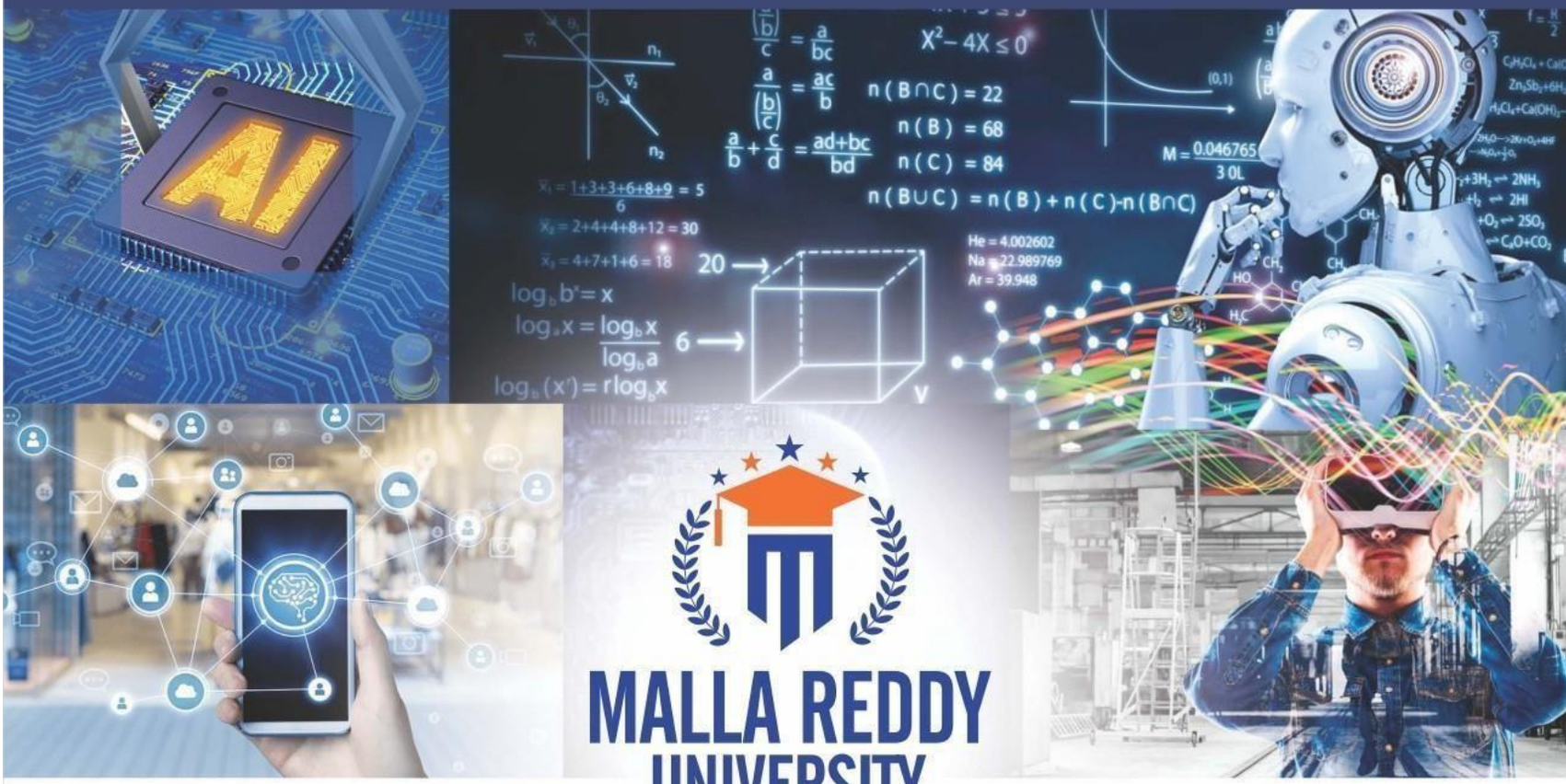
KNOWLEDGE EMPOWERMENT WORLD



MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Computer Aided Engineering Graphics & Workshop

Learning Manual

B.Tech : I Year

(2022-23)

School of Engineering

www.mallareddyuniversity.ac.in

Learning Manual

MALLA REDDY
UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and
G.O.Ms.No.14, Higher Education (UE) Department)

Name ....

Roll No. ..... Branch ....

Year .... Sem. .....

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and
G.O.Ms.No.14, Higher Education (UE) Department)

Maisammaguda, Kompally,
Medchal - Malkajgiri District
Hyderabad - 500100, Telangana State.
mruh@mallareddyuniversity.ac.in
www.mallareddyuniversity.ac.in

Certificate

School of Engineering

Certified that this is the bonafide record of practical work done by

Mr./Ms..... Roll. No..... of

B.Tech ..... year ..... Semester for Academic year 20..... - 20..... in

..... Laboratory.

Faculty Incharge HOD Dean-AIML

External Examiner

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:
   Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

4. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

5. Proper Dress code and Identity card.

6. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

7. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.

8. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

9. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.

10. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

11. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

12. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

COURSE OBJECTIVES:

- To understand the significance of big data in modern data-driven decision-making.
- To Identify and acquire data from various sources, including structured and unstructured data.
- To understand data storage solutions and databases designed for big data, such as Hadoop, NoSQL databases, and data lakes.
- Learn how to use **R** as a tool for statistical computing and data visualization.
- Implement clustering algorithms like **K-means** and **Hierarchical Clustering** in R to group large datasets.

COURSE OUTCOMES:

- Students will gain the ability to use big data and its operations.
- Students will be able to use classification of Analytics.
- Students will be able to Analyze the performance of the map reduce.
- Students will be able to use Hadoop, CQLSH, QHL technologies.
- Students will have Practical exposure on Hadoop.
- Learn how to use **R** as a tool for statistical computing and data visualization.
- Implement clustering algorithms like **K-means** and **Hierarchical Clustering** in R to group large datasets.

BIG DATA AND ANALYTICS

List of Contents

Week 1

Install, configure and run python, numpy and pandas.

AIM: To Installing and Running Applications On python, numpy and pandas.

How to Install Anaconda on Windows?

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called Aconda.

To begin working with Anaconda, one must get it installed first. Follow the below instructions to Download and install Anaconda on your system:

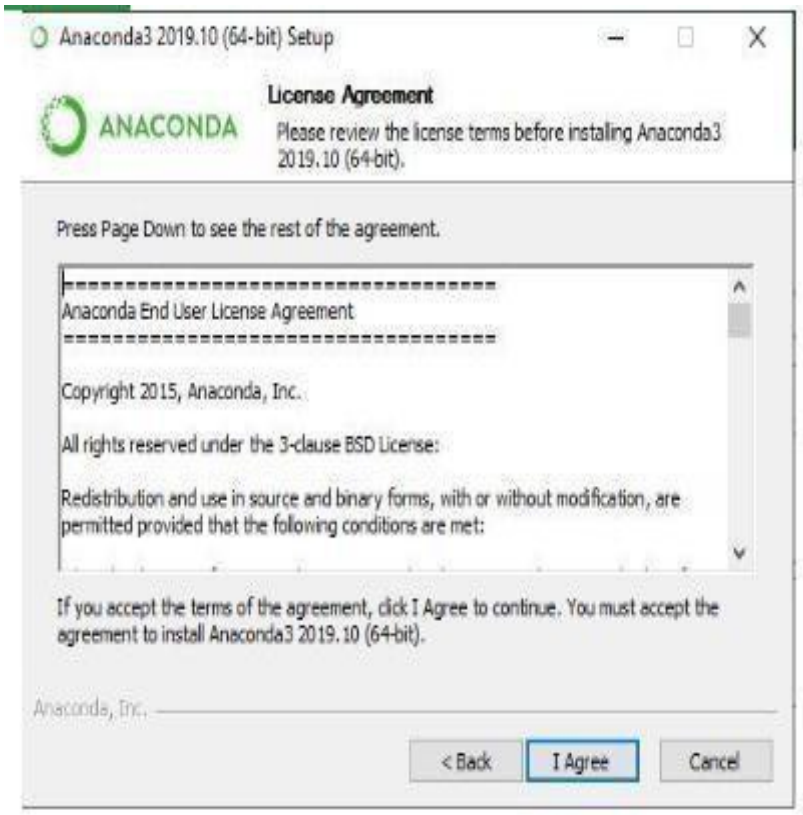
Download and install Anaconda:

Head over to anaconda.com and install the latest version of Anaconda. Make sure to download the ―Python 3.7 Version‖ for the appropriate architecture.

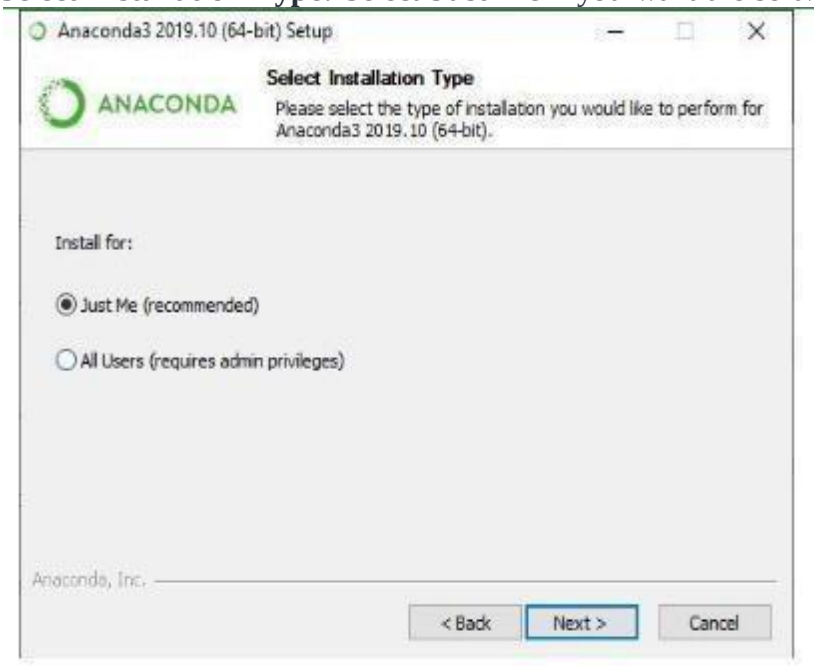
Begin with the installation process:

- **Getting Started**
- 

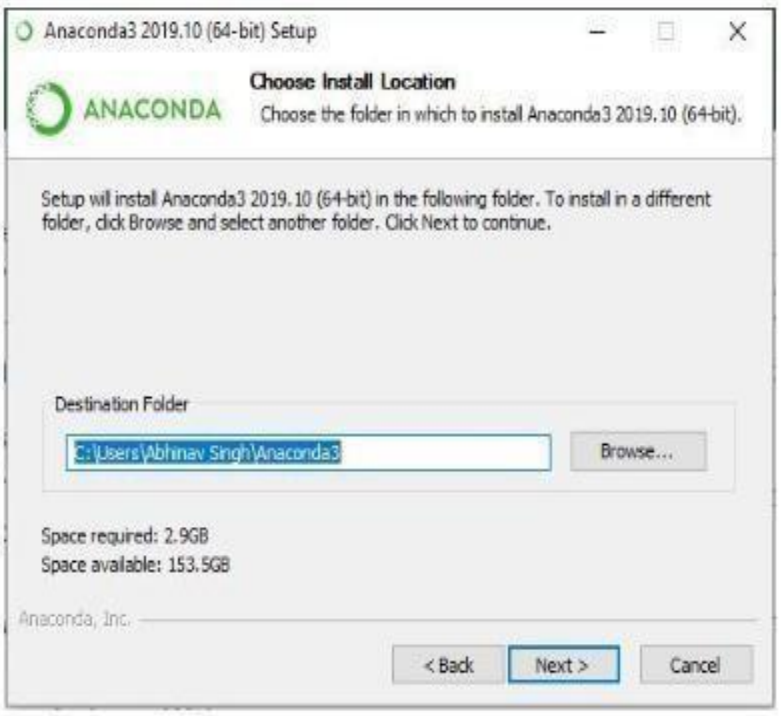
Getting through the License Agreement:



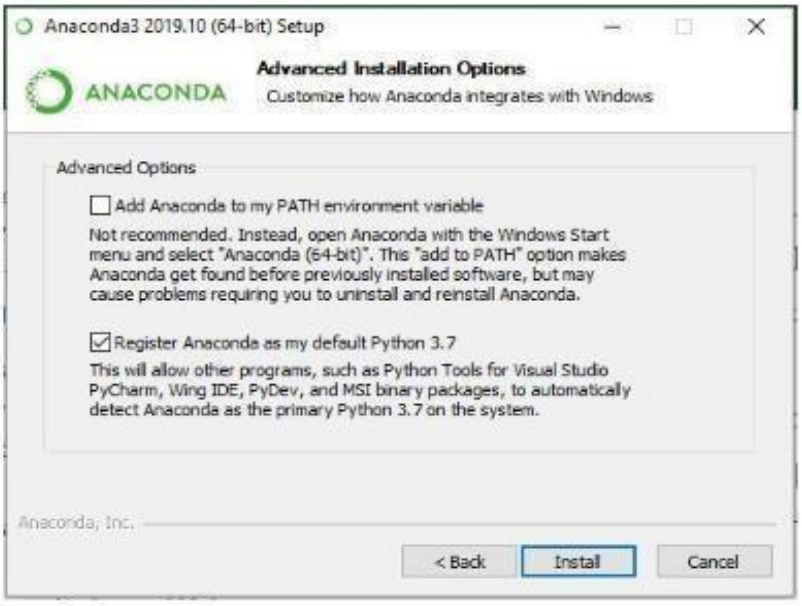
Select Installation Type: Select **Just Me** if you want the software to be used by a single User



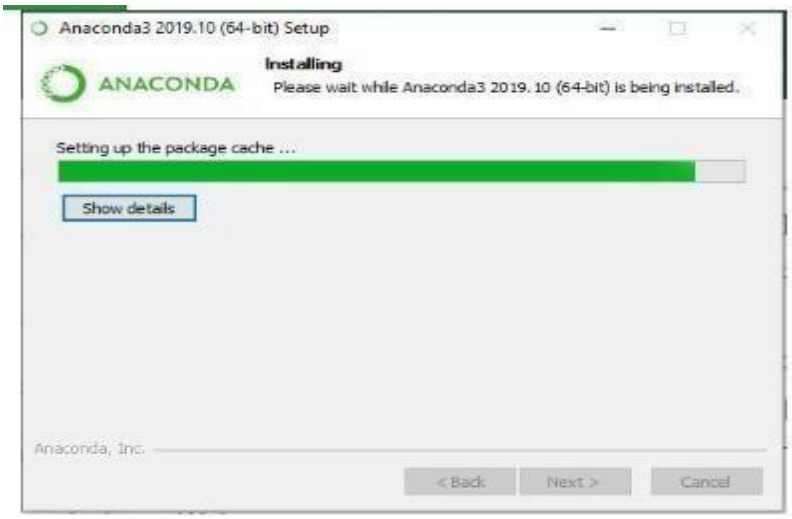
Choose Installation Location:



Advanced Installation Option:



Getting through the Installation Process:

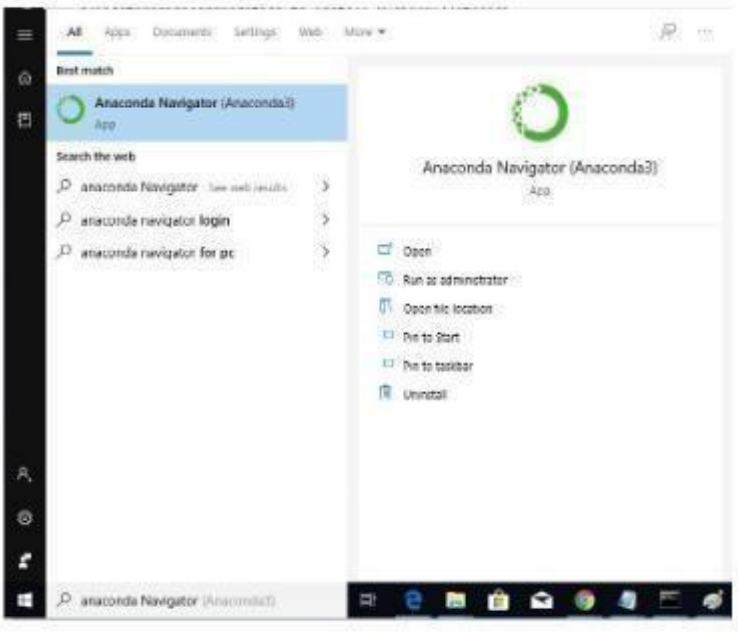


Recommendation to Install Pycharm:

Finishing up the Installation:

Working with Anaconda:

Once the installation process is done, Anaconda can be used to perform multiple operations. To begin using Anaconda, search for Anaconda Navigator from the Start Menu in Windows




```
#import pandas in jupyter notebook
import pandas

#loading the dataset which is excel
file dataset =
pandas.read_csv("crime.csv")

#displaying the data
dataset
```

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400
5	1970	18190740	1444	2875	81149	39145	267474	125674
6	1971	18391000	1823	3225	97682	42318	273704	127658
7	1972	18366000	2026	4199	86391	45926	239886	105081
8	1973	18265000	2040	4852	80795	47781	246246	112328
9	1974	18111000	1919	5240	86814	51454	271824	104095
10	1975	18120000	1996	5099	93499	54593	301996	116274
11	1976	18084000	1969	4663	95718	54638	318919	133504
12	1977	17924000	1919	5272	84703	57193	309735	133669
13	1978	17748000	1820	5168	83785	58484	292956	119264
14	1979	17649000	2092	5394	93471	60949	308302	124343
15	1980	17506690	2228	5405	112273	60329	360925	133041
16	1981	17594000	2166	5479	120344	60189	350422	136849
17	1982	17659000	2013	5159	107843	59818	295245	137880
18	1983	17667000	1958	5296	94783	59452	249115	127861
19	1984	17735000	1786	5599	89900	64872	222956	115392
20	1985	17783000	1683	5706	89706	68270	219633	106537
21	1986	17772000	1907	5415	91360	76528	217010	113247
22	1987	17825000	2016	5537	89721	82417	216826	125329
23	1988	17898000	2244	5479	97434	91239	218060	153898
24	1989	17950000	2246	5242	103983	91571	211130	171007
25	1990	17990455	2605	5368	112380	92105	208813	187591
26	1991	18058000	2571	5085	112342	90186	204499	181287
27	1992	18119000	2397	5152	108154	87608	193548	168922
28	1993	18197000	2420	5008	102122	85802	181709	151949
29	1994	18169000	2016	4700	86617	82100	164650	128873
30	1995	18136000	1550	4290	72492	74351	146562	102596
31	1996	18185000	1353	4174	61822	64857	129828	89900

```
import pandas as pd
dataset1 =
pd.read_csv("crime.csv")
dataset1
```

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400
5	1970	18190740	1444	2875	81149	39145	267474	125674
6	1971	18391000	1823	3225	97682	42318	273704	127658
7	1972	18366000	2026	4199	86391	45926	239886	105081
8	1973	18265000	2040	4852	80795	47781	246246	112328
9	1974	18111000	1919	5240	86814	51454	271824	104095
10	1975	18120000	1996	5099	93499	54593	301996	116274
11	1976	18084000	1969	4663	95718	54638	318919	133504
12	1977	17924000	1919	5272	84703	57193	309735	133669
13	1978	17748000	1820	5168	83785	58484	292956	119264
14	1979	17649000	2092	5394	93471	60949	308302	124343
15	1980	17506690	2228	5405	112273	60329	360925	133041
16	1981	17594000	2166	5479	120344	60189	350422	136849
17	1982	17659000	2013	5159	107843	59818	295245	137880
18	1983	17667000	1958	5296	94783	59452	249115	127861
19	1984	17735000	1786	5599	89900	64872	222956	115392
20	1985	17783000	1683	5706	89706	68270	219633	106537
21	1986	17772000	1907	5415	91360	76528	217010	113247
22	1987	17825000	2016	5537	89721	82417	216826	125329
23	1988	17898000	2244	5479	97434	91239	218060	153898
24	1989	17950000	2246	5242	103983	91571	211130	171007
25	1990	17990455	2605	5368	112380	92105	208813	187591
26	1991	18058000	2571	5085	112342	90186	204499	181287
27	1992	18119000	2397	5152	108154	87608	193548	168922
28	1993	18197000	2420	5008	102122	85802	181709	151949
29	1994	18169000	2016	4700	86617	82100	164650	128873
30	1995	18136000	1550	4290	72492	74351	146562	102596
31	1996	18185000	1353	4174	61822	64857	129828	89900

dataset1.head()

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400

dataset1.tail()

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
42	2007	19297729	801	2926	31094	45094	64857	28030
43	2008	19467789	836	2799	31789	42122	65537	25096
44	2009	19541453	781	2582	28141	43606	62769	21871
45	2010	19395206	868	2797	28630	44197	65839	20639
46	2011	19465197	774	2752	28396	45568	65397	19311

dataset1.head(10)

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400
5	1970	18190740	1444	2875	81149	39145	267474	125674
6	1971	18391000	1823	3225	97682	42318	273704	127658
7	1972	18366000	2026	4199	86391	45926	239886	105081
8	1973	18265000	2040	4852	80795	47781	246246	112328
9	1974	18111000	1919	5240	86814	51454	271824	104095

dataset1.tail(10)

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
37	2002	19134293	909	3885	36653	53583	76700	47366
38	2003	19212425	934	3775	35790	48987	75453	45204
39	2004	19280727	889	3608	33506	46911	70696	41002
40	2005	19315721	874	3636	35179	46150	68034	35736
41	2006	19306183	921	3169	34489	45387	68565	32134
42	2007	19297729	801	2926	31094	45094	64857	28030
43	2008	19467789	836	2799	31789	42122	65537	25096
44	2009	19541453	781	2582	28141	43606	62769	21871
45	2010	19395206	868	2797	28630	44197	65839	20639
46	2011	19465197	774	2752	28396	45568	65397	19311

type(dataset1)
pandas.core.frame.DataFrame

```
pandas.core.frame.DataFrame
```

#to find any null values in the last 5

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
42	False	False	False	False	False	False	False	False
43	False	False	False	False	False	False	False	False
44	False	False	False	False	False	False	False	False
45	False	False	False	False	False	False	False	False
46	False	False	False	False	False	False	False	False

rows dataset1.isnull().tail()
#to makesure that no null values exists
dataset1.notnull().tail()

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
42	True	True	True	True	True	True	True	True
43	True	True	True	True	True	True	True	True
44	True	True	True	True	True	True	True	True
45	True	True	True	True	True	True	True	True
46	True	True	True	True	True	True	True	True

#displays the number of null values in each columndataset1.isnull().sum()

```
Year          0
Population    0
Murder        0
Rape          0
Robbery       0
Assault       0
Burglary      0
CarTheft      0
dtype: int64
```

#helps to find null values with respect to ROBBERY column
dataset1[dataset1.Robbery.isnull()]

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft

dataset1.shape

```
(47, 8)
```

#helps to find how many times values in a particular column has repeated
dataset1['Robbery'].value_counts()

```
94783     1
34489     1
86814     1
86617     1
80795     1
97434     1
108154    1
120344    1
56094     1
28182     1
31094     1
30098     1
91360     1
59857     1
35790     1
36555     1
40202     1
83785     1
```

#consolidated value counts for all the columns in the
datasetfor col in dataset1.columns:
    display(dataset1[col].value_counts())

```
1983    1
1995    1
2004    1
2003    1
2002    1
2001    1
2000    1
1999    1
1998    1
1997    1
1996    1
1994    1
2006    1
1993    1
1992    1
1991    1
1990    1
1989    1
1988    1
```

#helps to find number of rows in the dataset
dataset_length=len(dataset1)
dataset_length

```
47
```

#helps to find number of columns in the dataset
dataset_col=len(dataset1.columns)
dataset_col

```
8
```

#helps to find the summary of numerical columns
dataset1.describe()

