DEPARTMENT OF INFORMATION TECHNOLOGY

BIG DATA ANALYTICS
LAB MANUAL

IV B.TECH I SEMESTER

DEPARTMENT

OF

INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous institution –UGC , Govt of India)
Affiliated to JNTUH & Approved by AICTE,New Delhi
2025-2026

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

- To achieve high quality in technical education that provides the skills and attitude to adapt to the global needs of the Information Technology sector, through academic and research excellence.

MISSION

- To equip the students with the cognizance for problem solving and to improve the teaching learning pedagogy by using innovative techniques.
- To strengthen the knowledge base of the faculty and students with motivation towards possession of effective academic skills and relevant research experience.
- To promote the necessary moral and ethical values among the engineers, for the betterment of the society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

- To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:** Able to understand the working principles of the computer System and its components, Apply the knowledge to build, asses, and analyze the software and hardware aspects of it.

2. **The comprehensive and applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

DEPARTMENT OF INFORMATION TECHNOLOGY

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:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

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

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.
8. 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.
9. 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.
10. 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.

HEAD OF THE DEPARTMENT **PRINCIPAL**

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

IV Year B.TECH -IT-I-SEM **L/T/P/C**

-/-/2/1

(R22A0590) BIG DATA ANALYTICS LAB

COURSE OBJECTIVES:

The objectives of this course are

1. To Understand Configuration of various big data Frame Works.
2. To learn various visualization techniques to explore data
3. To implement Map Reduce programs for processing big data.
4. To realize storage of big data using MongoDB.
5. To implement clustering techniques using SPARK and to analyze data using machine learning techniques.

List of Experiments

1. Install, configure and run python, numPy and Pandas.
2. Install, configure and run Hadoop and HDFS.
3. Visualize data using basic plotting techniques in Python.
4. Implement NoSQL Database Operations: CRUD operations, Arrays using MongoDB.
5. Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB.
6. Implement word count / frequency programs using MapReduce.
7. Implement a MapReduce program that processes a dataset.
8. Implement clustering techniques using SPARK.
9. Implement an application that stores big data in MongoDB / Pig using Hadoop / R.

Course Outcomes:

Upon completion of the course, the students should be able to:

1. Proficient in Configuration of various big data Frame Works.
2. Apply various visualization techniques to explore data.
3. Demonstrate data base operations using MongoDB.
4. Build and apply Map-Reduce & NoSQL Concepts.
5. Frame clustering techniques using SPARK.
6. Perform data analysis with machine learning methods.

BIG DATA ANALYTICS LAB

Table of Contents

EXPERIMENT: 1

Install, configure and run python, numpy and pandas.

PROGRAM:

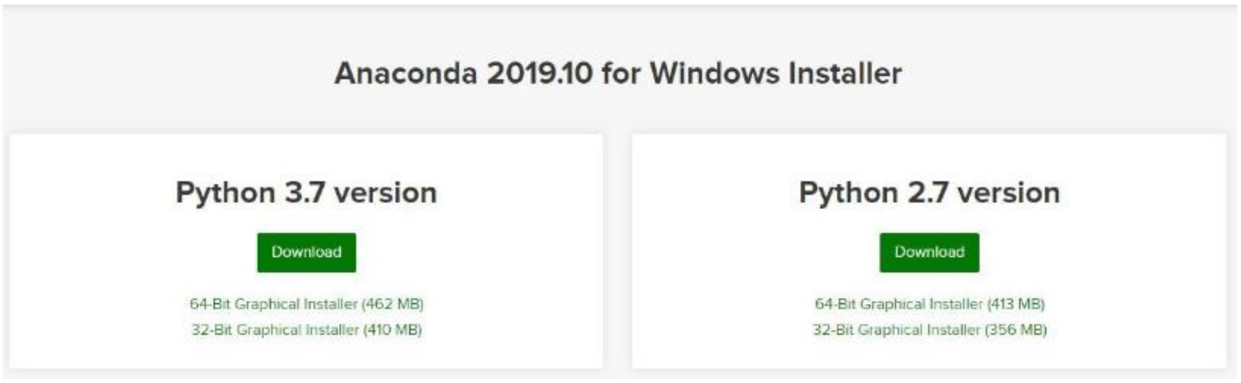
AIM: To Installing and Running Applications On python, numpy and pandas.

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 conda.
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.



Anaconda 2019.10 for Windows Installer

Python 3.7 version

Download

64-Bit Graphical Installer (462 MB)

32-Bit Graphical Installer (410 MB)

Python 2.7 version

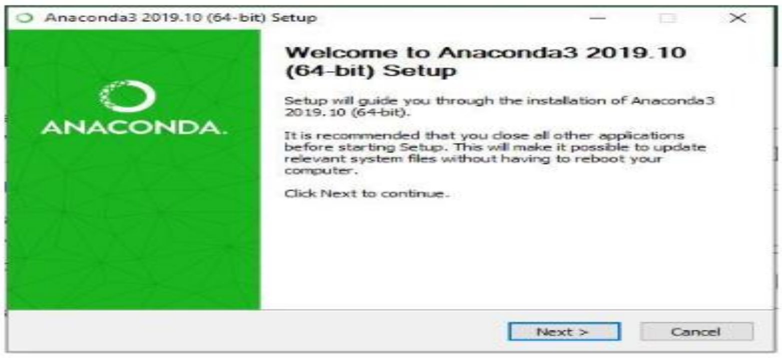
Download

64-Bit Graphical Installer (413 MB)

32-Bit Graphical Installer (356 MB)

Begin with the installation process:

- **Getting Started**



Anaconda3 2019.10 (64-bit) Setup

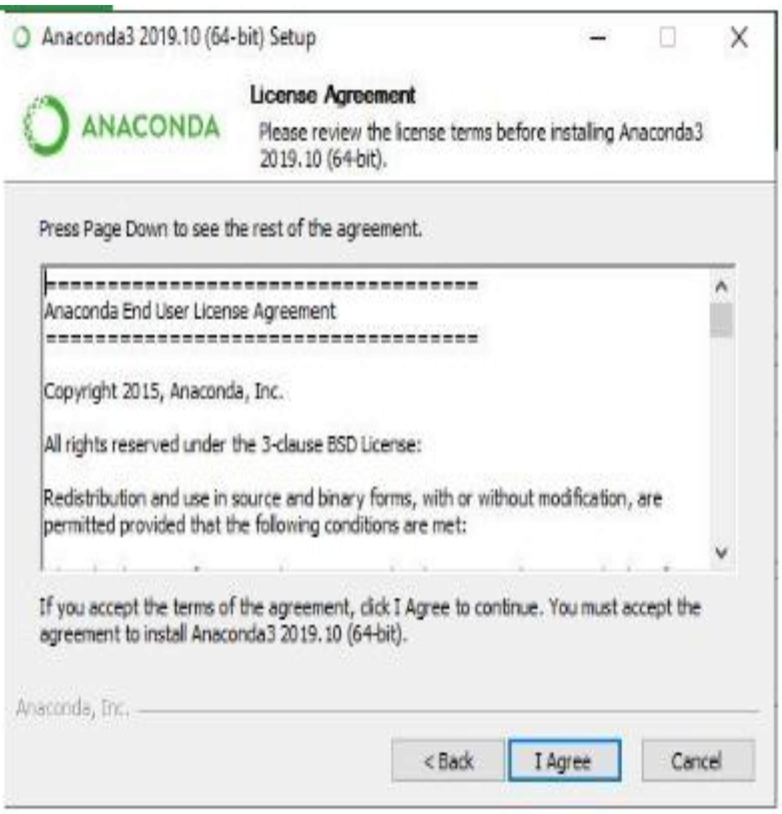
Welcome to Anaconda3 2019.10 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2019.10 (64-bit).

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

Click Next to continue.

Getting through the License Agreement:


Anaconda3 2019.10 (64-bit) Setup

ANACONDA

License Agreement

Please review the license terms before installing Anaconda3 2019.10 (64-bit).

Press Page Down to see the rest of the agreement.

