WEEK-1

Aim: Extract data from different file formats and display the summary statistics.

Description: Sometimes work with some datasets must have mostly worked with .csv(Comma Separated Value) files only. They are really a great starting point in applying Data Science techniques and algorithms. But many of us will land up in Data Science firms or take up real-world projects in Data Science sooner or later. Unfortunately, in real-world projects, the data won't be available to us in a neat .csv file. There we have to extract data from different sources like images, pdf files, doc files, image files, etc. In this article, we will see the perfect start to tackle those situations.

Program:

❖ **Extracting data from excel file**

```
from google.colab import files
uploaded = files.upload()

import io
import pandas as pd
df = pd.read_excel(io.BytesIO(uploaded['Book1.xlsx']))
# df
print(df)
print()
print(df.describe())
```

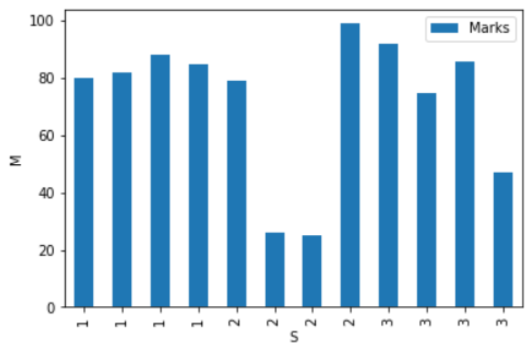
O/P:

```
   Marks  Section
0     80        1
1     82        1
2     88        1
3     85        2
4     79        2
5     26        2

           Marks   Section
count   6.000000  6.000000
mean   73.333333  1.500000
std    23.423635  0.547723
min    26.000000  1.000000
25%    79.250000  1.000000
50%    81.000000  1.500000
75%    84.250000  2.000000
max    88.000000  2.000000
```

```
df.plot(kind = 'bar', x='Section', y='Marks', xlabel = 'S', ylabel='M', legend='single element')
```

O/P:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15954cafd0>
```



❖ Extracting data from csv file

```
from google.colab import files
uploaded = files.upload()
```

```
import io
import pandas as pd
df = pd.read_csv(io.BytesIO(uploaded['sample-csv.csv']))
print(df)
print()
print(df.describe())
```

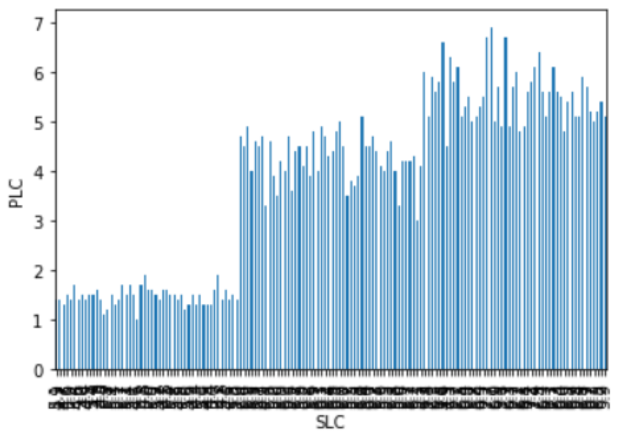
O/P:

```
    fixed acidity  volatile acidity  citric acid  residual sugar
0             7.4             0.700         0.00             1.9
1             7.8             0.880         0.00             2.6
2             7.8             0.760         0.04             2.3
3            11.2             0.280         0.56             1.9
4             7.4             0.700         0.00             1.9
5             7.4             0.660         0.00             1.8
6             7.9             0.600         0.06             1.6
7             7.3             0.650         0.00             1.2
8             7.8             0.580         0.02             2.0
9             7.5             0.500         0.36             6.1
10            6.7             0.580         0.08             1.8
11            7.5             0.500         0.36             6.1
12            5.6             0.615         0.00             1.6
13            7.8             0.610         0.29             1.6
14            8.9             0.620         0.18             3.8
15            8.9             0.620         0.19             3.9
       fixed acidity  volatile acidity  citric acid  residual sugar
count      33.000000         33.000000    33.000000       33.000000
mean        7.733333          0.549545     0.170303        2.366667
std         0.954812          0.154264     0.175686        1.189187
min         5.600000          0.220000     0.000000        1.200000
25%         7.400000          0.430000     0.000000        1.800000
50%         7.800000          0.590000     0.120000        1.900000
75%         7.900000          0.655000     0.280000        2.400000
max        11.200000          0.880000     0.560000        6.100000
```