	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
count	47.000000	4.700000e+01	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000
mean	1988.000000	1.834426e+07	1549.978723	4200.425532	70429.297872	58022.234043	189119.829787	97573.553191
std	13.711309	6.024504e+05	590.454265	1096.569507	30204.823764	17455.534367	90256.257143	46707.064488
min	1965.000000	1.750669e+07	774.000000	2320.000000	28141.000000	27464.000000	62769.000000	19311.000000
25%	1976.500000	1.793700e+07	922.500000	3197.000000	36604.000000	45477.500000	90581.500000	56246.000000
50%	1988.000000	1.816900e+07	1683.000000	4199.000000	81149.000000	57193.000000	208813.000000	106537.000000
75%	1999.500000	1.868373e+07	2016.000000	5241.000000	94141.000000	64864.500000	250016.500000	128367.000000
max	2011.000000	1.954145e+07	2605.000000	5706.000000	120344.000000	92105.000000	360925.000000	187591.000000

#helps to describe individual column
dataset1.Murder.describe()

```
count      47.000000
mean     1549.978723
std       590.454265
min       774.000000
25%       922.500000
50%      1683.000000
75%      2016.000000
max      2605.000000
Name: Murder, dtype: float64
```

dataset1.skew()

```
Year          0.000000
Population    0.795669
Murder        0.059733
Rape         -0.237130
Robbery      -0.134085
Assault       0.464637
Burglary     -0.020278
CarTheft     -0.129653
dtype: float64
```

```
Year          1.880000e+02
Population    3.629465e+11
Murder        3.486362e+05
Rape          1.202465e+06
Robbery       9.123314e+08
Assault       3.046957e+08
Burglary      8.146192e+09
CarTheft      2.181550e+09
dtype: float64
```

dataset1.var()

dataset1.kurtosis()

```
Year         -1.200000
Population   -0.692220
Murder       -1.513564
Rape         -1.471445
Robbery      -1.527674
Assault      -0.482013
Burglary     -1.186281
CarTheft     -0.951036
dtype: float64
```

```
Year          int64
Population    int64
Murder        int64
Rape          int64
Robbery       int64
Assault       int64
Burglary      int64
CarTheft      int64
dtype: object
```

print(dataset1.dtypes)

NUMPY

Numpy is the core library for scientific and numerical computing in Python. It provides highperformance multi-dimensional array object and tools for working with arrays.
Numpy main object is the multidimensional array, it is a table of elements (usually numbers) all ofthe same type indexed by a positive integers.

In Numpy dimensions are called as axes.
Numpy is fast, convenient and occupies less memory when compared to python list.

```
import numpy
arr = numpy.array([1, 2, 3, 4, 5])
print(arr)
```

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

NumPy is usually imported under the np alias.

```
import numpy as np
```

Now the NumPy package can be referred to as np instead of numpy.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

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

Checking NumPy Version

The version string is stored under __version___attribute.

```
import numpy as np

print(np. version )
```

```
1.18.1
```

Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called ndarray.We can create a NumPy ndarray object by using the array() function.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)

print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

type(): This built-in Python function tells us the type of the object passed to it. Like in abovecode it shows that arr is numpy.ndarray type.

To create an ndarray, we can pass a list, tuple or any array-like object into the array() method, and itwill be converted into an ndarray:

Use a tuple to create a NumPy array:

```
import numpy as np
arr = np.array((1, 2, 3, 4, 5))
```

```
print(arr)
  [1 2 3 4 5]
```

Dimensions in Arrays

A dimension in arrays is one level of array depth (nested arrays).

nested array: are arrays that have arrays as their elements.

0-D Arrays
0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array.

#Create a 0-D array with value
42 import numpy as np
arr = np.array(42)
print(arr)

```
42
```

1-D Arrays
These are the most common and basic arrays.

#Create a 1-D array containing the values 1,2,3,4,5:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])

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

print(arr)

2-D Arrays
An array that has 1-D arrays as its elements is called a 2-D array.These are often used to represent matrix or 2nd order tensors.

#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)

```
[[1 2 3]
 [4 5 6]]
```

3-D arrays
An array that has 2-D arrays (matrices) as its elements is called 3-D array.These are often used to represent a 3rd order tensor.

#Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6:import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr)

```
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
```

Check Number of Dimensions?
NumPy Arrays provides the ndim attribute that returns an integer that tells us how many dimensionsthe array have.

```
#Check how many dimensions the arrays have:
import numpy as npa
= np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(a.ndim
)
print(b.ndi
```

```
0
1
2
3
```

```
m)
print(c.ndim
)
print(d.ndi
m)
```

```
#Create an array with 5 dimensions and verify that it has 5
dimensions:import numpy as np
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
```

```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

```
print('number of dimensions :', arr.ndim)
```

NumPy Array
IndexingAccess Array
Elements

Array indexing is the same as accessing an array element.
You can access an array element by referring to its index number.
The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the secondhas index 1 etc.

```
#Get the first element from the following array:
import numpy as np
arr = np.array([1, 2, 3,
4]) print(arr[0])
```

```
#Get the second element from the following array.
import numpy as np
arr = np.array([1, 2, 3,
4]) print(arr[1])
```

```
#Get third and fourth elements from the following array and add
them.import numpy as np
arr = np.array([1, 2, 3,
```

4]) print(arr[2] + arr[3])

```
1
2
7
```

Access 2-D Arrays

To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.
Think of 2-D arrays like a table with rows and columns, where the dimension represents the row and the index represents the column.

#Access the element on the first row, second column:

```
import numpy as np
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
```

```
2nd element on 1st row:  2
```

#Access the element on the 2nd row, 5th column:

```
import numpy as np
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('5th element on 2nd row: ', arr[1, 4])
```