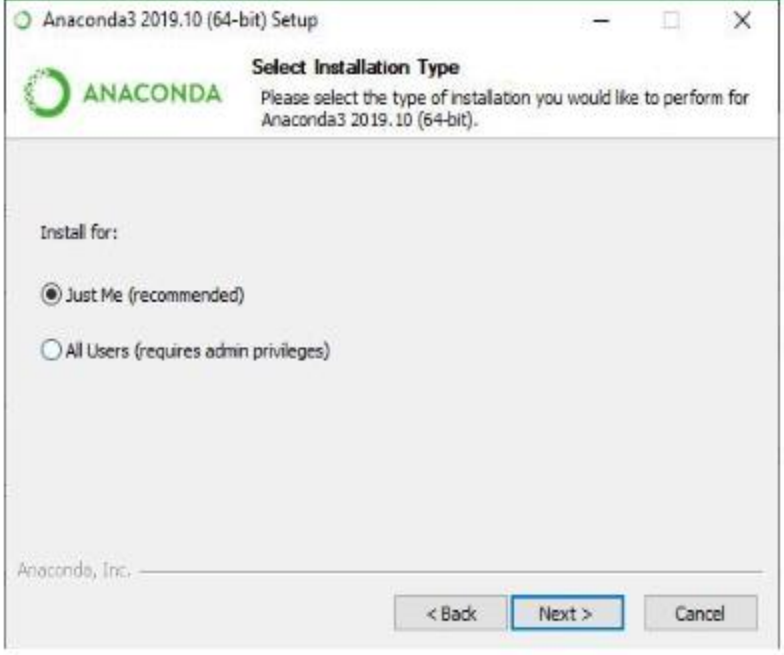
```
====================================
Anaconda End User License Agreement
====================================

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2019.10 (64-bit).

Anaconda, Inc.

< Back | I Agree | Cancel

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


Anaconda3 2019.10 (64-bit) Setup

ANACONDA

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2019.10 (64-bit).

Install for:

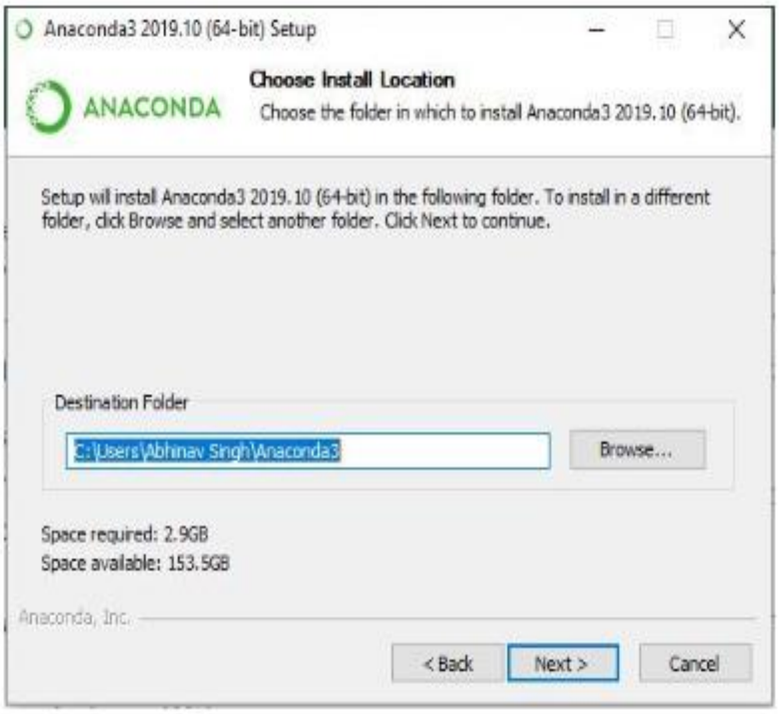
[✓ Just Me (recommended)]

All Users (requires admin privileges)

Anaconda, Inc.

< Back | Next > | Cancel

Choose Installation Location:



Anaconda3 2019.10 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2019.10 (64-bit).

Setup will install Anaconda3 2019.10 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

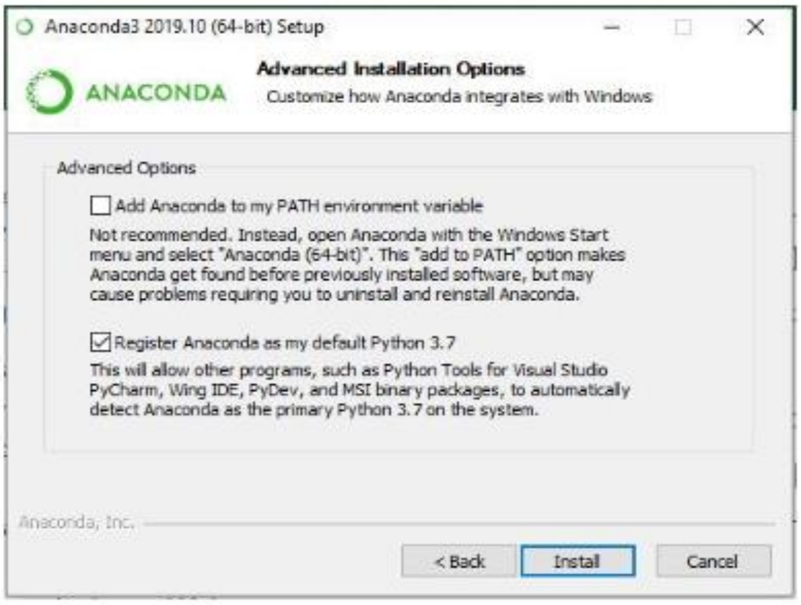
C:\Users\Abhinav Singh\Anaconda3

Space required: 2.9GB

Space available: 153.5GB

Anaconda, Inc.

Advanced Installation Option:



Anaconda3 2019.10 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda to my PATH environment variable

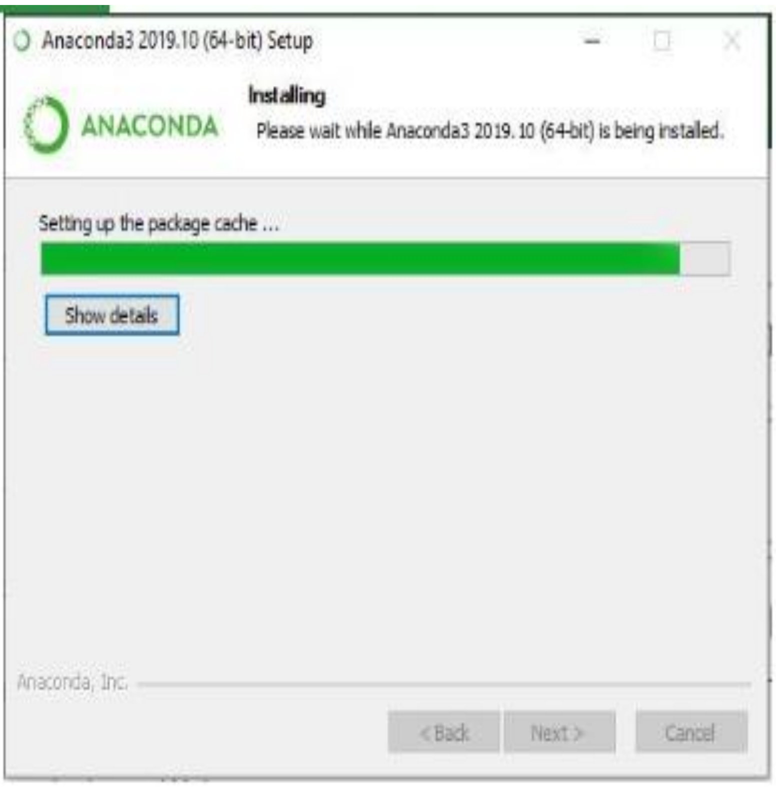
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

Anaconda, Inc.

Getting through the Installation Process:



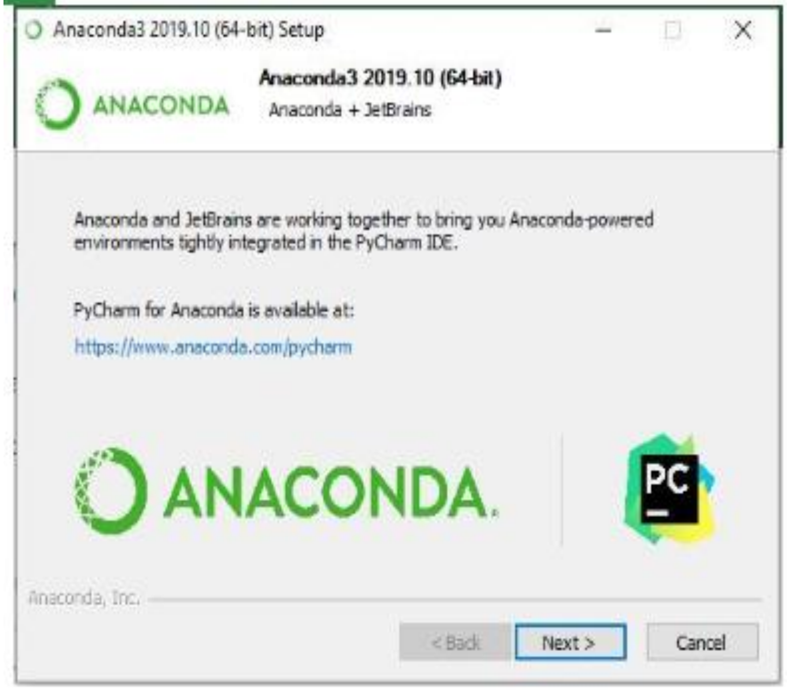
Anaconda3 2019.10 (64-bit) Setup

Installing

Please wait while Anaconda3 2019.10 (64-bit) is being installed.

Setting up the package cache ...

Recommendation to Install Pycharm:



Anaconda3 2019.10 (64-bit) Setup