```
df.plot(kind = 'bar', x='SepalLengthCm', y='PetalLengthCm', xlabel = 'SLC', ylabel='PLC',
legend=False)
```

O/P:

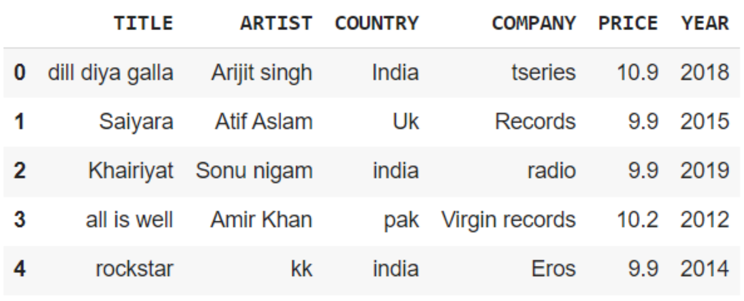
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1595132970>
```



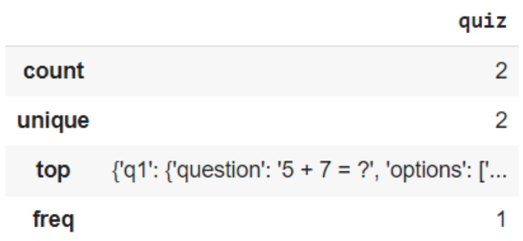
❖ Extracting data from xml file

```
import pandas as pd
df = pd.read_xml('Sample-XML-Files.xml')
df.head()
df.describe()
```

O/P:



	TITLE	ARTIST	COUNTRY	COMPANY	PRICE	YEAR
0	dill diya galla	Arijit singh	India	tseries	10.9	2018
1	Saiyara	Atif Aslam	Uk	Records	9.9	2015
2	Khairiyat	Sonu nigam	india	radio	9.9	2019
3	all is well	Amir Khan	pak	Virgin records	10.2	2012
4	rockstar	kk	india	Eros	9.9	2014



	quiz
count	2
unique	2
top	{'q1': {'question': '5 + 7 = ?', 'options': ['...
freq	1

❖ Extracting data from zip file

```
from google.colab import files
uploaded = files.upload()

import zipfile
filename = zipfile.ZipFile('zipfile.zip', 'r')
df = filename.read('sample-csv.csv')
print(df)
```

O/P:

```
b'fixed acidity,volatile acidity,citric acid,residual sugar\r\n7.4,0.7,0,1.9\r\n7.8,0.88,0,2.6\r\n7.8,0.76,0.04,2.3
```

❖ Extracting data from text file

```
from google.colab import files
uploaded = files.upload()

f1 = open("data.txt", 'r')
print(f1.read())
f1.close()
```

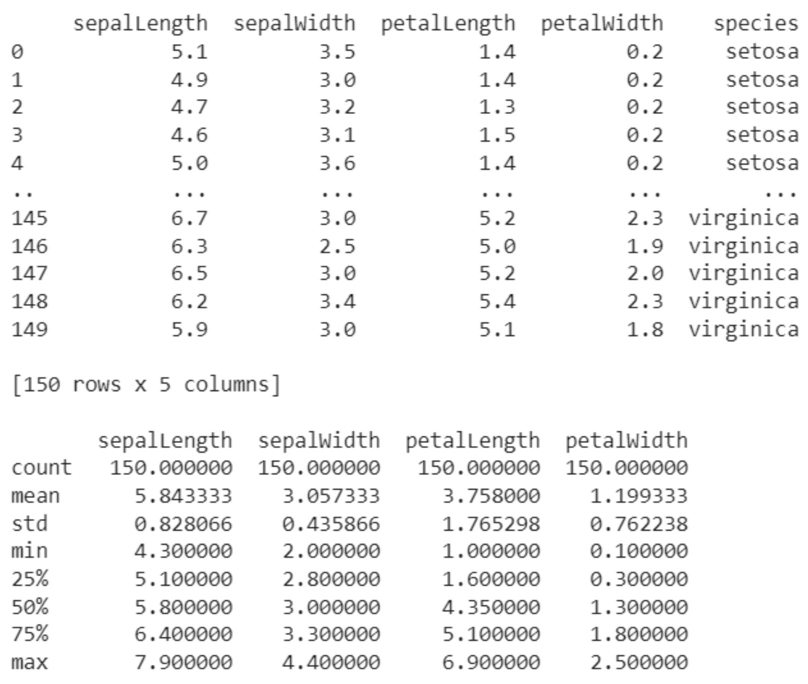
O/P:

```
jake
raj
sunny
praneeth
```

❖ Extracting data from json file

```
from google.colab import files
uploaded = files.upload()

import io
import pandas as pd
df = pd.read_json(io.BytesIO(uploaded['jsondata.json']))
print(df)
print()
print(df.describe())
```