```
5th element on 2nd row:  10
```

OUTPUT:

Record Notes

WEEK: 2

Install, Configure and Run Hadoop and HDFS

PROGRAM:

AIM: To Installing and Running Applications On Hadoop and HDFS.

HADOOP INSTALATION IN WINDOWS

1. Prerequisites

Hardware Requirement

* RAM — Min. 8GB, if you have SSD in your system then 4GB RAM would also work.
* CPU — Min. Quad core, with at least 1.80GHz

2. JRE 1.8 — Offline installer for JRE

3. Java Development Kit — 1.8

4. A Software for Un-Zipping like 7Zip or Win Rar

* I will be using a 64-bit windows for the process, please check and download the version supported by your system x86 or x64 for all the software.

5. Download Hadoop zip

* I am using Hadoop-2.9.2, you can use any other STABLE version for hadoop.

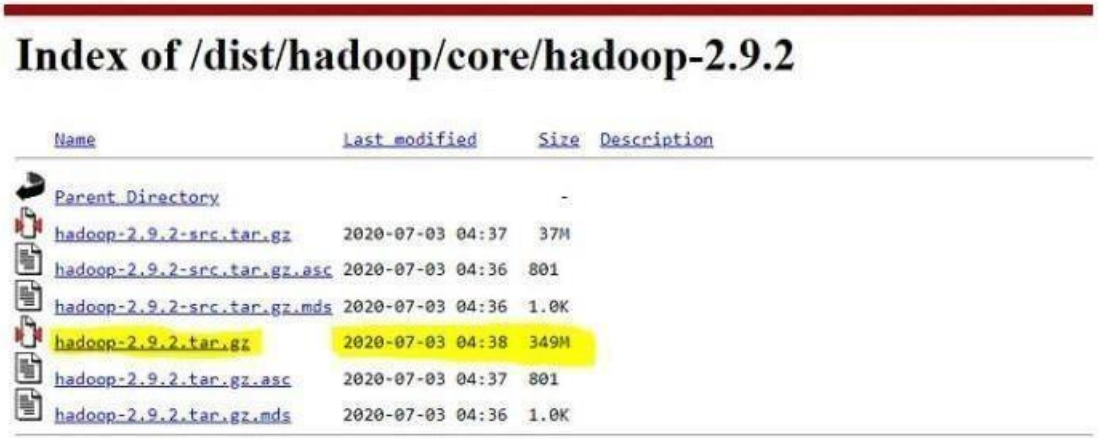


Fig. 1:- Download Hadoop 2.9.2

Once we have Downloaded all the above software, we can proceed with next steps in installing theHadoop.

2. Unzip and Install Hadoop

After Downloading the Hadoop, we need to Unzip the hadoop-2.9.2.tar.gz file.

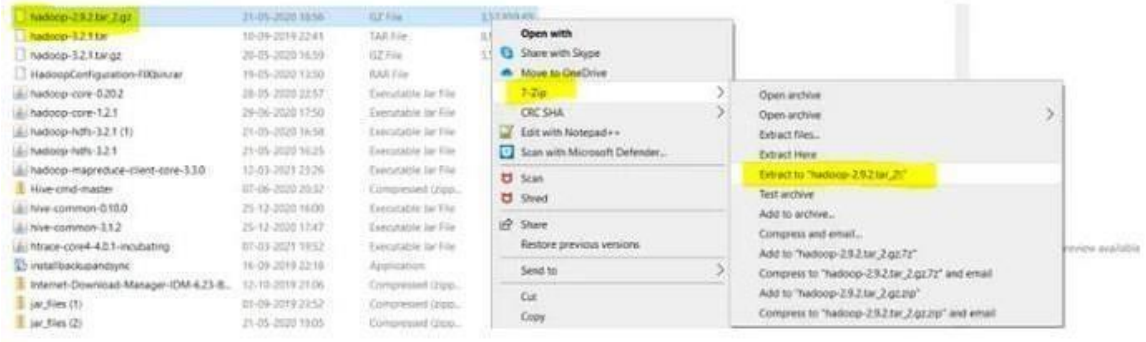


Fig. 2:- Extracting Hadoop Step-1

Once extracted, we would get a new file hadoop-2.9.2.tar.

Now, once again we need to extract this tar file.

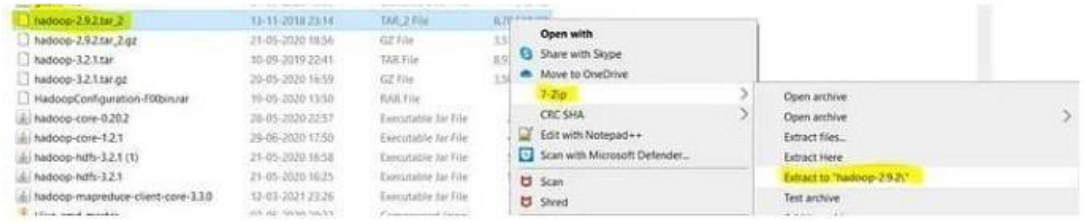


Fig. 3:- Extracting Hadoop Step-2

Now we can organize our Hadoop installation, we can create a folder and move the final extractedfile in it. For Eg. :-

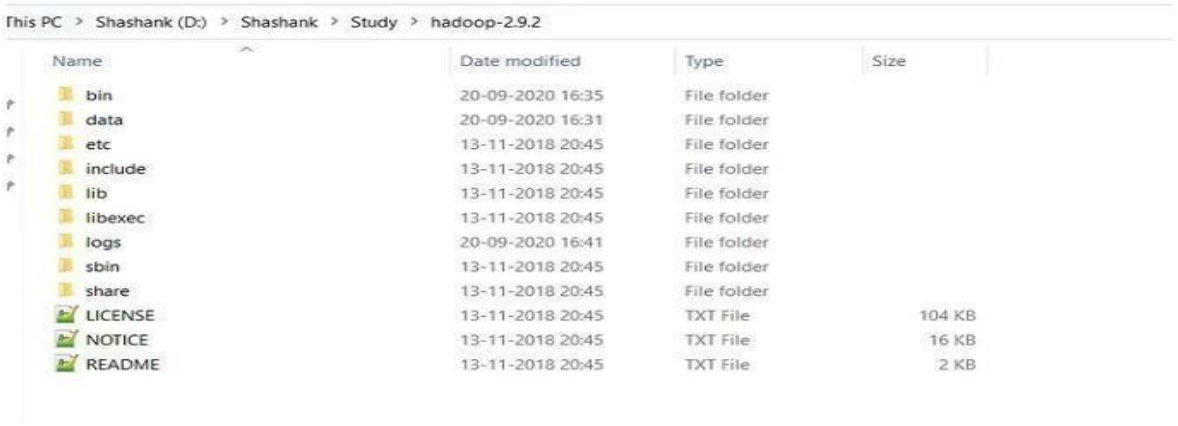


Fig. 4:- Hadoop Directory

Please note while creating folders, DO NOT ADD SPACES IN BETWEEN THE FOLDERNAME.(it can cause issues later)
I have placed my Hadoop in D: drive you can use C: or any other drive also.

3. Setting Up Environment Variables

Another important step in setting up a work environment is to set your Systems environmentvariable. To edit environment variables, go to Control Panel > System > click on the ―Advanced system settings‖ link Alternatively, We can Right click on This PC icon and click on Properties and click on the ―Advanced system settings‖ link
Or, easiest way is to search for Environment Variable in search bar and there you GO…

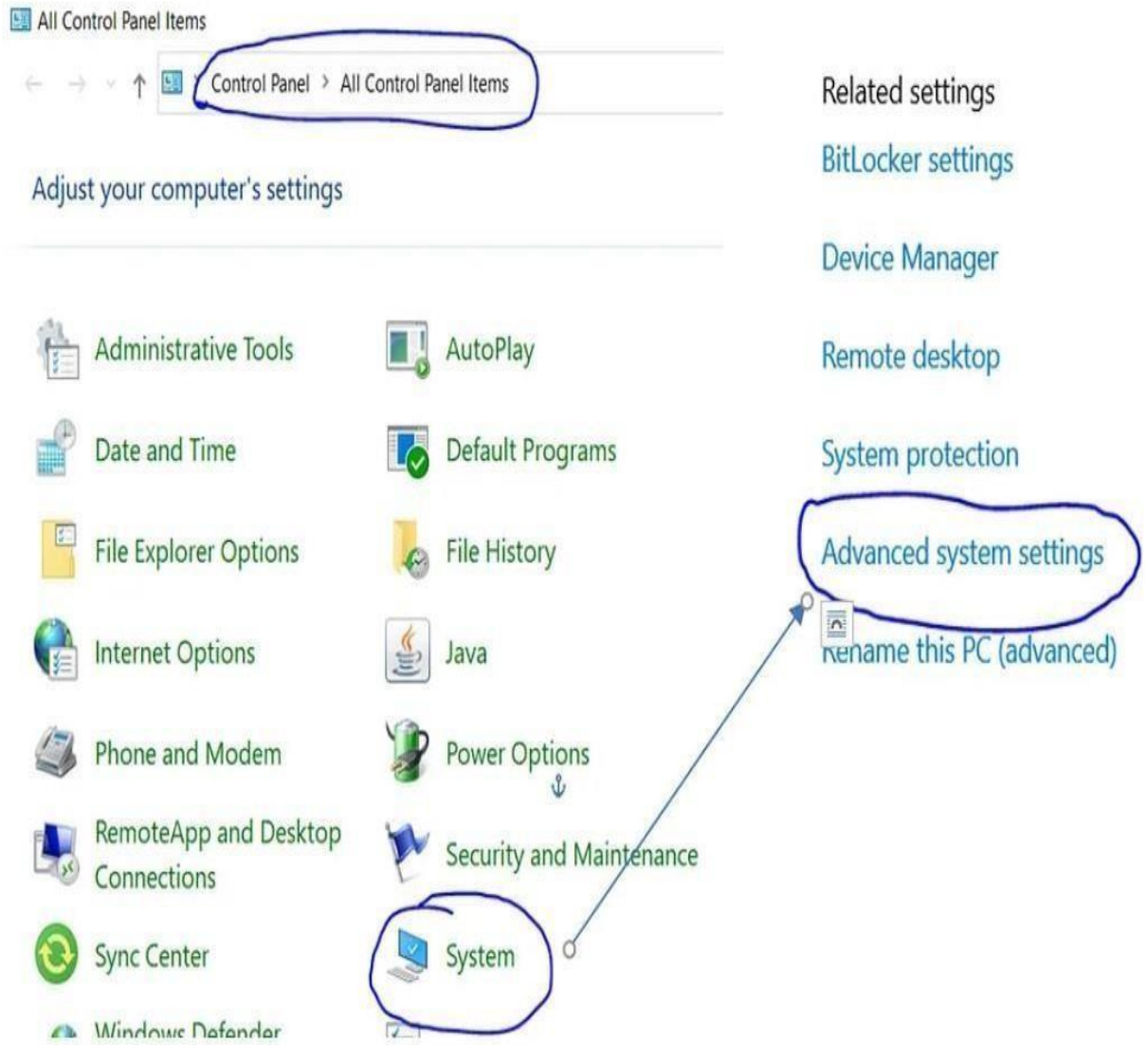


Fig. 5:- Path for Environment Variable

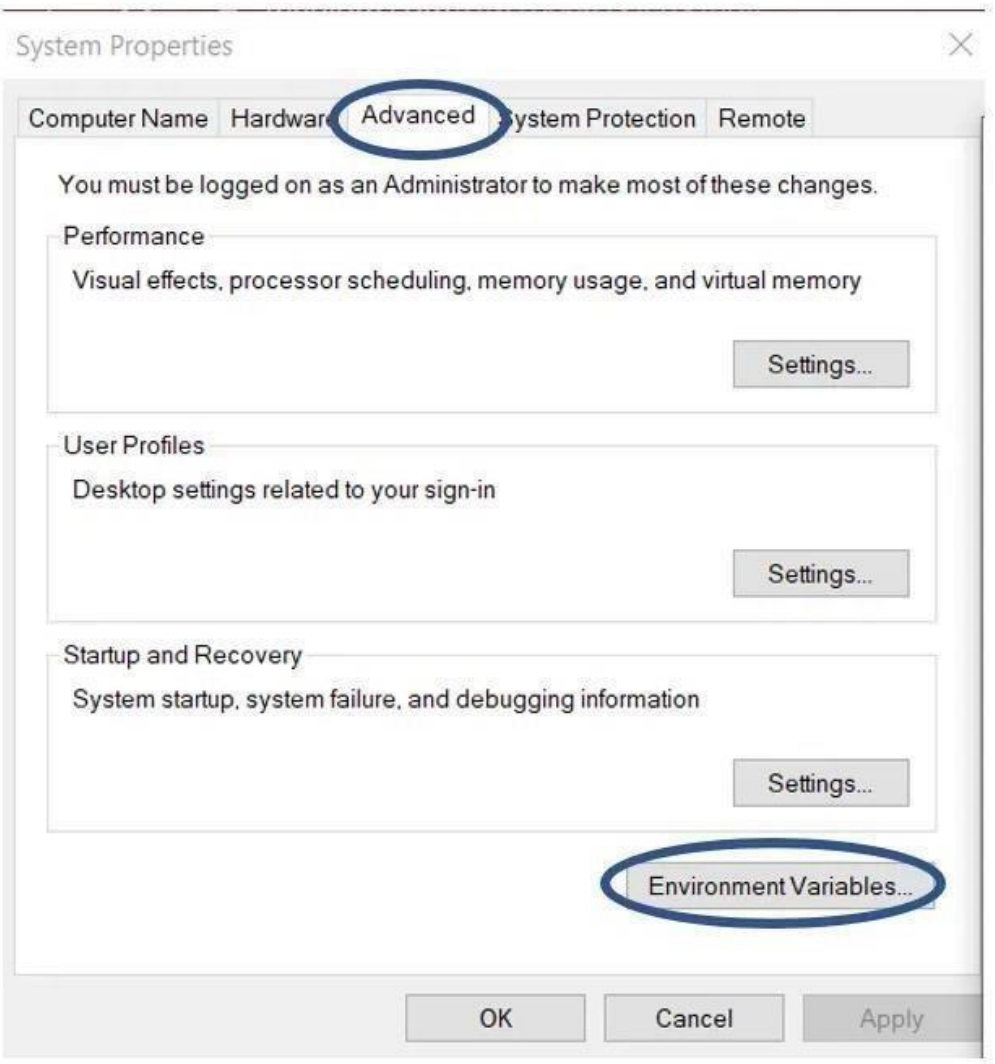


Fig. 6:- Advanced System Settings Screen

3.1 Setting JAVA_HOME

Open environment Variable and click on ―New‖ in ―User Variable‖

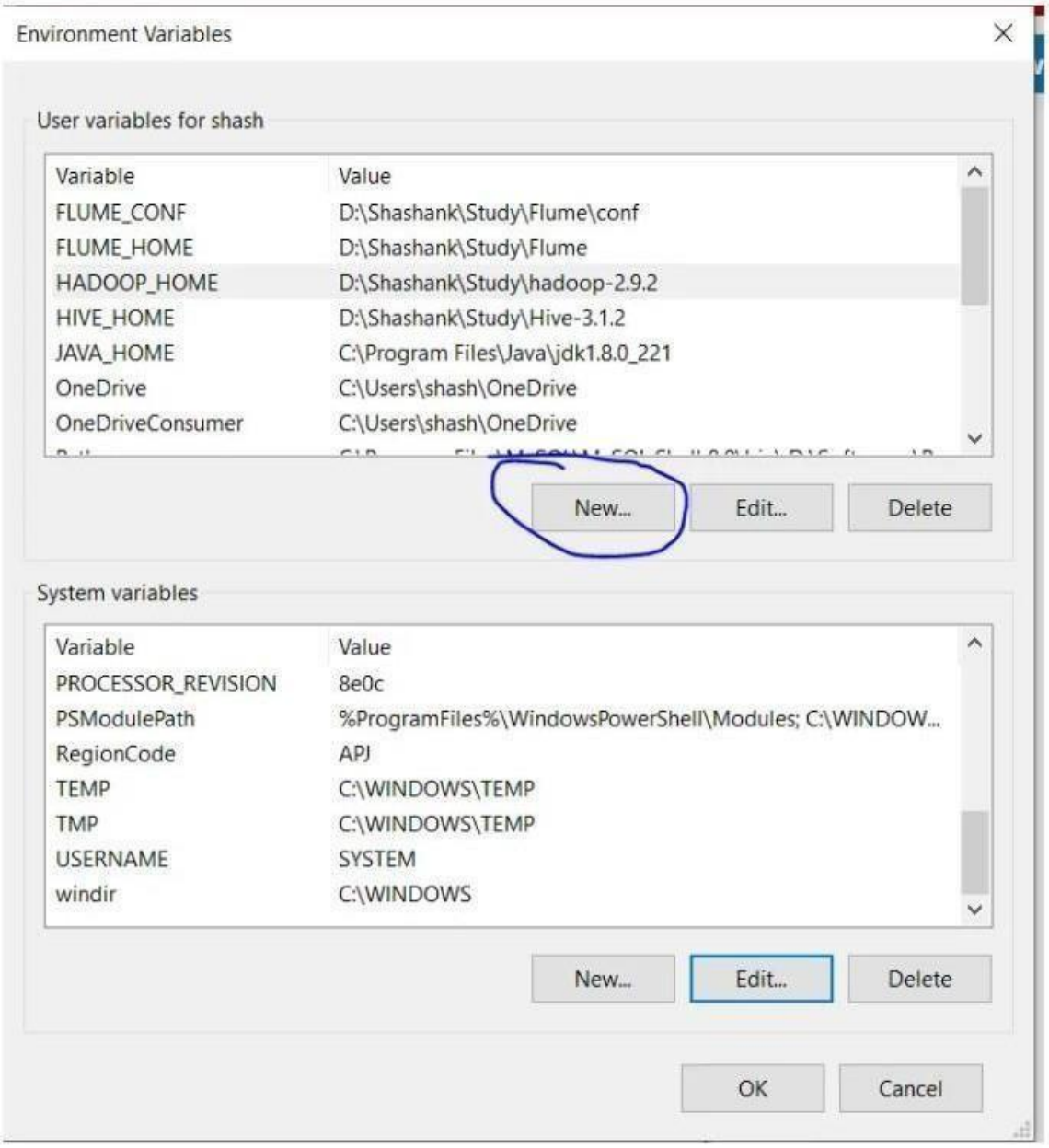


Fig. 7:- Adding Environment Variable