Anaconda3 2019.10 (64-bit)

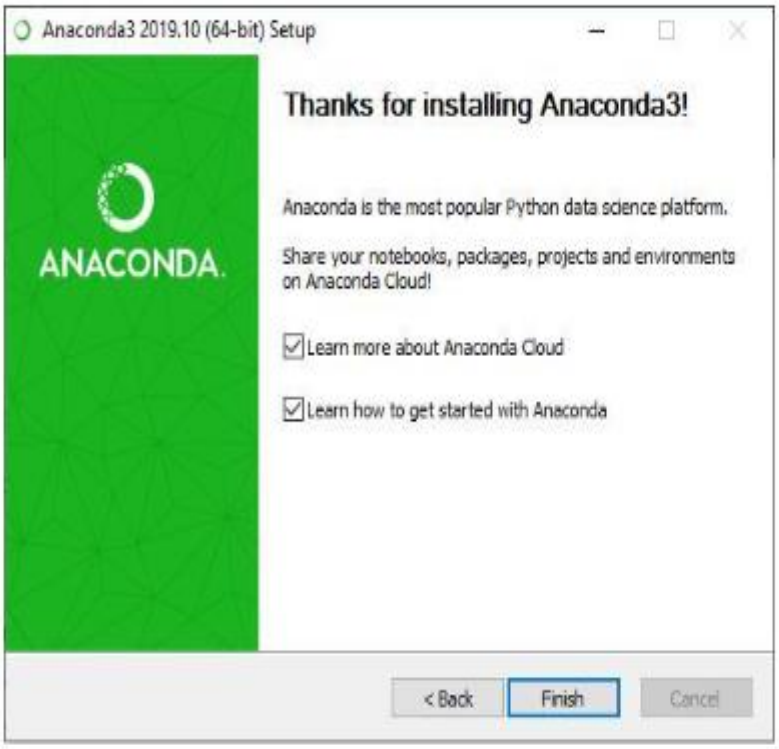
Anaconda + JetBrains

Anaconda and JetBrains are working together to bring you Anaconda-powered environments tightly integrated in the PyCharm IDE.

PyCharm for Anaconda is available at:

https://www.anaconda.com/pycharm

Finishing up the Installation:


Anaconda3 2019.10 (64-bit) Setup

Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

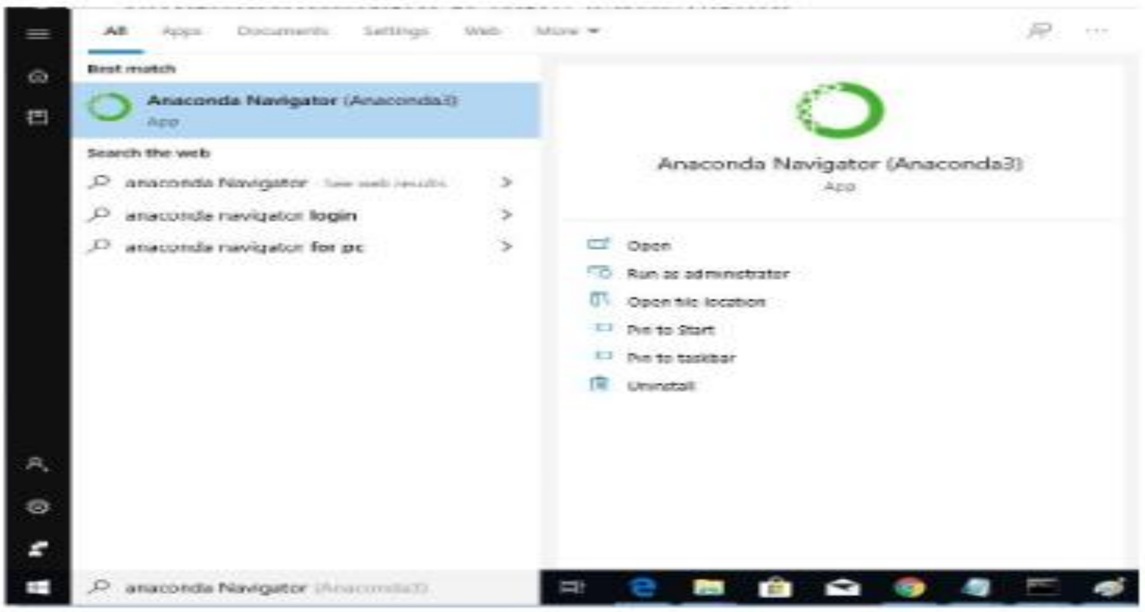
☑ Learn more about Anaconda Cloud

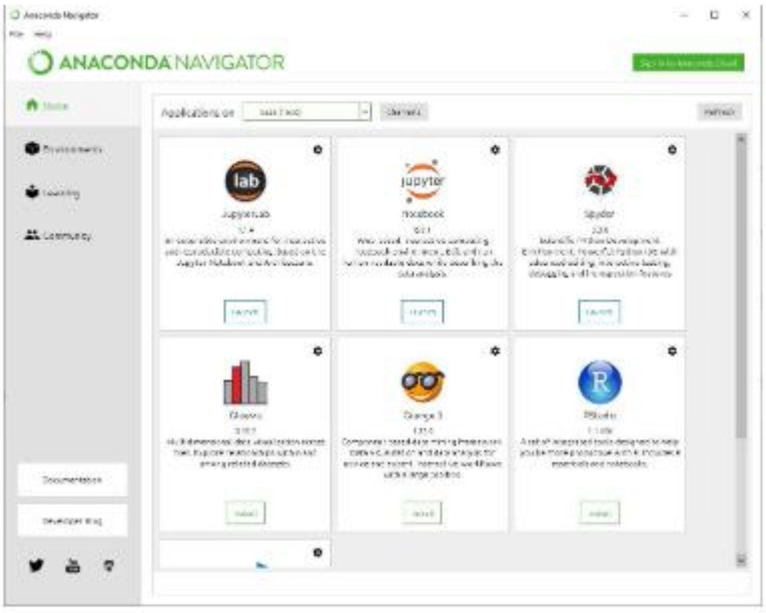
☑ Learn how to get started with Anaconda

< Back | Finish | Cancel

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.

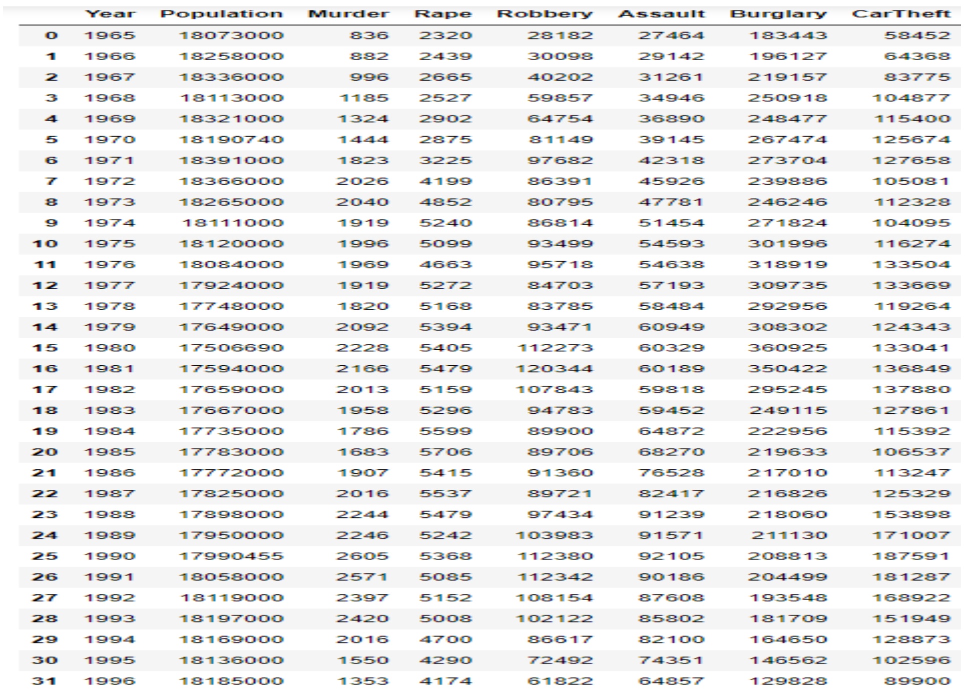




#import pandas in jupyter notebook
import pandas

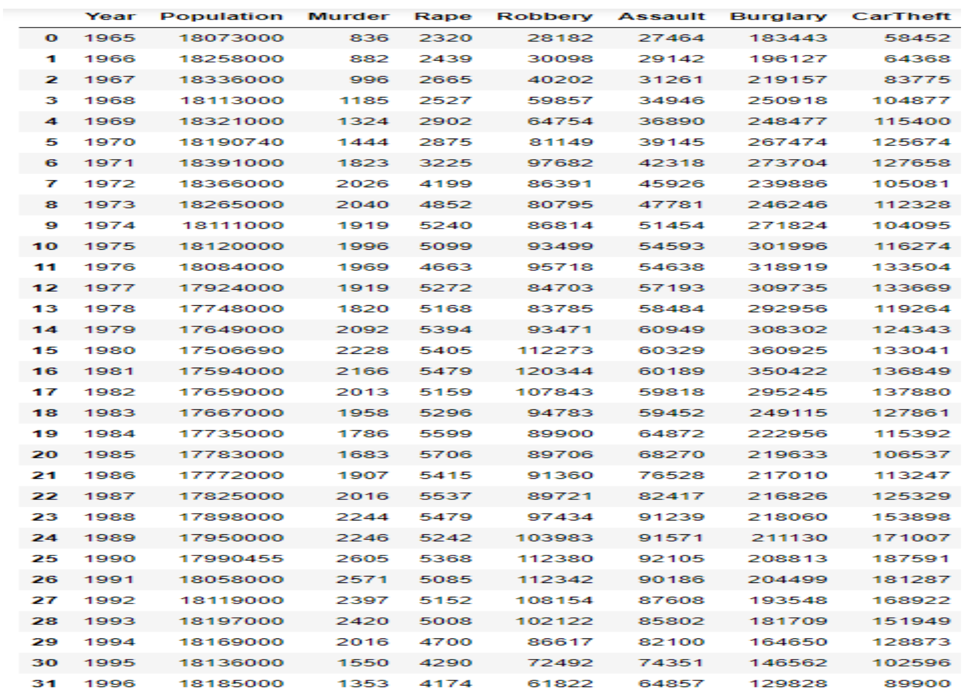
#loading the dataset which is excel file
dataset = pandas.read_csv("crime.csv")

#displaying the data
dataset



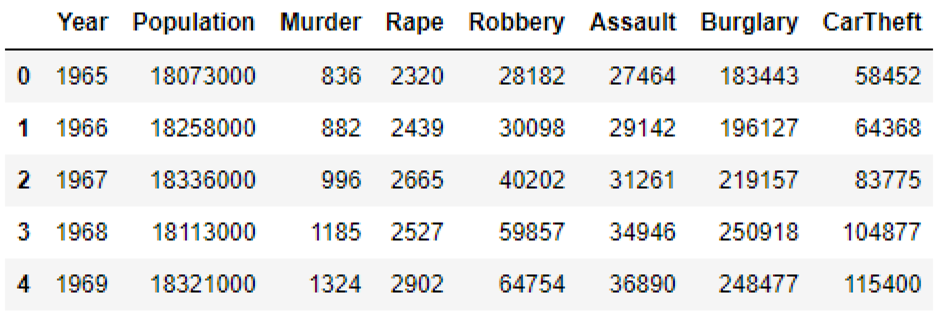
	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890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



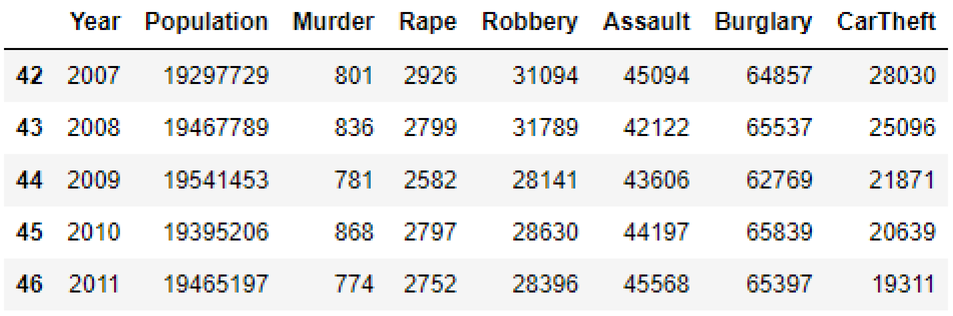
	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
0	1965	18073000	836	2320	28182	27464	183443	58452
1	1966	18258000	882	2439	30098	29142	196127	64368
2	1967	18336000	996	2665	40202	31261	219157	83775
3	1968	18113000	1185	2527	59857	34946	250918	104877
4	1969	18321000	1324	2902	64754	36890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

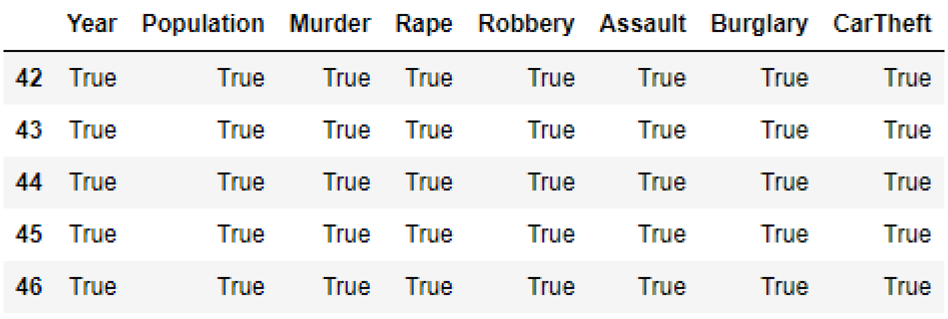
type(dataset1)
pandas.core.frame.DataFrame