O/P:



```
     sepalLength  sepalWidth  petalLength  petalWidth    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
..           ...         ...          ...         ...        ...
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]

       sepalLength  sepalWidth  petalLength  petalWidth
count   150.000000  150.000000   150.000000  150.000000
mean      5.843333    3.057333     3.758000    1.199333
std       0.828066    0.435866     1.765298    0.762238
min       4.300000    2.000000     1.000000    0.100000
25%       5.100000    2.800000     1.600000    0.300000
50%       5.800000    3.000000     4.350000    1.300000
75%       6.400000    3.300000     5.100000    1.800000
max       7.900000    4.400000     6.900000    2.500000
```

❖ Finding covariance and correlation

Program:

```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
df=pd.read_csv("Iris.csv")
print("summary stats")
print(df.describe())

sl=df['SepalLengthCm'].values
pl=df['PetalLengthCm'].values
mat=np.stack((sl,pl),axis=0)
covmat=np.cov(mat)
print("covariance matrix")
print(covmat)
print("correlation matrix")
corrmat = df.corr()
print(corrmat)
```

Output:

```
summary stats
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000

covariance matrix
[[0.68569351 1.27368233]
 [1.27368233 3.11317942]]

correlation matrix
                     Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  \
Id             1.000000       0.716676     -0.397729       0.882747
SepalLengthCm  0.716676       1.000000     -0.109369       0.871754
SepalWidthCm  -0.397729      -0.109369      1.000000      -0.420516
PetalLengthCm  0.882747       0.871754     -0.420516       1.000000
PetalWidthCm   0.899759       0.817954     -0.356544       0.962757

               PetalWidthCm
Id                 0.899759
SepalLengthCm      0.817954
SepalWidthCm      -0.356544
PetalLengthCm      0.962757
PetalWidthCm       1.000000
```

WEEK-2

Aim: Write a program that extracts the words (features) used in a sentence.

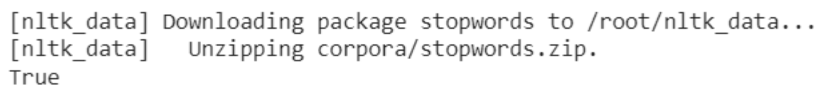
Description: First, we have to extract all words from a String, as a string may contain many sentences with punctuation marks. For extracting words from a String first replace all the punctuation marks with spaces and then find the unique words in a sentence.

Program:

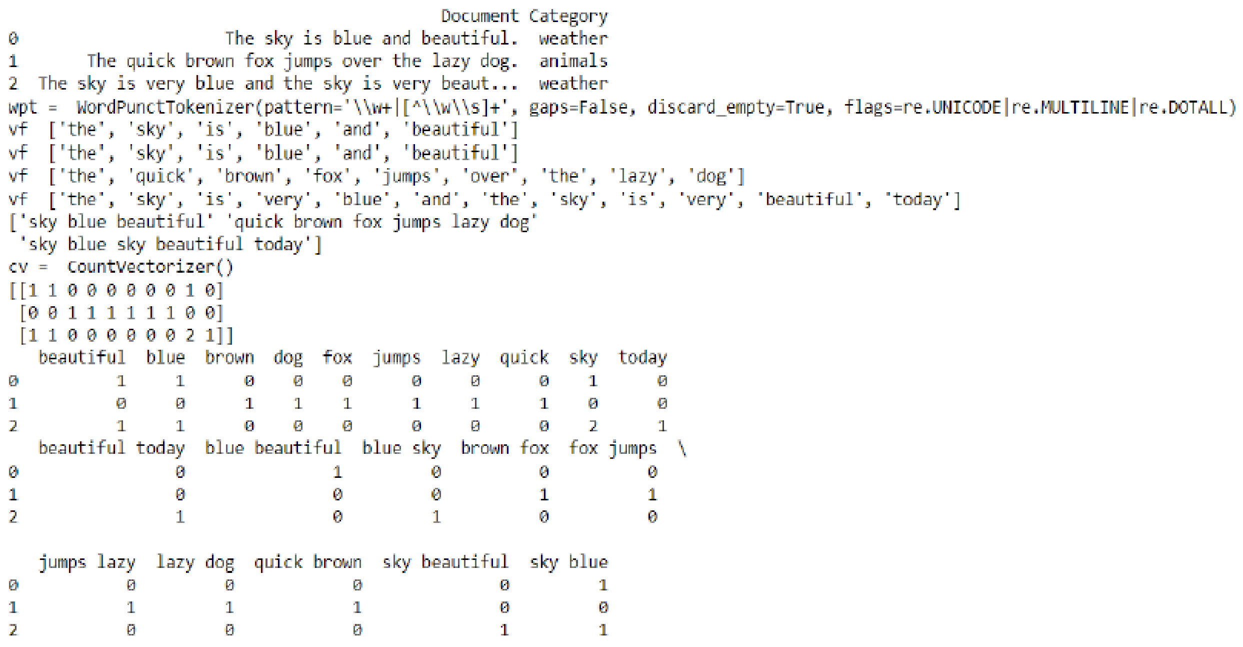
```
import pandas as pd
import numpy as np
import re
import nltk
nltk.download('stopwords')

corpus = ['The sky is blue and beautiful.',
'The quick brown fox jumps over the lazy dog.',
'The sky is very blue and the sky is very beautiful today']
labels = ['weather', 'animals', 'weather']
corpus_df = pd.DataFrame({'Document': corpus,
'Category': labels})
print(corpus_df)

wpt = nltk.WordPunctTokenizer()
print('wpt = ',wpt)
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
  doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc)
  doc = doc.lower()
  doc = doc.strip()
  tokens = wpt.tokenize(doc)
  print('vf ',tokens)
  filtered_tokens = [token for token in tokens if token not in stop_words]
  doc = ' '.join(filtered_tokens)
  return doc

normalize_corpus = np.vectorize(normalize_document)
norm_corpus = normalize_corpus(corpus)
print(norm_corpus)

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
print('cv = ', cv)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix = cv_matrix.toarray()
print(cv_matrix)
```

```
vocab = cv.get_feature_names()
print(pd.DataFrame(cv_matrix, columns=vocab))

bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)
bv_matrix = bv_matrix.toarray()
vocab = bv.get_feature_names()
print(pd.DataFrame(bv_matrix, columns=vocab))
```

Output:



```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```



```
                                           Document Category
0                     The sky is blue and beautiful.  weather
1      The quick brown fox jumps over the lazy dog.  animals
2  The sky is very blue and the sky is very beaut...  weather
wpt =  WordPunctTokenizer(pattern='\\w+|[^\\w\\s]+', gaps=False, discard_empty=True, flags=re.UNICODE|re.MULTILINE|re.DOTALL)
vf  ['the', 'sky', 'is', 'blue', 'and', 'beautiful']
vf  ['the', 'sky', 'is', 'blue', 'and', 'beautiful']
vf  ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
vf  ['the', 'sky', 'is', 'very', 'blue', 'and', 'the', 'sky', 'is', 'very', 'beautiful', 'today']
['sky blue beautiful' 'quick brown fox jumps lazy dog'
 'sky blue sky beautiful today']
cv =  CountVectorizer()
[[1 1 0 0 0 0 0 0 1 0]
 [0 0 1 1 1 1 1 1 0 0]
 [1 1 0 0 0 0 0 0 2 1]]
   beautiful  blue  brown  dog  fox  jumps  lazy  quick  sky  today
0          1     1      0    0    0      0     0      0    1      0
1          0     0      1    1    1      1     1      1    0      0
2          1     1      0    0    0      0     0      0    2      1
   beautiful today  blue beautiful  blue sky  brown fox  fox jumps  \
0                0               1         0          0          0
1                0               0         0          1          1
2                1               0         1          0          0

   jumps lazy  lazy dog  quick brown  sky beautiful  sky blue
0           0         0            0              0         1
1           1         1            1              0         0
2           0         0            0              1         1
```

WEEK-3

Aim: Write a program for edge detection to extract edge-based features from a sample image.

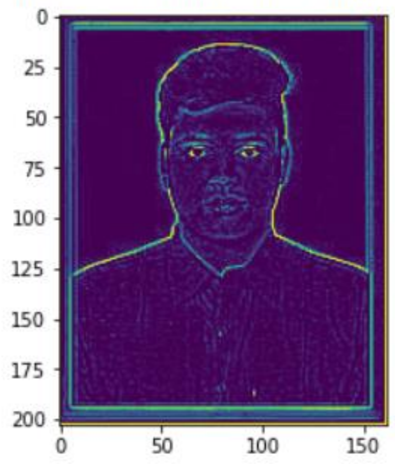
Description: Edge detection is an image processing technique for finding the boundaries of an object in the given image. The edges are the part of the image that represents the boundary or the shape of the object in the image. Also, the pixel values around the edge show a significant difference or a sudden change in the pixel values. Based on this fact we can identify which pixels represent the edge or which pixel lie on the edge.

Program:

```
from google.colab import files
uploaded = files.upload()

from PIL import Image, ImageFilter
import matplotlib.pyplot as plt
image = Image.open("resized photo for gvp.jpg")
image = image.convert("L")
image = image.filter(ImageFilter.FIND_EDGES)
image.save("Edge_Sample.png")
plt.imshow(image)
```

Output:

```
<matplotlib.image.AxesImage at 0x7fb6d4102df0>
```



WEEK-5

Aim: Write a program to perform Exploratory Data Analysis on real time datasets.

a) Univariate Analysis b) Multivariate Analysis c) Visualization using correlation matrix

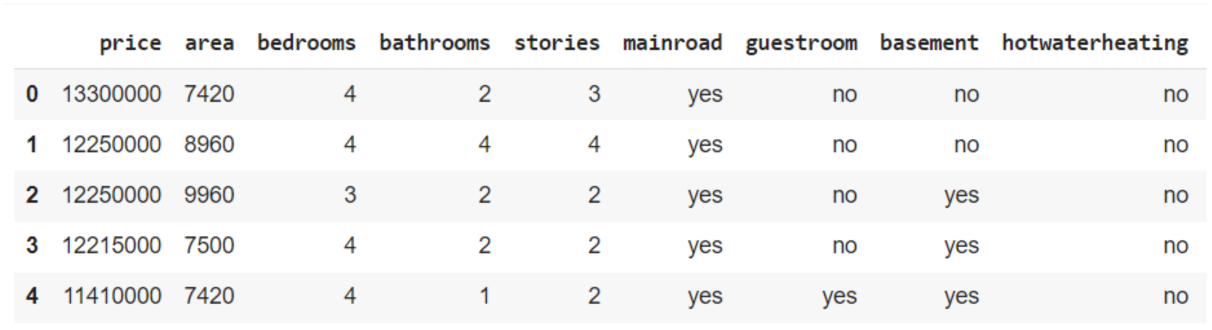
Description: Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations. The term univariate analysis refers to the analysis of one variable. There are three common ways to perform univariate analysis on one variable: Summary statistics – Measures the center and spread of values, Frequency table – Describes how often different values occur, Charts – Used to visualize the distribution of values. The term bivariate analysis refers to the analysis of two variables. The purpose of bivariate analysis is to understand the relationship between two variables. There are three common ways to perform bivariate analysis: Scatterplots, Correlation Coefficients, Simple Linear Regression. The term multivariate analysis refers to the analysis of more than two variables.

❖ Univariate Analysis

Program:

```
import pandas as pd
df=pd.read_csv('Hotels.csv')
df.head()
```

O/P:



	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	13300000	7420	4	2	3	yes	no	no	no
1	12250000	8960	4	4	4	yes	no	no	no
2	12250000	9960	3	2	2	yes	no	yes	no
3	12215000	7500	4	2	2	yes	no	yes	no
4	11410000	7420	4	1	2	yes	yes	yes	no

****1.Summary Statistics****

df['bathrooms'].mean()
O/P: 3.8

df['bathrooms'].median()
O/P: 4.0

df['bathrooms']. std()
O/P: 0.44721359549990

****2.Create Frequency Table****

df['bathrooms'].value_counts()

O/P:

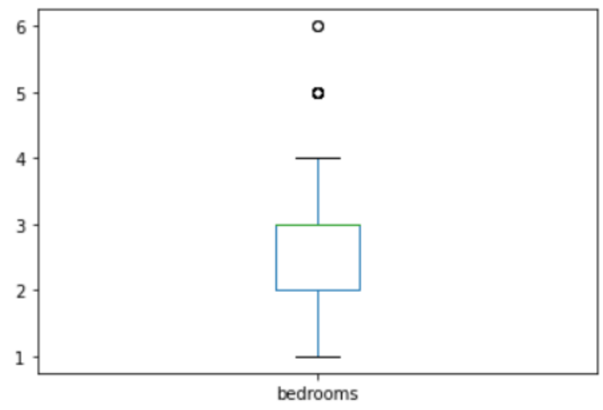
```
3    300
2    136
4     95
5     10
6      2
1      2
Name: bedrooms, dtype: int64
```

****3. Create Charts****

import matplotlib.pyplot as plt
df.boxplot(column=['bedrooms'],grid=False)

O/P:

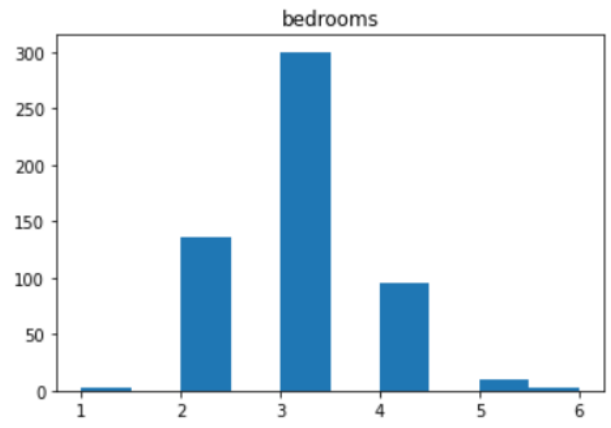
```
<AxesSubplot:>
```



import matplotlib.pyplot as plt
df.hist(column=['bedrooms'],grid=False)

O/P:

```
array([[<AxesSubplot:title={'center':'bedrooms'}>]], dtype=object)
```



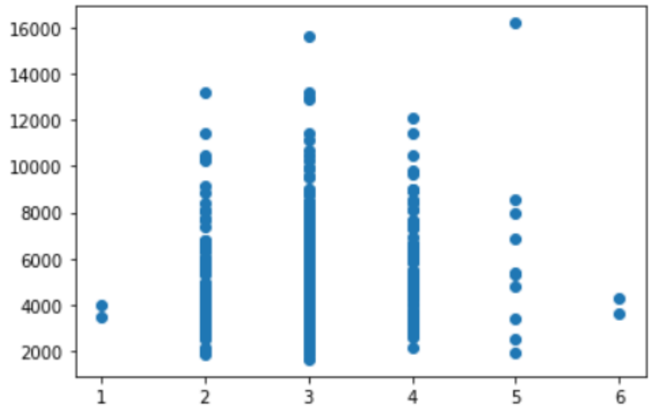
❖ BIVARIATE ANALYSIS

****1. Scatterplots****

plt.scatter(df['bedrooms'],df.area)

O/P:

```
<matplotlib.collections.PathCollection at 0x7f72fd4c8c40>
```



****2. Correlation Coefficients****

corr=df.corr()
corr

O/P:

	price	area	bedrooms	bathrooms	stories	parking
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.384394
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.352980
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	0.139270
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.177496
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.045547
parking	0.384394	0.352980	0.139270	0.177496	0.045547	1.000000

****3. Simple Linear Regression****

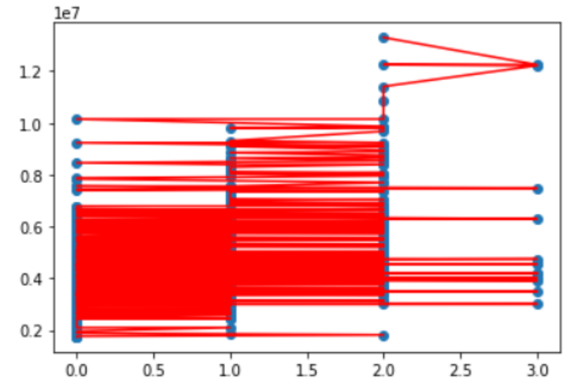
from sklearn.linear_model import LinearRegression
model=LinearRegression()

x=df[['parking']]
y=df['price']
model.fit(x,y)

y_pred=model.predict(x)
plt.scatter(x, y)
plt.plot(x,y, color='red')

O/P:

```
[<matplotlib.lines.Line2D at 0x7f72fd306670>]
```



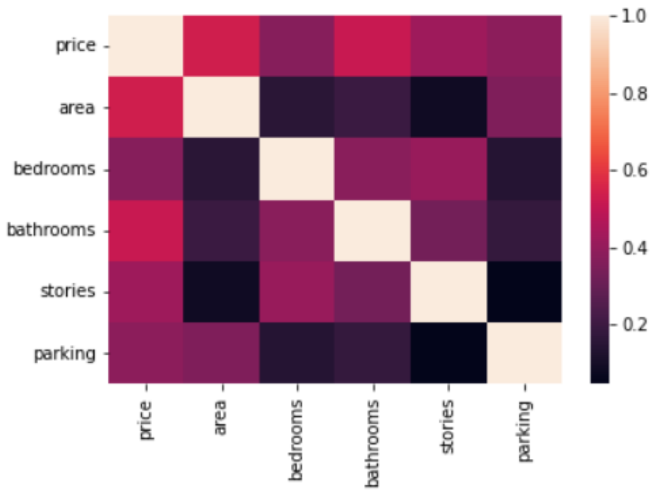
❖ VISUALIZATION USING CORRELATION MATRIX

Program:

```
corr=df.corr()
sns.heatmap(corr)
```

Output:

```
<AxesSubplot:>
```



WEEK-6

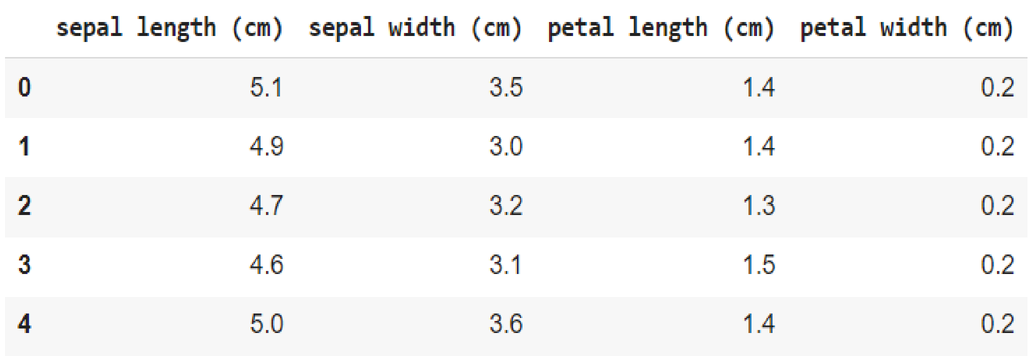
Aim: Write a program to perform Dimensionality Reduction using Principle Component Analysis techniques on real time data sets.

Description: Dimensionality Reduction is a statistical/ML-based technique wherein we try to reduce the number of features in our dataset and obtain a dataset with an optimal number of dimensions. One of the most common ways to accomplish Dimensionality Reduction is Feature Extraction. Principal Component Analysis is a technique of feature extraction that maps a higher dimensional feature space to a lower-dimensional feature space. The new features obtained after applying PCA are called Principal Components and are denoted as ***PCi (i=1,2,3…n)***.

Program:

```
from sklearn import datasets
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns

iris=datasets.load_iris()
df=pd.DataFrame(iris['data'],columns=iris['feature_names'])
df.head()

scalar=StandardScaler()
scaled_data=pd.DataFrame(scalar.fit_transform(df))
scaled_data

sns.heatmap(scaled_data.corr())

pca=PCA(n_components=3)
pca.fit(scaled_data)
data_pca=pca.transform(scaled_data)
data_pca=pd.DataFrame(data_pca,columns=['PC1','PC2','PC3'])
data_pca.head()

sns.heatmap(data_pca.corr())
```

Output:

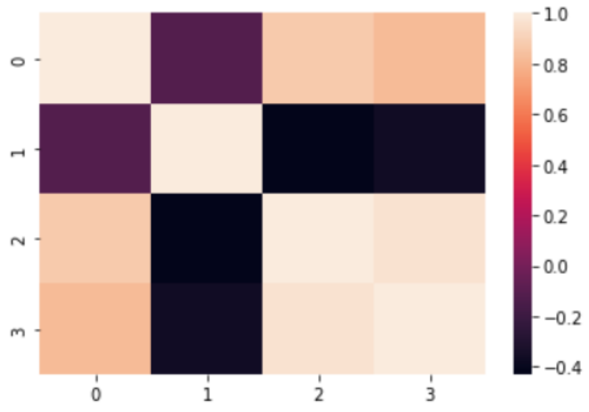


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	0	1	2	3
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444
...	...	...	...	...
145	1.038005	-0.131979	0.819596	1.448832
146	0.553333	-1.282963	0.705921	0.922303
147	0.795669	-0.131979	0.819596	1.053935
148	0.432165	0.788808	0.933271	1.448832
149	0.068662	-0.131979	0.762758	0.790671

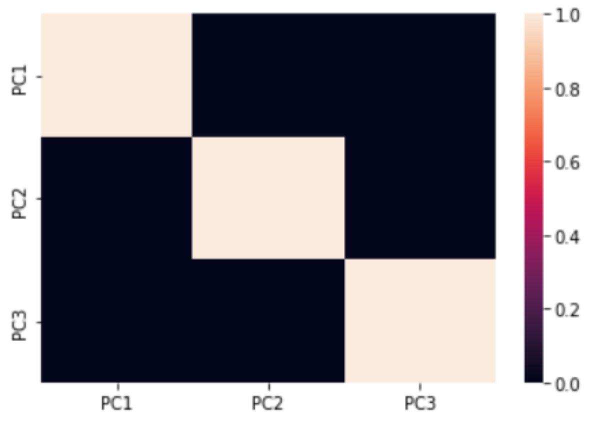
150 rows × 4 columns