On clicking ―New‖, we get below screen.

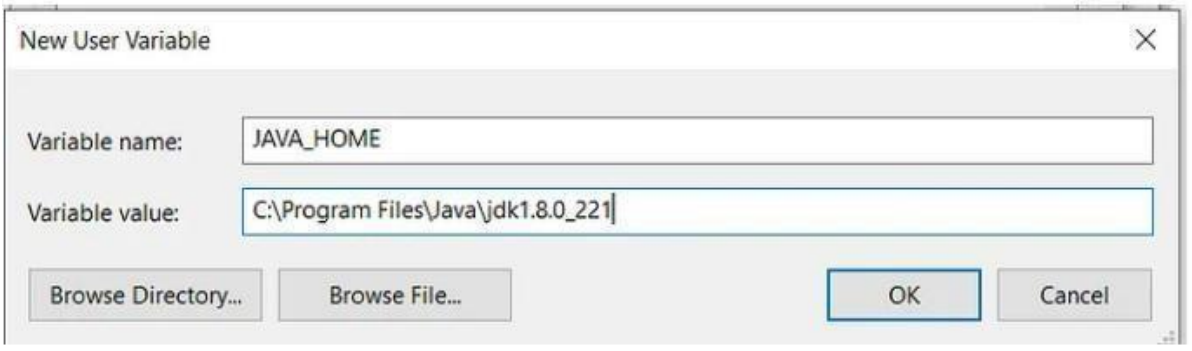


Fig. 8:- Adding JAVA_HOME

Now as shown, add JAVA_HOME in variable name and path of Java(jdk) in Variable Value.Click OK and we are half done with setting JAVA_HOME.

3.2 Setting HADOOP_HOME

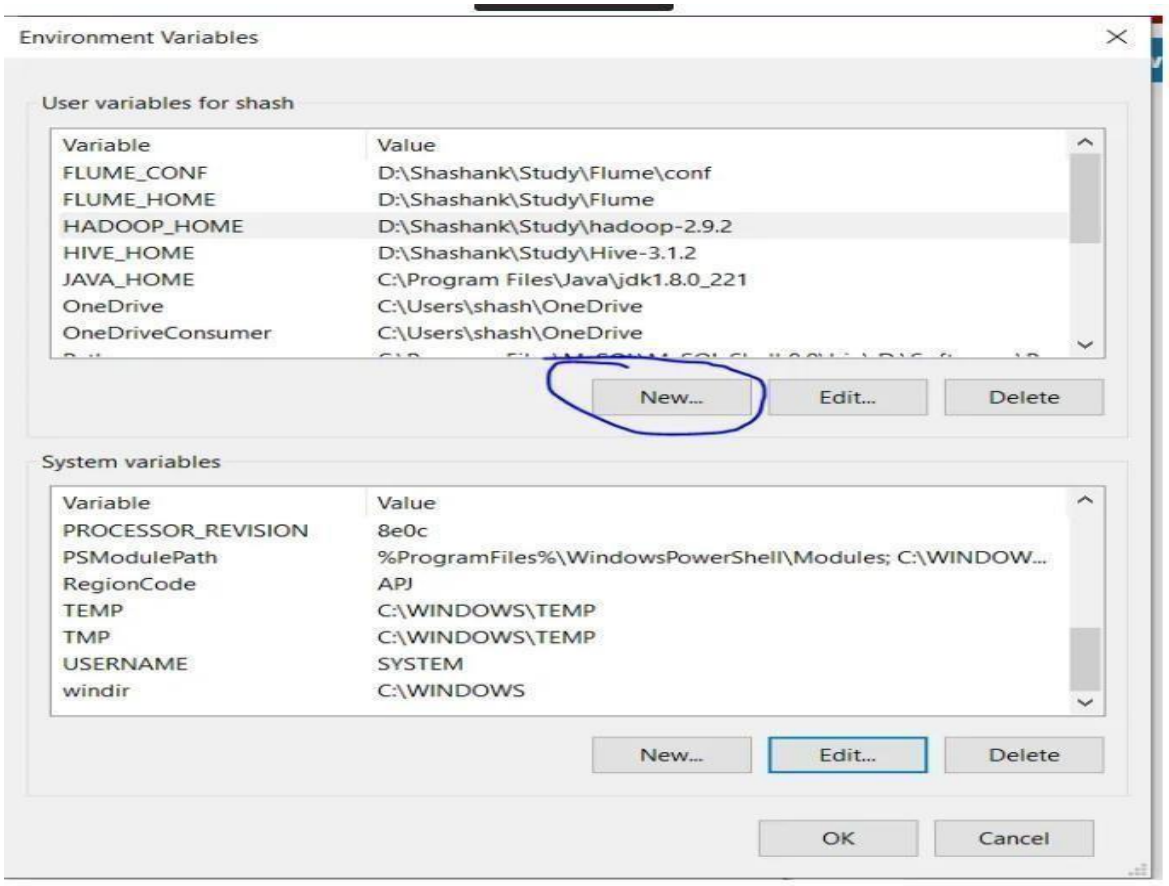


Fig. 9:- Adding Environment Variable

Open environment Variable and click on ―New‖ in ―User Variable‖
On clicking ―New‖, we get below screen.

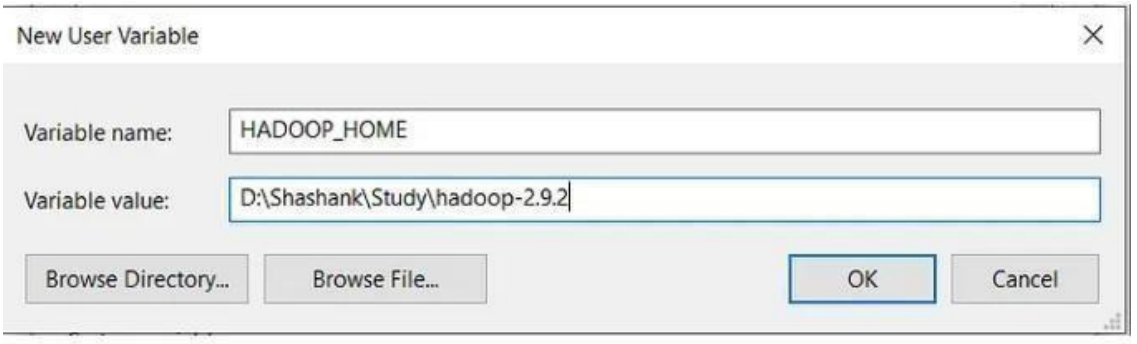


Fig. 10:- Adding HADOOP_HOME

Now as shown, add HADOOP_HOME in variable name and path of Hadoop folder in VariableValue. Click OK and we are half done with setting HADOOP_HOME.
Note:- If you want the path to be set for all users you need to select ―New‖ from System Variables.

3.3 Setting Path Variable

Last step in setting Environment variable is setting Path in System Variable.

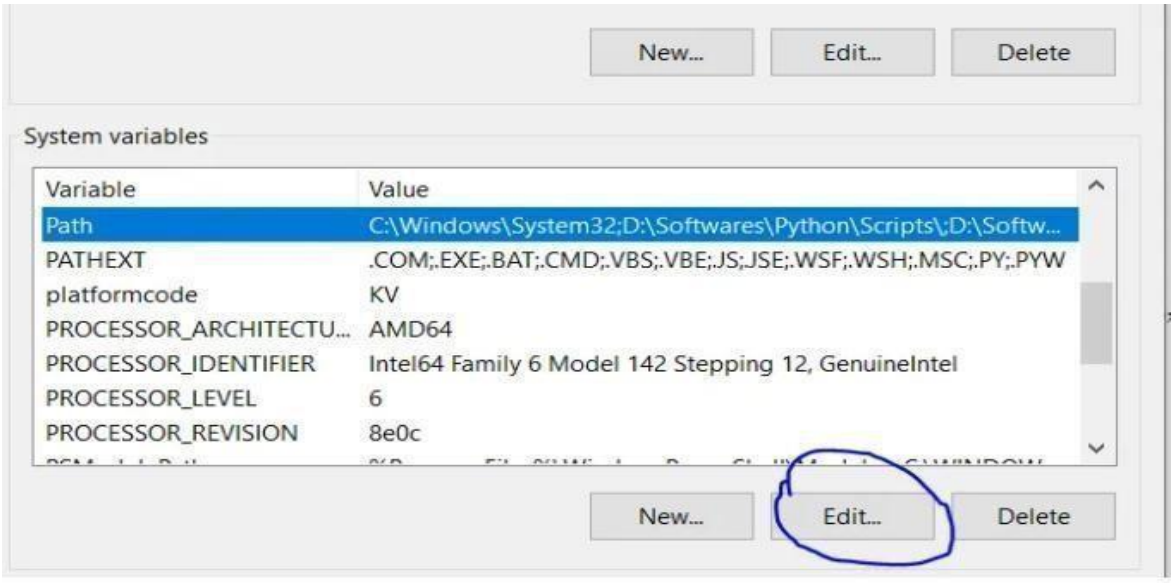


Fig. 11:- Setting Path Variable

Select Path variable in the system variables and click on ―Edit‖.

Fig. 12:- Adding Path

Now we need to add these paths to Path Variable one by one:-
* %JAVA_HOME%\bin
* %HADOOP_HOME%\bin
* %HADOOP_HOME%\sbin
Click OK and OK. & we are done with Setting Environment Variables.

3.4 Verify the Paths

Now we need to verify that what we have done is correct and reflecting.Open a NEW Command Window

Run following commands

echo %JAVA_HOME%
echo %HADOOP_HOME%
echo %PATH%

4. Editing Hadoop files

Once we have configured the environment variables next step is to configure Hadoop. It has 3 parts:-

4.1 Creating Folders

We need to create a folder data in the hadoop directory, and 2 sub folders namenode and datanode

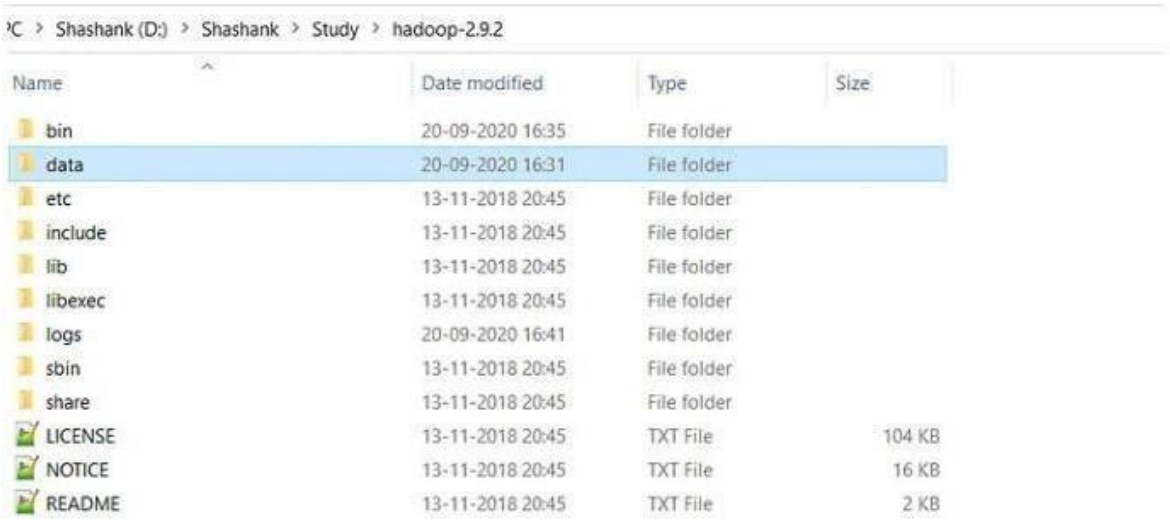


Fig. 13:- Creating Data Folder

Create DATA folder in the Hadoop directory

Fig. 14:- Creating Sub-folders

Once DATA folder is created, we need to create 2 new folders namely, namenode and datanodeinside the data folder
These folders are important because files on HDFS resides inside the datanode.

4.2 Editing Configuration Files

Now we need to edit the following config files in hadoop for configuring it
:-(We can find these files in Hadoop -> etc -> hadoop)
* core-site.xml
* hdfs-site.xml
* mapred-site.xml
* yarn-site.xml
* hadoop-env.cmd

4.2.1 Editing core-site.xml

Right click on the file, select edit and paste the following content within <configuration> </configuration> tags.
Note:- Below part already has the configuration tag, we need to copy only the part inside it.