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

#to find any null values in the last 5 rows
dataset1.isnull().tail()

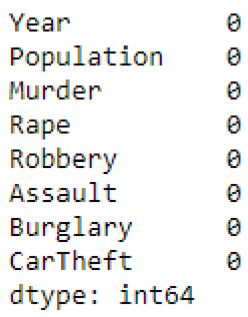
	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
42	False	False	False	False	False	False	False	False
43	False	False	False	False	False	False	False	False
44	False	False	False	False	False	False	False	False
45	False	False	False	False	False	False	False	False
46	False	False	False	False	False	False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

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



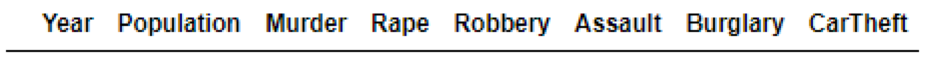
	Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft
42	True	True	True	True	True	True	True	True
43	True	True	True	True	True	True	True	True
44	True	True	True	True	True	True	True	True
45	True	True	True	True	True	True	True	True
46	True	True	True	True	True	True	True	True

#displays the number of null values in each column
dataset1.isnull().sum()



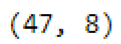
```
Year          0
Population    0
Murder        0
Rape          0
Robbery       0
Assault       0
Burglary      0
CarTheft      0
dtype: int64
```

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



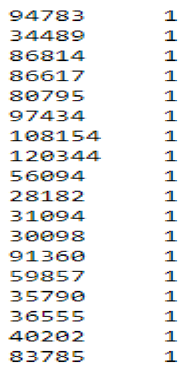
Year	Population	Murder	Rape	Robbery	Assault	Burglary	CarTheft

dataset1.shape



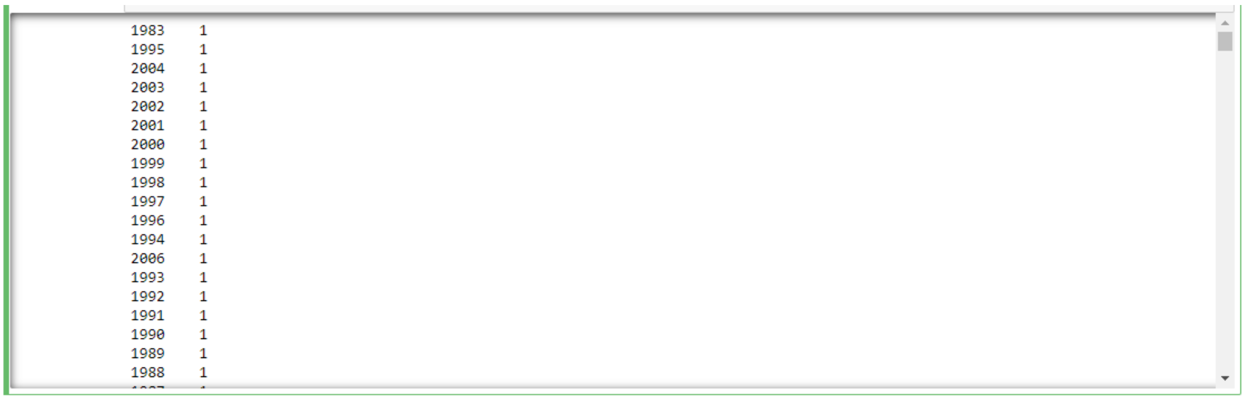
```
(47, 8)
```

#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 dataset
for 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
```

