```

Objective:

5. **Demonstrate Linear Regression operation using python programming.**

Procedure:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt import

seaborn as sns

dataset = pd.read_csv('advertising.csv')

dataset.head(10)

dataset.shape

dataset.isna().sum()

dataset.duplicated().any()

fig, axs = plt.subplots(3, figsize = (5,5))

plt1 = sns.boxplot(dataset['TV'], ax = axs[0])

plt2 = sns.boxplot(dataset['Newspaper'], ax = axs[1])

plt3 = sns.boxplot(dataset['Radio'], ax = axs[2])

plt.tight_layout()

sns.distplot(dataset['Sales']);

sns.pairplot(dataset, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=4, aspect=1,
kind='scatter')

plt.show()

sns.heatmap(dataset.corr(), annot = True)

plt.show()

from sklearn.model_selection

import train_test_split from sklearn.linear_model
import LinearRegression from sklearn

import metrics

x = dataset[['TV']]

y = dataset['Sales']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 100) slr=

LinearRegression()

slr.fit(x_train, y_train) print('Intercept: ',

slr.intercept_) print('Coefficient:',

slr.coef_)

print('Regression Equation: Sales = 6.948 + 0.054 * TV')

plt.scatter(x_train, y_train)

plt.plot(x_train, 6.948 + 0.054*x_train, 'r') plt.show()

#Prediction of Test and Training set result

y_pred_slr= slr.predict(x_test) x_pred_slr=

slr.predict(x_train)

print("Prediction for test set: {}".format(y_pred_slr))

slr_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred_slr})

slr_diff

#Predict for any value slr.predict([[56]])

# print the R-squared value for the model from

sklearn.metrics import accuracy_score

print('R squared value of the model: {:.2f}'.format(slr.score(x,y)*100))
```

Output:

https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjQ4ODAwOTUyMTg2/details

Objective:

6. **Train a regularized logistic regression classifier on the in-build iris dataset using scikit- learn. Train the model and report the best classification accuracy.**

Procedure:

```
# Importing the necessary libraries

import numpy as np

import matplotlib.pyplot as plt import

pandas as pd

# Importing the dataset

dataset = pd.read_csv('iris.csv')

dataset.describe()   dataset.info()

# Splitting the dataset into the Training set and Test set

X = dataset.iloc[:, [0,1,2, 3]].values

y = dataset.iloc[:, 4].values


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0) #

Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

# Fitting Logistic Regression to the Training set

from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0, solver='lbfgs', multi_class='auto')
```

```
classifier.fit(X_train, y_train)

# Predicting the Test set results

y_pred = classifier.predict(X_test)

# Predict probabilities

probs_y=classifier.predict_proba(X_test)

probs_y = np.round(probs_y, 2)

res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y_test", "y_pred", "Setosa(%)", "versicolor(%)", "virginica(%)\n")

res += "-"*65+"\n"

res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c) for x, y, a, b, c in zip(y_test, y_pred, probs_y[:,0], probs_y[:,1], probs_y[:,2]))
res += "\n"+"-"*65+"\n"
print(res)
# Making the Confusion Matrix


from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
# Plot confusion matrix
import seaborn as sns
import pandas as pd
# confusion matrix sns heatmap
## https://www.kaggle.com/agungor2/various-confusion-matrix-plots
ax = plt.axes()
df_cm = cm
sns.heatmap(df_cm, annot=True, annot_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax )
ax.set_title('Confusion Matrix')
plt.show()
```

Output:

https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjY1Mjg0NzYwOTkz/details

Objective:

7. **Write a python program to perform Data Manipulation operations using Pandas package.**

Procedure:

```
import pandas as pd
data={
        'Name':['John','Emma','Sant','Lisa','Tom'],
        'Age':[25,30,28,32,27],
         'Country':['USA','Canada','India','UK','Australia'],
         'Salary':[50000,60000,70000,80000,65000]
      }

df=pd.DataFrame(data)
print("Original DataFrame")
print(df)


name_age=df[['Name','Age']]
print("Original DataFrame") print(df)
name_age=df[['Name','Age']]
print("Name and Age columns")
print(name_age)
```

```
filtered_df=df[df['Country']=='USA']
print("\nfiltered DataFrame(Country='USA')")
print(filtered_df)

sorted_df=df.sort_values("Salary",ascending=False) print("\nsorted
DataFrame(by ssalary in descending order)") print(sorted_df)
average_Salary=df['Salary'].mean()

print("\nAverage salary",average_Salary)


df['Experience']=[3,6,4,8,5] print("\nDataFrame
with added experience") print(df)
df.loc[df['Name']=='Emma','Salary']=65000
print("\nDataFrame with updating emma salary")
print(df)

df.drop('Experience',axis=1)  print("\nDataFrame after deleting the column ")
print(df)
```

Output:

```
Original DataFrame
Name             Age       Country       Salary
a.  John         25         USA          50000
b.  Emma         30         Canada       60000
c.  Sant         28         India        70000
d.  Lisa         32         UK           80000
e.  Tom          27         Australia    65000

Name and Age columns
0        John            25
1        Emma            30
2        Sant            28
3        Lisa            32
4        Tom             27
```

```
filtered DataFrame(Country='USA')

  Name  Age Country  Salary

0  John   25     USA   50000

sorted DataFrame(by ssalary in descending order)

   Name  Age    Country  Salary

3  Lisa   32        UK   80000

2  Sant   28     India   70000

4   Tom   27  Australia   65000

1  Emma   30    Canada   60000

0  John   25       USA   50000
```

```
Average salary 65000.0


DataFrame with added experience

   Name  Age    Country  Salary  Experience

0  John   25        USA   50000
                                   3
1  Emma   30     Canada   60000    6

2  Sant   28      India   70000    4

3  Lisa   32         UK   80000    8

4   Tom   27  Australia   65000    5


DataFrame with updating emma salary

   Name  Age    Country  Salary  Experience

0  John   25       USA   50000         3

1  Emma   30    Canada   65000           6

2  Sant   28     India   70000        4

3  Lisa   32        UK   80000        8

4   Tom   27  Australia   65000         5


DataFrame after deleting the column

   Name  Age    Country  Salary  Experience

0  John   25       USA   50000         3

1  Emma   30    Canada   65000           6

2  Sant   28     India   70000        4

3  Lisa   32        UK   80000        8

4   Tom   27  Australia   65000         5
```