```
<configuration>
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
```

```
</property>
</configuration>
```

4.2.2 Editing hdfs-site.xml

Right click on the file, select edit and paste the following content within <configuration></configuration>tags.

Note:- Below part already has the configuration tag, we need to copy only the part inside it.

Also replace PATH~1 and PATH~2 with the path of namenode and datanode folder that we created recently(step 4.1).

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>C:\hadoop\data\namenode</value>
   </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>C:\hadoop\data\datanode</value>
 </property>
</configuration>
```

4.2.3 Editing mapred-site.xml

Right click on the file, select edit and paste the following content within <configuration> </configuration> tags.

Note:- Below part already has the configuration tag, we need to copy only the part inside it.

```
<configuration>
 <property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
 </property>
</configuration>
```

4.2.4 Editing yarn-site.xml

Right click on the file, select edit and paste the following content within <configuration> </configuration> tags.

Note:- Below part already has the configuration tag, we need to copy only the part inside it.

```
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
</configuration>
```

4.2.5 Verifying hadoop-env.cmd

Right click on the file, select edit and check if the JAVA_HOME is set correctly or not.

We can replace the JAVA_HOME variable in the file with your actual JAVA_HOME that we configured in the System Variable.

```
set JAVA_HOME=%JAVA_HOME%OR
set JAVA_HOME="C:\Program Files\Java\jdk1.8.0_221"
```

4.3 Replacing bin

Last step in configuring the hadoop is to download and replace the bin folder.

* Go to this GitHub Repo and download the bin folder as a zip.
* Extract the zip and copy all the files present under bin folder to %HADOOP_HOME%\bin
Note:- If you are using different version of Hadoop then please search for its respective bin folder and download it.

5. Testing Setup

Congratulation..!!!!!
We are done with the setting up the Hadoop in our System.Now we need to check if everything works smoothly…

5.1 Formatting Namenode

Before starting hadoop we need to format the namenode for this we need to start a NEW CommandPrompt and run below command
hadoop namenode –format

Fig. 15:- Formatting Namenode

Note:- This command formats all the data in namenode. So, its advisable to use only at the start anddo not use it every time while starting hadoop cluster to avoid data loss.

5.2 Launching Hadoop

Now we need to start a new Command Prompt remember to run it as administrator to avoidpermission issues and execute below commands
start-all.cmd

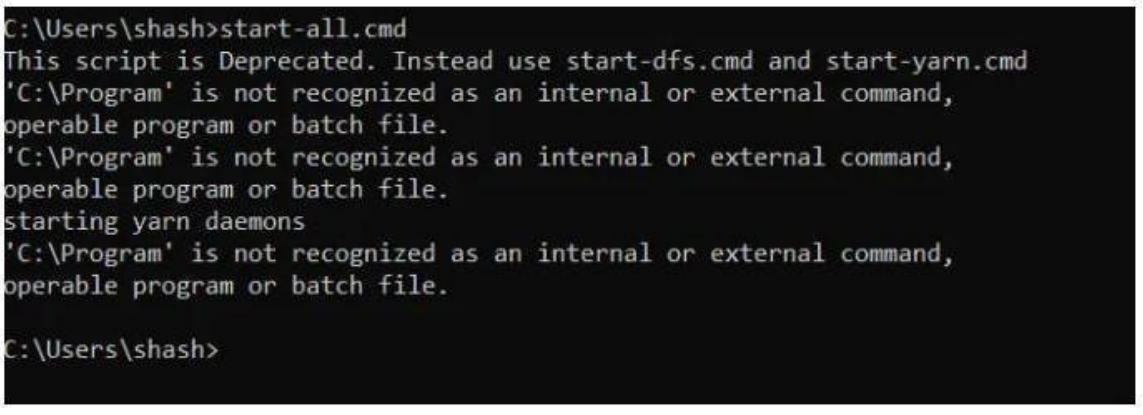


Fig. 16:- start-all.cmd

This will open 4 new cmd windows running 4 different Daemons of hadoop:-
* Namenode
* Datanode
* Resourcemanager
* Nodemanager

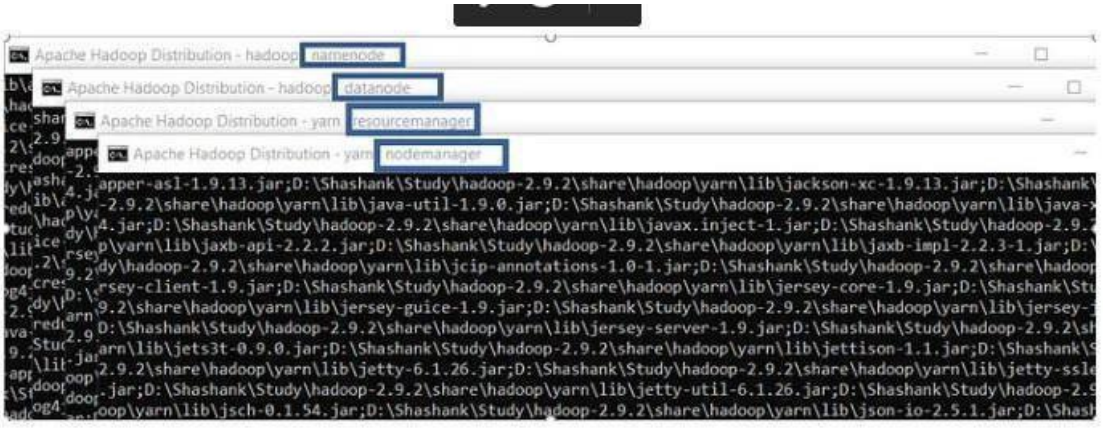


Fig. 17:- Hadoop Deamons

Note:- We can verify if all the daemons are up and running using jps command in new cmd window.

6. Running Hadoop (Verifying Web UIs)

6.1 Namenode

Open localhost:50070 in a browser tab to verify namenode health.

Fig. 18:- Namenode Web UI

6.2 Resourcemanger

Open localhost:8088 in a browser tab to check resourcemanager details.

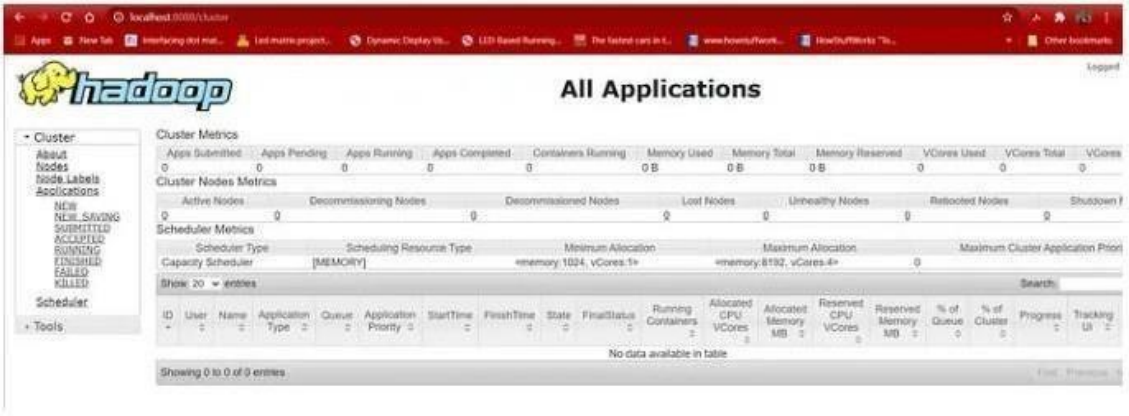


Fig. 19:- Resourcemanager Web UI

6.3 Datanode

Open localhost:50075 in a browser tab to checkout datanode.

Fig. 20:- Datanode Web UI

OUTPUT:

Record Notes

WEEK: 3

Visualize Data Using Basic Plotting Techniques In Python.

PROGRAM:

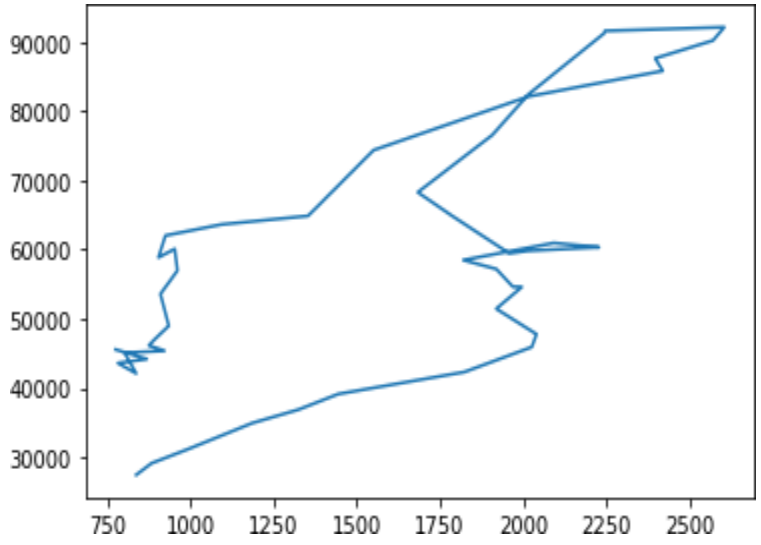
AIM: To create an application that takes the Visualize Data Using Basic Plotting Techniques.

Import pandas as pb
import matplotlib.pyplot as plt import
seaborn as sns

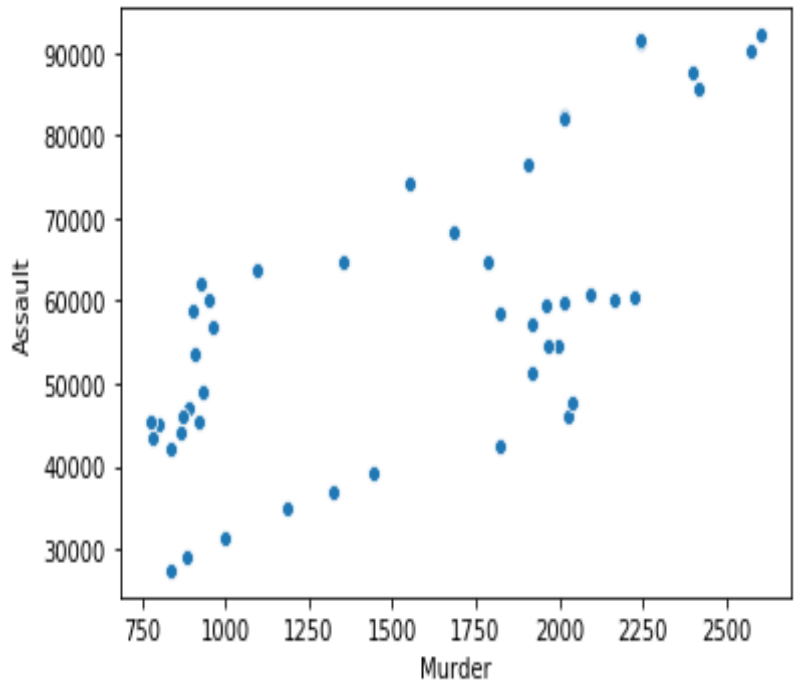
	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890	248477	115400
5	1970	18190740	1444	2875	81149	39145	267474	125674
6	1971	18391000	1823	3225	97682	42318	273704	127658
7	1972	18366000	2026	4199	86391	45926	239886	105081
8	1973	18265000	2040	4852	80795	47781	246246	112328
9	1974	18111000	1919	5240	86814	51454	271824	104095
10	1975	18120000	1996	5099	93499	54593	301996	116274
11	1976	18084000	1969	4663	95718	54638	318919	133504
12	1977	17924000	1919	5272	84703	57193	309735	133669
13	1978	17748000	1820	5168	83785	58484	292956	119264
14	1979	17649000	2092	5394	93471	60949	308302	124343
15	1980	17506690	2228	5405	112273	60329	360925	133041
16	1981	17594000	2166	5479	120344	60189	350422	136849
17	1982	17659000	2013	5159	107843	59818	295245	137880
18	1983	17667000	1958	5296	94783	59452	249115	127861
19	1984	17735000	1786	5599	89900	64872	222956	115392
20	1985	17783000	1683	5706	89706	68270	219633	106537

crime=pb.read_csv('crime.csv')crime

plt.plot(crime.Murder,crime.Assault);