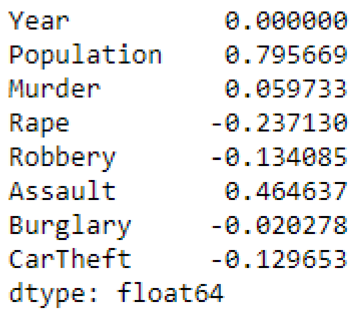
#helps to find the summary of numerical columns
dataset1.describe()

	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()

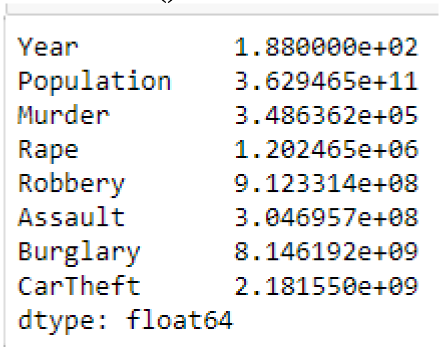
```
count      47.000000
mean     1549.978723
std       590.454265
min       774.000000
25%       922.500000
50%      1683.000000
75%      2016.000000
max      2605.000000
Name: Murder, dtype: float64
```

dataset1.skew()



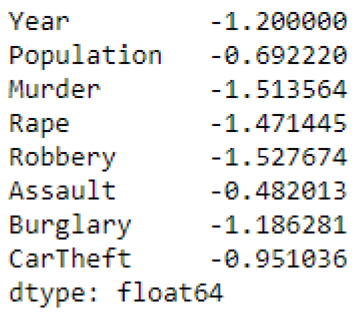
```
Year           0.000000
Population     0.795669
Murder         0.059733
Rape          -0.237130
Robbery       -0.134085
Assault        0.464637
Burglary      -0.020278
CarTheft      -0.129653
dtype: float64
```

dataset1.var()



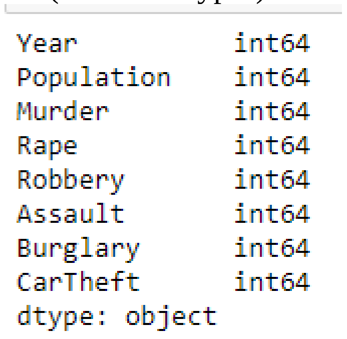
```
Year          1.880000e+02
Population    3.629465e+11
Murder        3.486362e+05
Rape          1.202465e+06
Robbery       9.123314e+08
Assault       3.046957e+08
Burglary      8.146192e+09
CarTheft      2.181550e+09
dtype: float64
```

dataset1.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
```

print(dataset1.dtypes)



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

NUMPY

Numpy is the core library for scientific and numerical computing in Python. It provides high performance 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 of the 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 above code 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 it will 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]
```

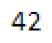
Dimensions in Arrays
A dimension in arrays is one level of array depth (nested arrays).

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)
```

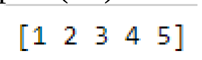


```
42
```

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])
print(arr)
```

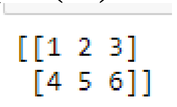


```
[1 2 3 4 5]
```

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)
```

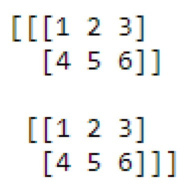


```
[[1 2 3]
 [4 5 6]]
```

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)
```

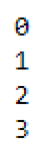


```
[[[1 2 3]
  [4 5 6]]

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

Check Number of Dimensions?

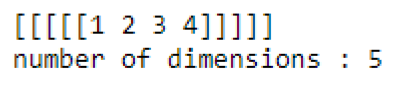
NumPy Arrays provides the ndim attribute that returns an integer that tells us how many dimensions the array have.

```
#Check how many dimensions the arrays have:
import numpy as np
a = 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.ndim)
print(c.ndim)
print(d.ndim)
```



```
0
1
2
3
```

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



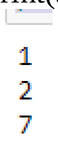
```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

NumPy Array Indexing

Access 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 second has 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])
```

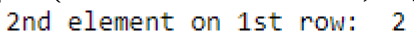


```
1
2
7
```

Access 2-D Arrays

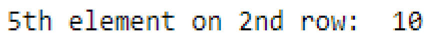
To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.
Think of 2-D arrays like a table with rows and columns, where the dimension represents the row and the index represents the column.

```
#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
```

EXPERIMENT: 2

Install, Configure and Run Hadoop and HDFS

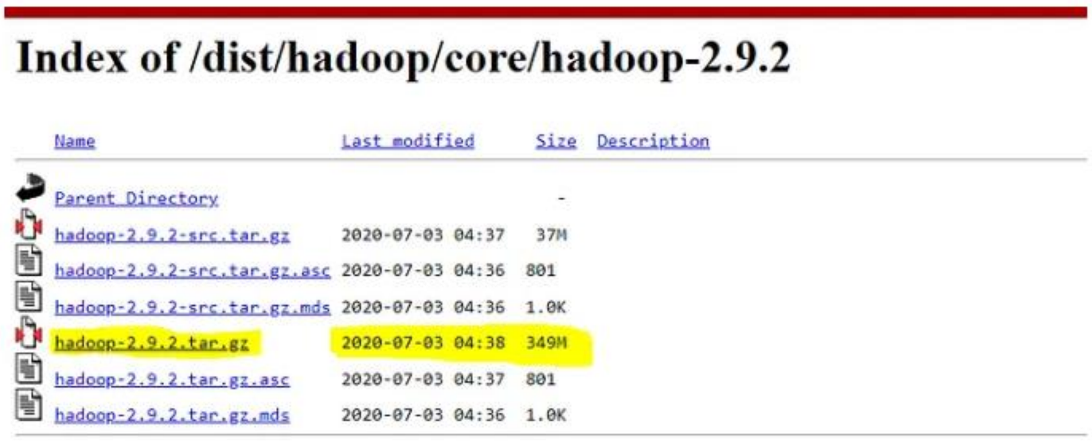
PROGRAM:

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.



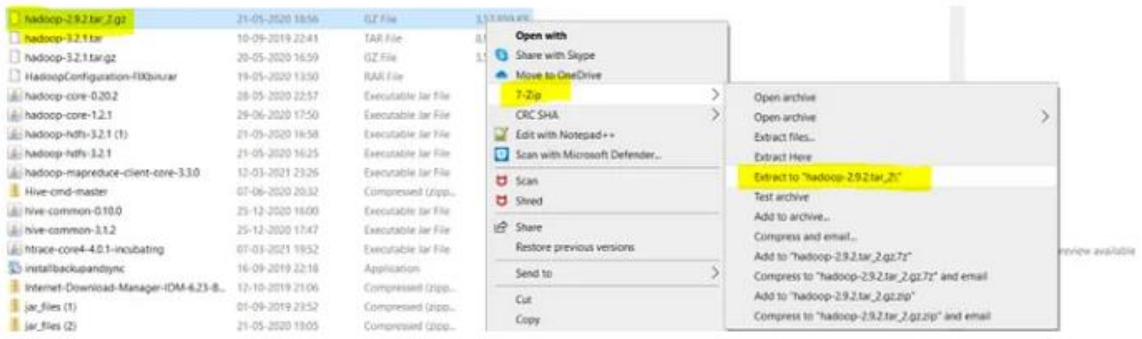
```
Index of /dist/hadoop/core/hadoop-2.9.2