```
<AxesSubplot:>
```



	PC1	PC2	PC3
0	-2.264703	0.480027	-0.127706
1	-2.080961	-0.674134	-0.234609
2	-2.364229	-0.341908	0.044201
3	-2.299384	-0.597395	0.091290
4	-2.389842	0.646835	0.015738

```
<AxesSubplot:>
```



WEEK-8

Aim: Write a program to implement the linear Regression for a sample training data set stored as a .CSV file

Description: Linear Regression is a machine learning algorithm based on supervised learning. Regression models a target prediction value based on independent variables. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. The linear regression model provides a sloped straight line representing the relationship between the variables. Mathematically, we can represent a linear regression as: $\mathbf{y= a_0+a_1x+ \varepsilon}$. Here, Y= Dependent Variable (Target Variable), X= Independent Variable (predictor Variable), a0= intercept of the line (Gives an additional degree of freedom), a1=Linear regression coefficient (scale factor to each input value), ε = random error,

Program:

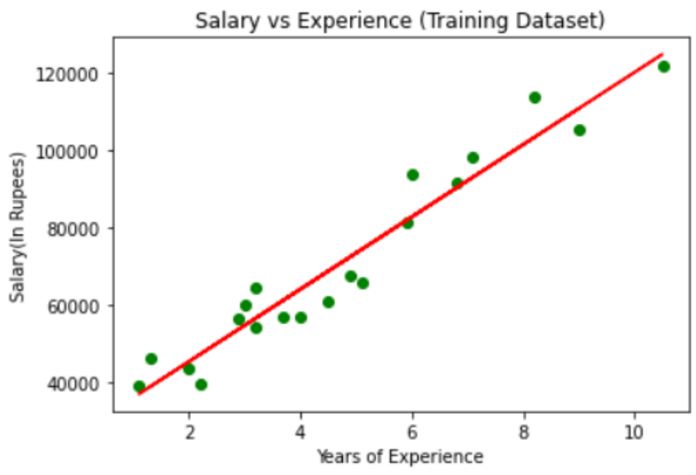
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

data_set= pd.read_csv('salary_data.csv')
data_set.head()

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)

from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)

#Prediction of Test set result
#y_pred= regressor.predict(x_test)
y_pred= regressor.predict(5)

#visualizing the Train set results
plt.scatter(x_train, y_train, color="green")
plt.plot(x_train, x_pred, color="red")
plt.title("Salary vs Experience (Training Dataset)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary(In Rupees)")
plt.show()

#visualizing the Test set results
plt.scatter(x_test, y_test, color="blue")
plt.plot(x_train, x_pred, color="red")
plt.title("Salary vs Experience (Test Dataset)")
plt.xlabel("Years of Experience")
plt.ylabel("Salary(In Rupees)")
plt.show()
```

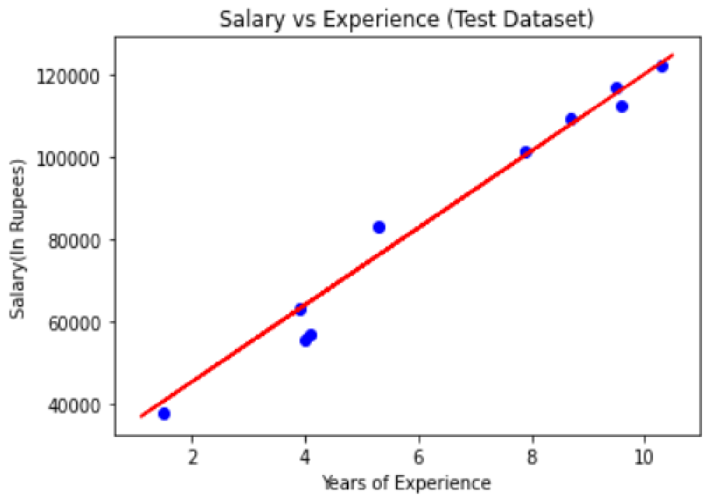
Output:

```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
```

```
▾ LinearRegression
LinearRegression()
```

```
[73545.90445964]
```


Salary vs Experience (Training Dataset)
Salary(In Rupees)
Years of Experience


Salary vs Experience (Test Dataset)
Salary(In Rupees)
Years of Experience