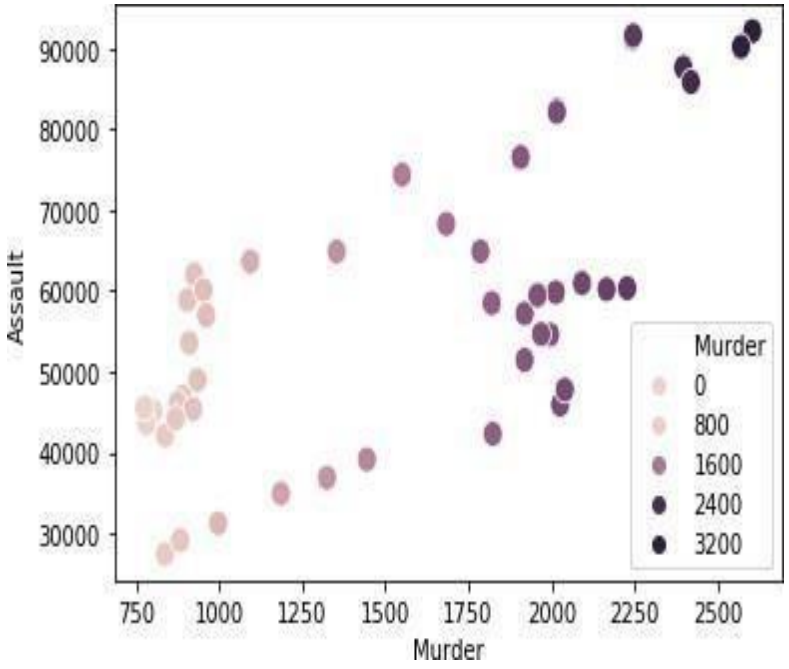


import seaborn as sns

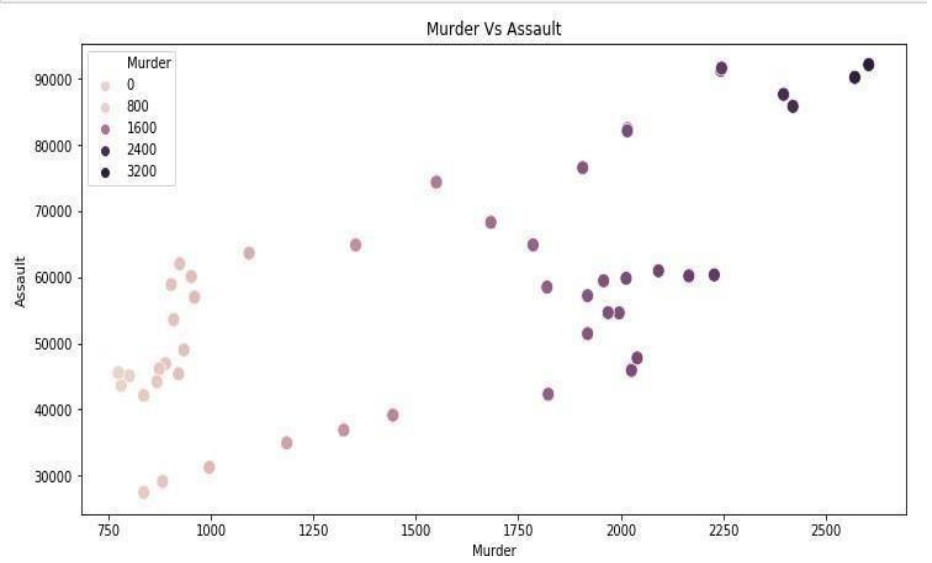



```
sns.scatterplot(crime.Murder,crime.Assault);
```

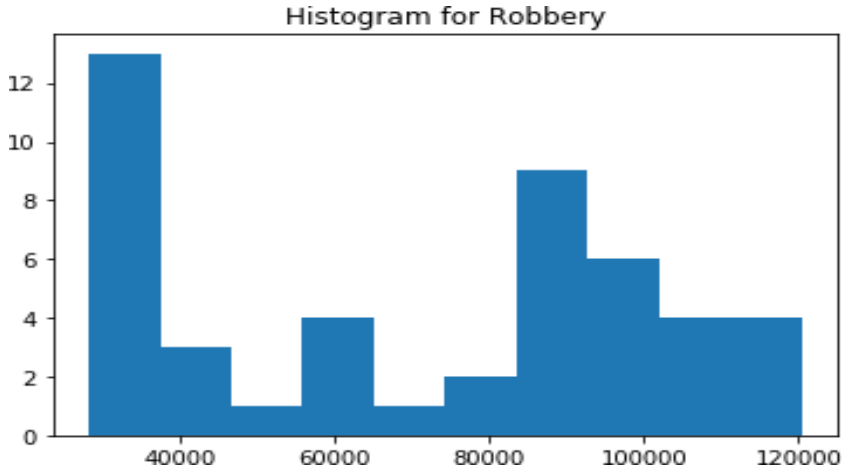
```
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



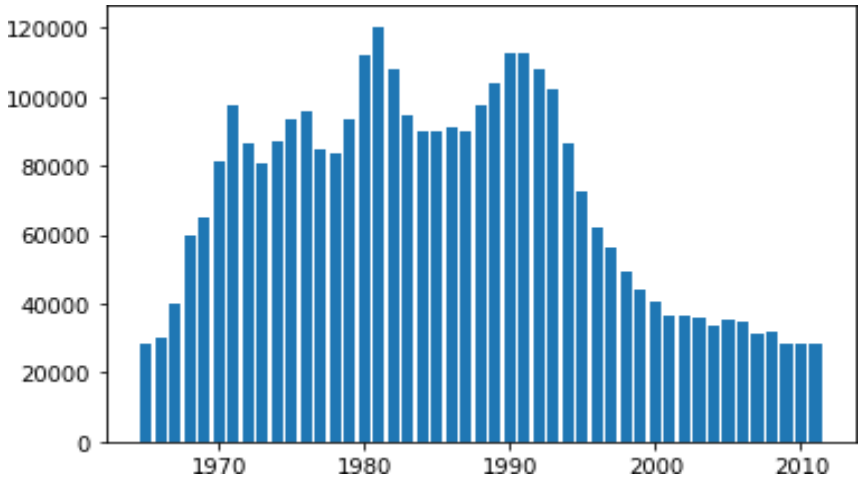

```
plt.figure(figsize=(12,6))
plt.title('Murder Vs Assault')
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



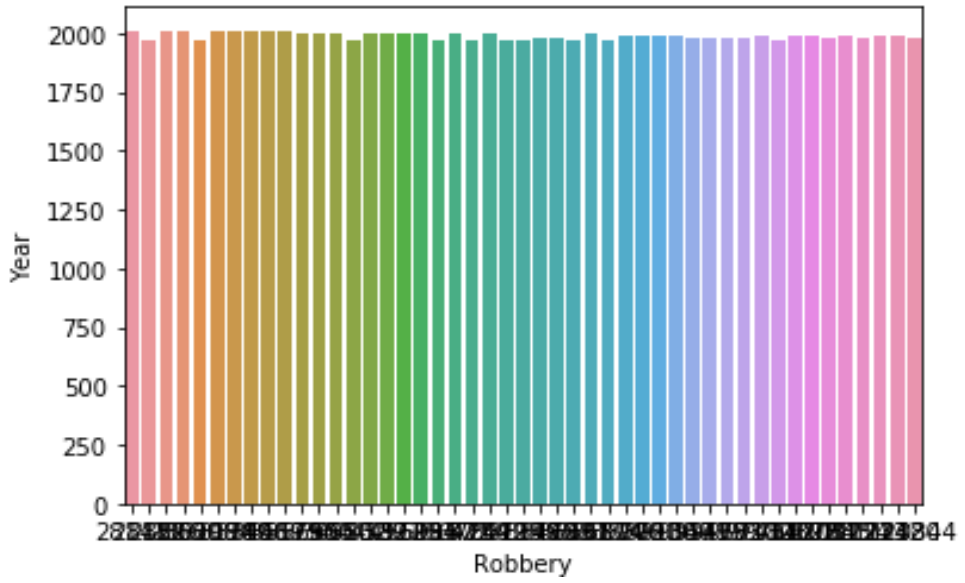
plt.title('Histogram for Robbery')
plt.hist(crime.Robbery);



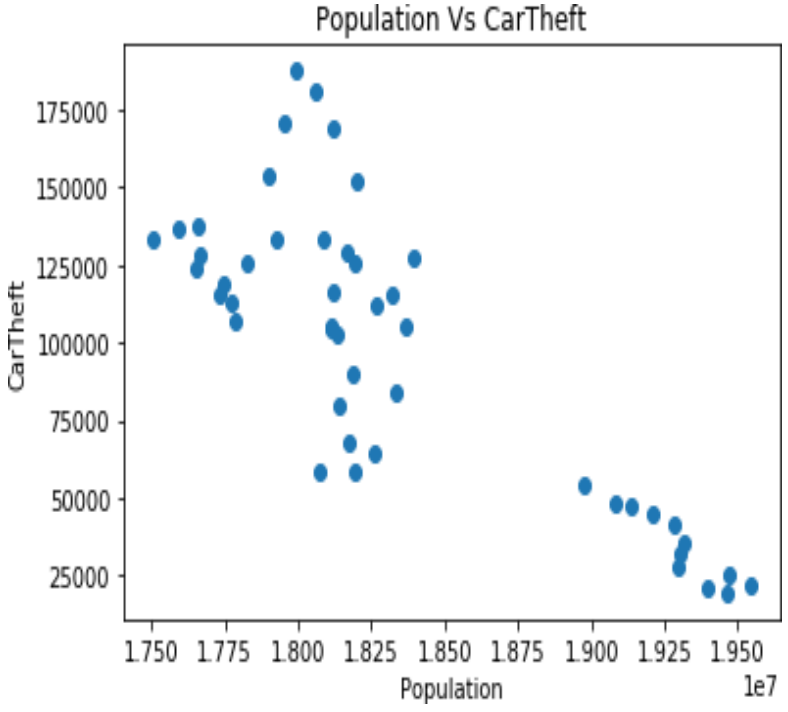
plt.bar(crime_bar.index,crime_bar.Robbery);



```
sns.barplot('Robbery','Year',data=crime);
```




```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data=pd.read_csv('crime.csv')
x=data.Population
y=data.CarTheft
plt.scatter(x,y)
plt.xlabel('Population')
plt.ylabel('CarTheft')
plt.title('Population Vs
CarTheft') plt.show();
```



OUTPUT:

Record Notes

WEEK: 4

Implement Word Count/ Frequency Programs Using Map Reduce.

PROGRAM:

AIM**:** To count a given number using map reduce functions.

Hadoop Streaming API for helping us passing data between our Map and Reduce codevia

STDIN (standard input) and STDOUT (standard output).