Name                          Last modified      Size  Description

Parent Directory                                   -
hadoop-2.9.2-src.tar.gz       2020-07-03 04:37   37M
hadoop-2.9.2-src.tar.gz.asc   2020-07-03 04:36   801
hadoop-2.9.2-src.tar.gz.mds   2020-07-03 04:36   1.0K
hadoop-2.9.2.tar.gz           2020-07-03 04:38   349M
hadoop-2.9.2.tar.gz.asc       2020-07-03 04:37   801
hadoop-2.9.2.tar.gz.mds       2020-07-03 04:36   1.0K
```

Fig. 1:- Download Hadoop 2.9.2

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

2. Unzip and Install Hadoop

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

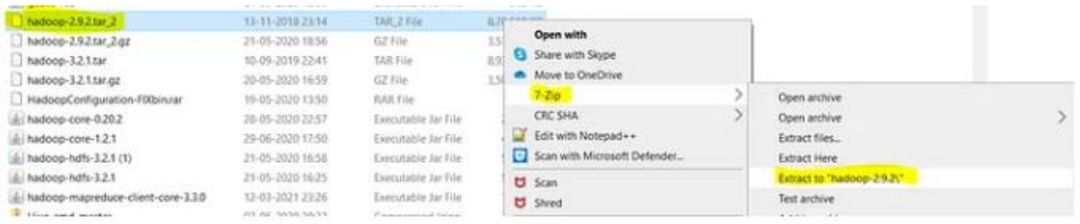


```
hadoop-2.9.2.tar_2.gz                        21-05-2020 18:56   GZ File
hadoop-3.2.1.tar                             10-09-2019 22:41   TAR File
hadoop-3.2.1.tar.gz                          20-05-2020 16:59   GZ File
HadoopConfiguration-FIXbin.rar               19-05-2020 13:50   RAR File
hadoop-core-0.20.2                           28-05-2020 22:57   Executable Jar File
hadoop-core-1.2.1                            29-06-2020 17:50   Executable Jar File
hadoop-hdfs-3.2.1 (1)                        21-05-2020 16:58   Executable Jar File
hadoop-hdfs-3.2.1                            21-05-2020 16:25   Executable Jar File
hadoop-mapreduce-client-core-3.3.0           12-03-2021 23:26   Executable Jar File
Hive-cmd-master                              07-06-2020 20:32   Compressed (zipp...
hive-common-0.10.0                           25-12-2020 16:00   Executable Jar File
hive-common-3.1.2                            25-12-2020 17:47   Executable Jar File
htrace-core4-4.0.1-incubating                07-03-2021 19:52   Executable Jar File
installbackupandsync                         16-09-2019 22:18   Application
Internet-Download-Manager-IDM-6.23-B...      12-10-2019 21:06   Compressed (zipp...
jar_files (1)                                01-09-2019 23:52   Compressed (zipp...
jar_files (2)                                21-05-2020 19:05   Compressed (zipp...
```

Open with
Share with Skype
Move to OneDrive
7-Zip
CRC SHA
Edit with Notepad++
Scan with Microsoft Defender...
Scan
Shred
Share
Restore previous versions
Send to
Cut
Copy
Open archive
Open archive
Extract files...
Extract Here
Extract to "hadoop-2.9.2.tar_2\"
Test archive
Add to archive...
Compress and email...
Add to "hadoop-2.9.2.tar_2.gz.7z"
Compress to "hadoop-2.9.2.tar_2.gz.7z" and email
Add to "hadoop-2.9.2.tar_2.gz.zip"
Compress to "hadoop-2.9.2.tar_2.gz.zip" and email

Fig. 2:- Extracting Hadoop Step-1

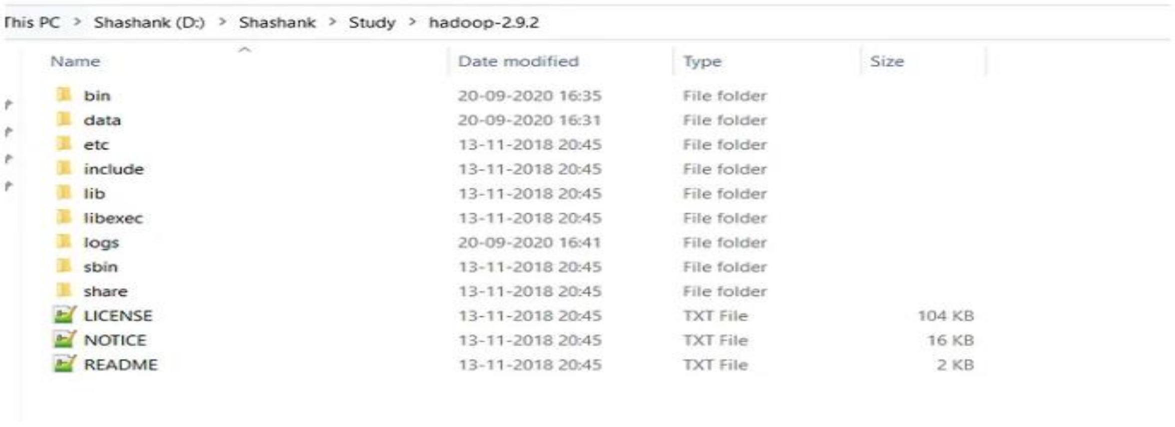
Once extracted, we would get a new file hadoop-2.9.2.tar.
Now, once again we need to extract this tar file.



hadoop-2.9.2.tar_2	13-11-2018 23:14	TAR_2 File
hadoop-2.9.2.tar_2.gz	21-05-2020 18:56	GZ File
hadoop-3.2.1.tar	10-09-2019 22:41	TAR File
hadoop-3.2.1.tar.gz	20-05-2020 16:59	GZ File
HadoopConfiguration-FIXbin.rar	19-05-2020 13:50	RAR File
hadoop-core-0.20.2	28-05-2020 22:57	Executable Jar File
hadoop-core-1.2.1	29-06-2020 17:50	Executable Jar File
hadoop-hdfs-3.2.1 (1)	21-05-2020 16:58	Executable Jar File
hadoop-hdfs-3.2.1	21-05-2020 16:25	Executable Jar File
hadoop-mapreduce-client-core-3.3.0	12-03-2021 23:26	Executable Jar File

Fig. 3:- Extracting Hadoop Step-2

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



This PC > Shashank (D:) > Shashank > Study > hadoop-2.9.2