Note : Change the file has execution permission (chmod +x /home/hduser/mapper.py)Change

the file has execution permission (chmod +x /home/hduser/reducer.py

Mapper program

mapper.py

```
import sys
# input comes from STDIN (standard
input) for line in sys.stdin:
    line = line.strip() # remove leading and trailing
  whitespace words = line.split()# split the line into words
  # increase counters
  for word in words:
    # write the results to STDOUT (standard
    output); # what we output here will be the
    input for the  # Reduce step, i.e. the input for
    reducer.py
    # tab-delimited; the trivial word count is 1
    print '%s\t%s' % (word, 1)
Reducer program
"""reducer.py"""
from operator import itemgetter
import sys
current_word =
None current_count
= 0 word = None

# input comes from
STDIN for line in
sys.stdin:
  line = line.strip() # remove leading and trailing
  whitespace # parse the input we got from mapper.py
  word, count = line.split('\t', 1)
  # convert count (currently a string) to
  int try:
    count = int(count)
  except ValueError:
    # count was not a number, so
    silently # ignore/discard this line
    continue

  # this IF-switch only works because Hadoop sorts map
  output # by key (here: word) before it is passed to the
  reducer
  if current_word ==
    word: current_count
```

count else:

```
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if
needed! if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Test the code (cat data | map | sort | reduce)

```
hduser@ubuntu:~$ echo "foo foo quux labs foo bar quux" | /home/hduser/mapper.py

foo    1
foo    1
quux    1
labs    1
foo    1
bar    1
quux    1

hduser@ubuntu:~$ echo "foo  foo  quux  labs  foo  bar  quux" | /home/hduser/mapper.py | sort -k1,1 |
/home/hduser/reducer.py
bar    1
foo    3
labs    1
quux    2

hduser@ubuntu:~$ cat /tmp/gutenberg/20417-8.txt |
 /home/hduser/mapper.py The      1
 Project 1
 Gutenberg      1
 EBook  1
 of     1
```

OUTPUT:

Record Notes

WEEK: 5

Implement a MapReduce Program that process a dataset.

AIM:

To createprocess dataset using map reducefunctions**.**

PROGRAM:

The python program reads the data from a dataset (stored in the file data.csv- wine quality).The data mapped is stored in shuffled.pkl using mapper.py.
The contents of shuffled.pkl are reduced using reducer.py

Mapper Program

```
import pandas as pd
import pickle

data = pd.read_csv('data.csv')

#Slicing Data
slice1 = data.iloc[0:399,:]
slice2 = data.iloc[400:800,:]
slice3 = data.iloc[801:1200,:]
slice4 = data.iloc[1201:,:]

def mapper(data):
    mapped = []

    for index,row in data.iterrows():
        mapped.append((row['quality'],row['volatile  acidity']))
    return mapped

map1 = mapper(slice1)
map2 = mapper(slice2)
map3 = mapper(slice3)
map4 = mapper(slice4)

shuffled = {
    3.0: [],
    4.0: [],
    5.0: [],
    6.0: [],
    7.0: [],
    8.0: [],
}
for i in [map1,map2,map3,map4]:
    for j in i:
```

```
        shuffled[j[0]].append(j[1])

file= open('shuffled.pkl','ab')
pickle.dump(shuffled,file)
file.close()

print("Data has been mapped. Now, run reducer.py to reduce the contents in
shuffled.pkl file.")
```

Reducer Program

```
import
pickle
        file= open('shuffled.pkl','rb')
        shuffled = pickle.load(file)
        def reduce(shuffled_dict):
            reduced = {}

            for i in shuffled_dict:

                reduced[i] = sum(shuffled_dict[i])/len(shuffled_dict[i])

            return reduced
        final = reduce(shuffled)
         print("Average volatile acidity in different classes of wine: ")
        for i in final:
                print(i,':',final[i])
```

OUTPUT:

Record Notes

WEEK: 6

Implement Clustering Techniques Using SPARK.

AIM: To create a clusteringusing SPARK.

PROGRAM:

```
# Loads data.
dataset = spark.read.format("libsvm").load("data/mllib/sample_kmeans_data.txt")

# Trains a k-means model.
kmeans = KMeans().setK(2).setSeed(1)
model = kmeans.fit(dataset)

# Evaluate clustering by computing Within Set Sum of Squared
Errors. wssse = model.computeCost(dataset)
print("Within Set Sum of Squared Errors = " + str(wssse))

# Shows the result.
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
print(center)
```

OUTPUT:

Record Notes

WEEK: 7

Implement an Application that Stores Big Data in MONGODB / PIG Using Hadoop /R.

AIM: To design application to stores data in mongdob using hadoop.

PROGRAM:

R Shiny Tutorial: How to Make Interactive Web Applications in RIntroduction

In this modern technological era, various apps are available for all of us –from tracking our fitness level, sleep to giving us the latest information about the stock markets. Apps like Robinhood, GoogleFit and Workit seem so amazingly useful because they use real-time data and statistics. As R is a frontrunner in the field of statistical computing and programming, developers need a system to use its power to build apps.

This is where R Shiny comes to save the day. In this, R Shiny tutorial, you will come to know the basics.

What is R Shiny?

Shiny is an R package that was developed for building interactive web applications in R. Using this, you can create web applications utilizing native HTML and CSS code along with R Shiny code. You can build standalone web apps on a website that will make data visualization easy. These applications made through R Shiny can seamlessly display R objects such as tables and plots.

Let us look at some of the features of R Shiny:

- Build web applications with fewer lines of code, without JavaScript.
- These applications are live and are accessible to users like spreadsheets. The outputs may alter in real-time if the users change the input.
- Developers with little knowledge of web tools can also build apps using R Shiny.
- You get in-built widgets to display tables, outputs of R objects and plots.
- You can add live visualizations and reports to the web application using this package.
- The user interfaces can be coded in R or can be prepared using HTML, CSS or JavaScript.
- The default user interface is built using Bootstrap.
- It comes with a WebSocket package that enables fast communication between the web server and R.

Components of an R Shiny app

A Shiny app has two primary components – a user interface object and a server function. These are the arguments passed on to the shinyApp method. This method creates an application object using the arguments.

Let us understand the basic parts of an R Shiny app in detail:

User interface function

This function defines the appearance of the web application. It makes the application interactive by obtaining input from the user and displaying it on the screen. HTML and CSS tags can be used for making the application look better. So, while building the ui.R file you create an HTML file with R functions.

If you type fluidPage() in the R console, you will see that the method returns a tag .

The different input functions are:

- selectInput() – This method is used for creating a dropdown HTML that has various choices to select.
- numericInput() – This method creates an input area for writing text or numbers.
- radioButtons() – This provides radio buttons for the user to select an input.

Layout methods
The various layout features available in Bootstrap are implemented by R Shiny. The components are:

Panels
These are methods that group elements together into a single panel. These include:

- absolutePanel()
- inputPanel()
- conditionalPanel()
- headerPanel()
- fixedPanel()

Layout functions
These organize the panels for a particular layout. These include:

- fluidRow()
- verticalLayout()
- flowLayout()
- splitLayout()
- sidebarLayout()

Output methods
These methods are used for displaying R output components images, tables and plots. They are:

- tableOutput() – This method is used for displaying an R table
- plotOutput() – This method is used for displaying an R plot object

Server function
After you have created the appearance of the application and the ways to take input values from the user, it is time to set up the server. The server functions help you to write the server-side code for the Shiny app. You can create functions that map the user inputs to the corresponding outputs. This function is called by the web browser when the application is loaded.

It takes an input and output parameter, and return values are ignored. An optional session parameteris also taken by this method.

R Shiny tutorial: How to get started with R Shiny?
Steps to start working with the R Shiny package are as follows:

- Go to the R console and type in the command – install.packages(―shiny‖)
- The package comes with 11 built-in application examples for you to understand how Shiny works

You can start with the Hello Shiny example to understand the basic structure. Type this code to runHello Shiny:

library(shiny)
runExample("01_hello")

The steps to create a new Shiny app are:

- Open RStudio and go to the File option
- Select New Project in a directory and click on the ―Shiny Web‖ Application
- You will get a histogram and a slider to test the changes in output with respect to the input
- You will get two scripts ui.R and server.R for coding and customizing the application

Tips for Shiny app development

- Test the app in the browser to see how it looks before sending it for production
- Run the entire script while debugging the app
- Be careful about common error such as commas

OUTPUT

Record Notes:

WEEK: 8

Implement different String Manipulation functions in R.

AIM: To implement different String Manipulation functions in R.

.

BRIEF DISCUSSION AND EXPLANATION

In R, there are several string manipulation functions available in the base and additional packages. Here are examples of some commonly used string manipulation functions:

1. Concatenation:

```
# Concatenate strings

string1 <- "Hello"

string2 <- "World"


result <- paste(string1, string2)

cat("Concatenated String:", result, "\n")
```

2. Substring Extraction:

```
# Extract substring

original_string <- "DataScience"

substring <- substr(original_string, start = 5, stop = 9
```

```
cat("Substring:", substring, "\n")
```

3. String Length:

```
# Calculate string length

string <- "Programming"

length_result <- nchar(string)

cat("String Length:", length_result, "\n")
```

4. Uppercase and Lowercase:

```
# Convert to uppercase and lowercase

uppercase_string <- toupper(string)

lowercase_string <- tolower(string)


cat("Uppercase String:", uppercase_string, "\n")

cat("Lowercase String:", lowercase_string, "\n")
```

5. String Replacement:

```
# Replace a substring

original_string <- "I love programming in R"

modified_string <- gsub("R", "Python", original_string)


cat("Modified String:", modified_string, "\n")
```

6. Splitting Strings:

```
# Split a string
text <- "apple,orange,banana"
split_result <- strsplit(text, ",")
cat("Split Result:", unlist(split_result), "\n")
```

OUTPUT

Exercise:

How readable and usable was the code using these string manipulation functions? Did any of them enhance or hinder code clarity?

WEEK: 9

Create a data set and do statistical analysis on the data using R.

AIM: To Create a data set and do statistical analysis on the data using R.

BRIEF DISCUSSION AND EXPLANATION

Let's create a simple dataset and perform some basic statistical analysis using R. In this example, I'll create a dataset with two variables, 'Height' and 'Weight', and then calculate descriptive statistics and conduct a t-test.

```
# Create a dataset

set.seed(123) # Setting seed for reproducibility

height <- rnorm(50, mean = 170, sd = 10)

weight <- rnorm(50, mean = 70, sd = 5)


# Combine variables into a data frame

my_data <- data.frame(Height = height, Weight = weight)


# Display the first few rows of the dataset

print("First few rows of the dataset:")
```

```
print(head(my_data))

# Descriptive statistics
print("Descriptive statistics:")
print(summary(my_data))
# T-test for comparing means of 'Height' between two groups
group1 <- my_data$Height[1:25]
group2 <- my_data$Height[26:50]

t_test_result <- t.test(group1, group2)
print("T-test for comparing means of 'Height' between two groups:")
print(t_test_result)
```

OUTPUT

Exercise:

What does the t-test result suggest about the difference in means between the two groups?

WEEK: 10

Write a program to read a csv file and analyze the data in the file in R.

BRIEF DISCUSSION AND EXPLANATION

In R, you can use the **read.csv()** function to read a CSV file and then perform various analyses on the data. Here's a simple example:

Assuming you have a CSV file named "data.csv" with the following content:

Name, Age, Grade Alice,

25, A

Bob, 30, B

Charlie, 22, C

Now, let's write a program to read and analyze this CSV file:

```
# Read CSV file

data <- read.csv("data.csv", header = TRUE)
```

```
# Display the data
print("Data in the CSV
file:") print(data)
# Summary statistics
summary_stats <- summary(data$Age)
print(paste("Summary statistics for Age:",
summary_stats))
# Mean of Age
mean_age <-
mean(data$Age)
print(paste("Mean Age:",
mean_age))
# Maximum Age
max_age <- max(data$Age)
print(paste("Maximum Age:",
max_age))
# Minimum Age
min_age <- min(data$Age) print(paste("Minimum Age:", min_age))
```

This program reads the CSV file, displays the data, calculates summary statistics for the "Age" column, calculates the mean, maximum, and minimum values of the "Age" column. Adjust the code according to your specific requirements and the structure of your CSV file.

OUTPUT

Exercise:

How would you describe the quality of the data in the CSV file? Were there any issues encountered during data analysis that might indicate data quality concerns??

"A thought beyond horizons of success committed for professional excellence"

Vision

To be a world class university visualizing a great future for the young aspirants, with innovative nature, research culture and ethical sensitivities to meet the global challenges improving the quality of human life.

Mission

To impart value based futuristic higher education moulding students into globally competent empowered youth, rich in culture and ethics along with professional expertise.

To promote innovation, entrepreneurship, research and development for the broad purpose of fulfilling societal goals such as societal welfare and benefit.

Quality Policy

To pursue continual improvement of teaching learning process of Undergraduate, Post Graduate programs and Research Programs vigorously.

To provide state of the art infrastructure and expertise to impart the quality education.

To groom the students to become intellectually creative and professionally competitive.

To explore the opportunities in the professional fields.

Academic Best Practices

Industry Oriented Curriculum

Technology Training and Certifications

Institution to Corporate Exposure

Interactive Learning

International Career Guidance

Choice Based Flexible Curriculum

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Maisammaguda, Kompally, Hyderabad - 500 100

Telangana State

www.mallareddyuniversity.ac.in