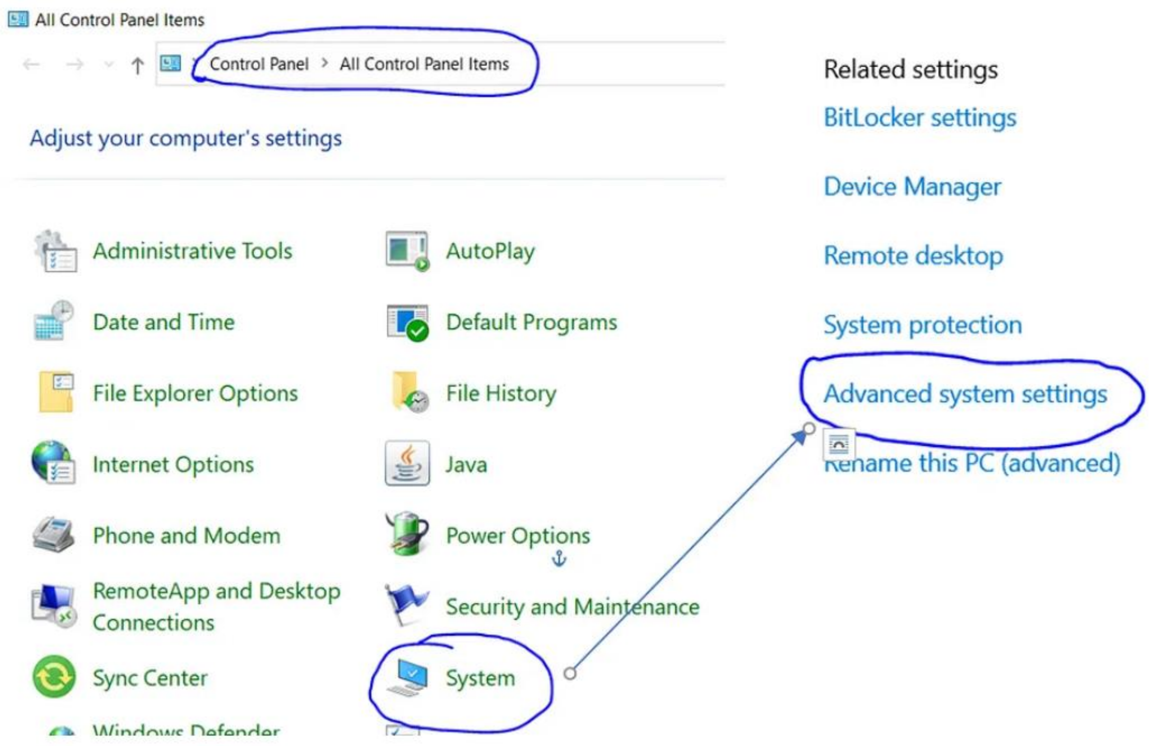
Name	Date modified	Type	Size
bin	20-09-2020 16:35	File folder	
data	20-09-2020 16:31	File folder	
etc	13-11-2018 20:45	File folder	
include	13-11-2018 20:45	File folder	
lib	13-11-2018 20:45	File folder	
libexec	13-11-2018 20:45	File folder	
logs	20-09-2020 16:41	File folder	
sbin	13-11-2018 20:45	File folder	
share	13-11-2018 20:45	File folder	
LICENSE	13-11-2018 20:45	TXT File	104 KB
NOTICE	13-11-2018 20:45	TXT File	16 KB
README	13-11-2018 20:45	TXT File	2 KB

Fig. 4:- Hadoop Directory

Please note while creating folders, DO NOT ADD SPACES IN BETWEEN THE FOLDER NAME.(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 environment variable.
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…


All Control Panel Items

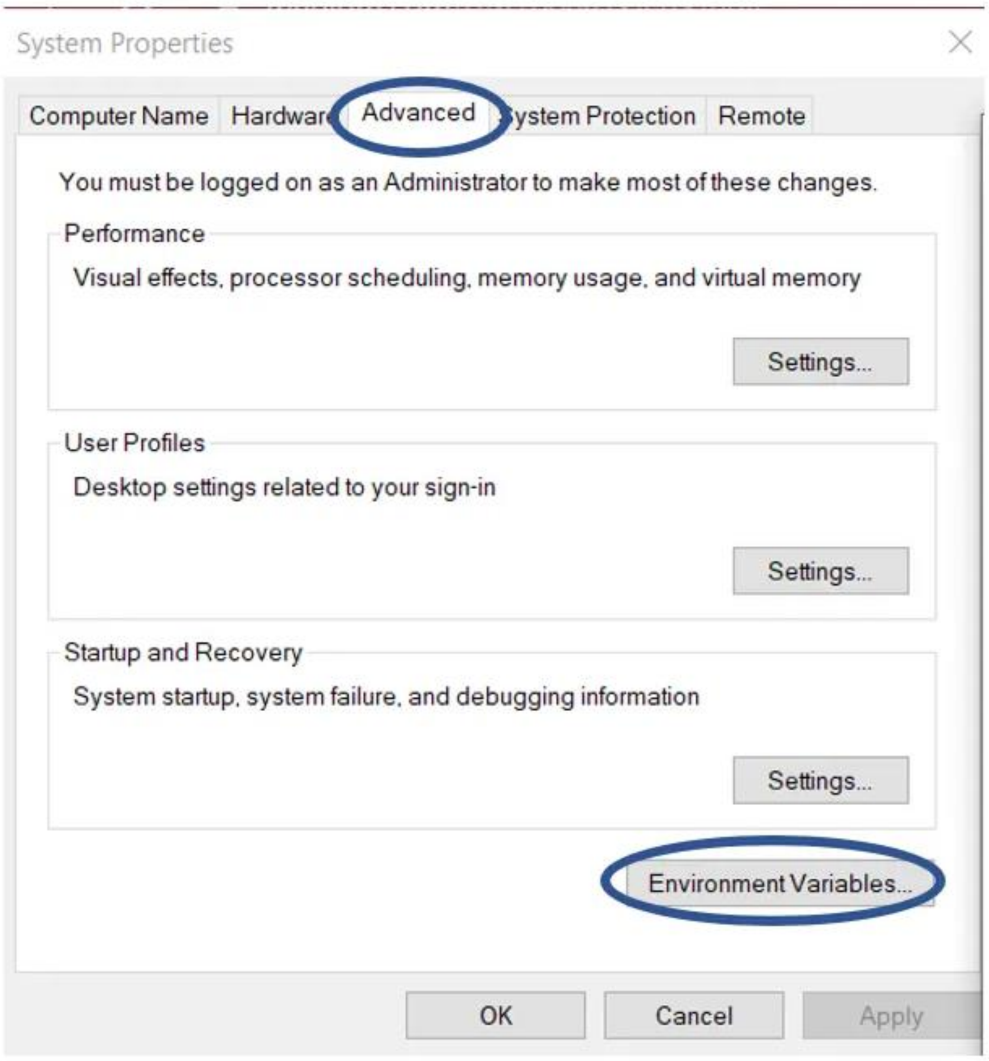
Control Panel > All Control Panel Items

Adjust your computer's settings

Administrative Tools
AutoPlay
Date and Time
Default Programs
File Explorer Options
File History
Internet Options
Java
Phone and Modem
Power Options
RemoteApp and Desktop Connections
Security and Maintenance
Sync Center
System
Windows Defender

Related settings
BitLocker settings
Device Manager
Remote desktop
System protection
Advanced system settings
Rename this PC (advanced)

Fig. 5:- Path for Environment Variable



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

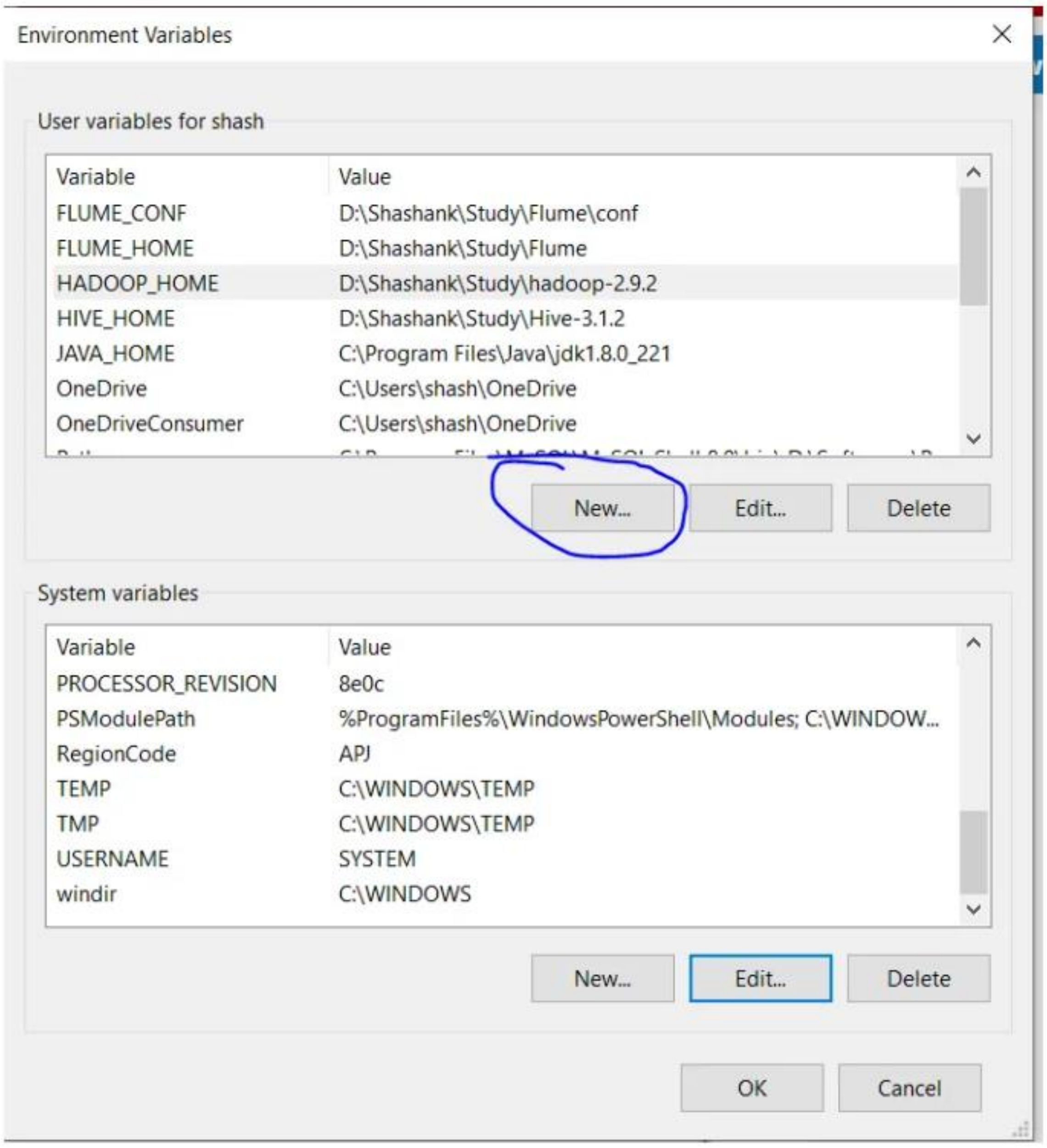
Startup and Recovery

System startup, system failure, and debugging information

Fig. 6:- Advanced System Settings Screen

3.1 Setting JAVA_HOME

Open environment Variable and click on "New" in "User Variable"


Environment Variables

User variables for shash

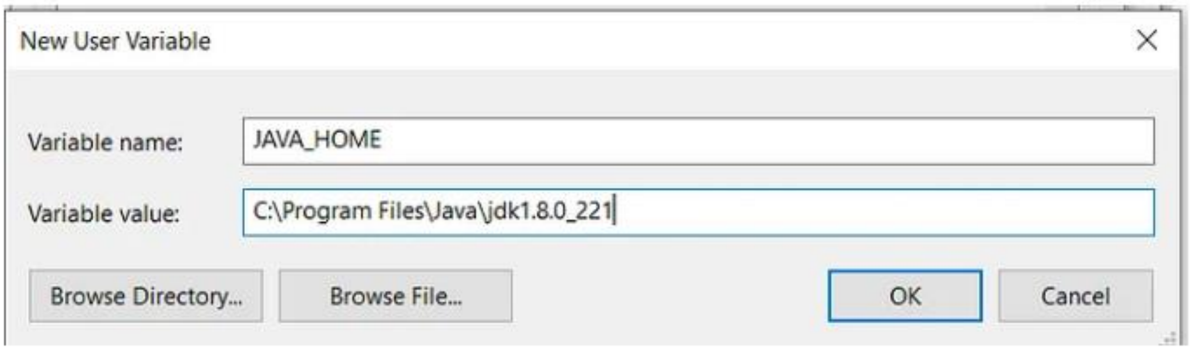
Variable	Value
FLUME_CONF	D:\Shashank\Study\Flume\conf
FLUME_HOME	D:\Shashank\Study\Flume
HADOOP_HOME	D:\Shashank\Study\hadoop-2.9.2
HIVE_HOME	D:\Shashank\Study\Hive-3.1.2
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_221
OneDrive	C:\Users\shash\OneDrive
OneDriveConsumer	C:\Users\shash\OneDrive

System variables

Variable	Value
PROCESSOR_REVISION	8e0c
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules; C:\WINDOW...
RegionCode	APJ
TEMP	C:\WINDOWS\TEMP
TMP	C:\WINDOWS\TEMP
USERNAME	SYSTEM
windir	C:\WINDOWS

Fig. 7:- Adding Environment Variable

On clicking "New", we get below screen.


New User Variable

Variable name: JAVA_HOME

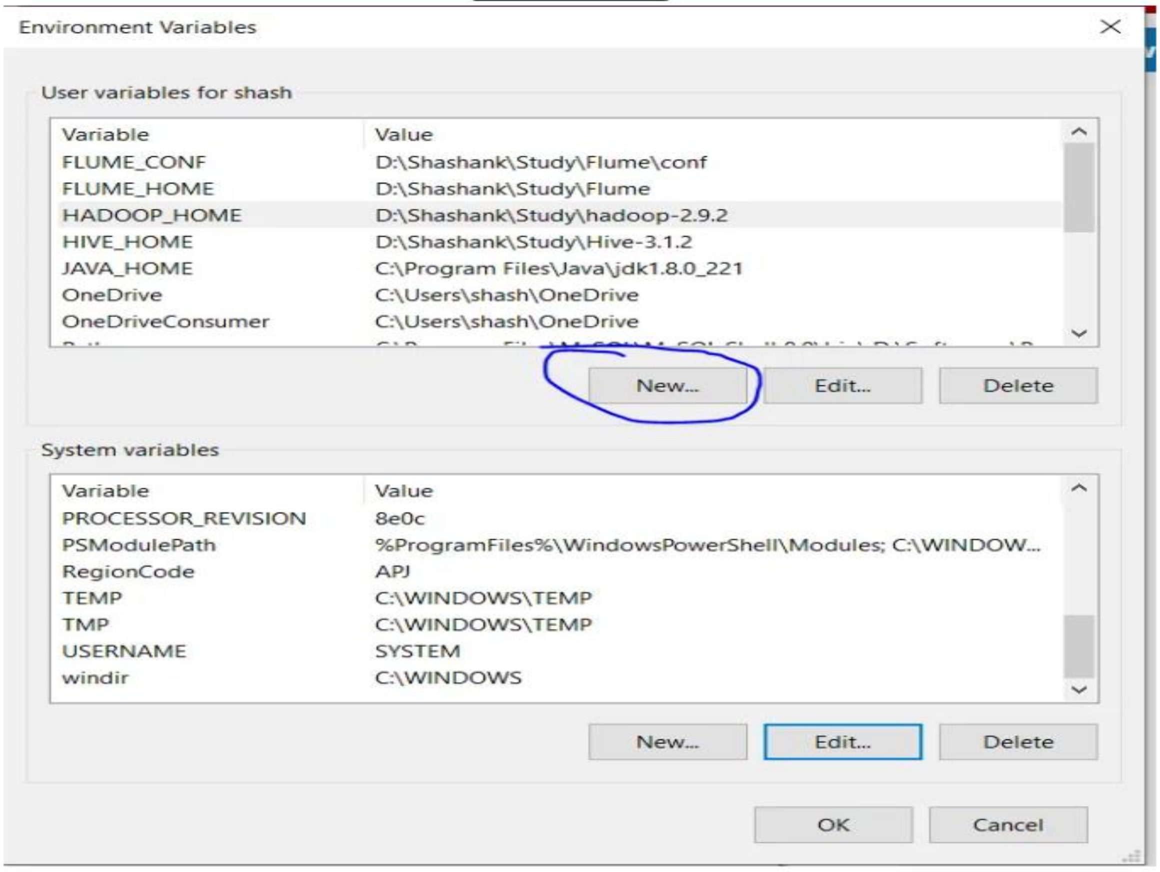
Variable value: C:\Program Files\Java\jdk1.8.0_221

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

Open environment Variable and click on "New" in "User Variable"



Environment Variables

User variables for shash

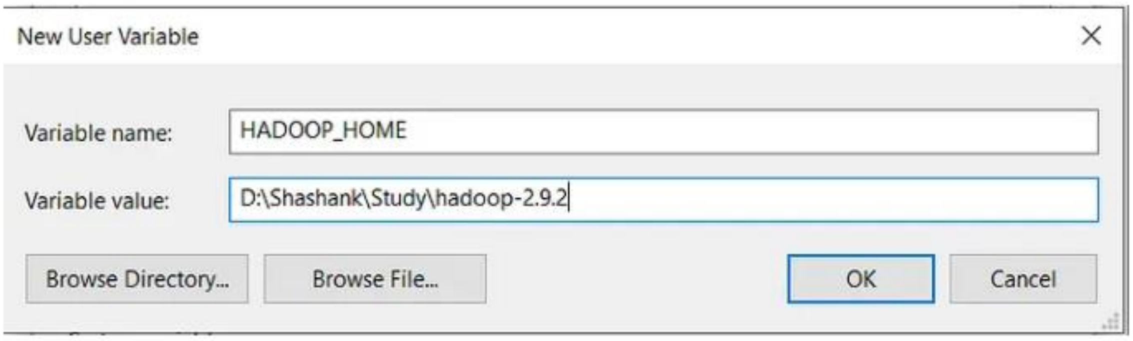
Variable	Value
FLUME_CONF	D:\Shashank\Study\Flume\conf
FLUME_HOME	D:\Shashank\Study\Flume
HADOOP_HOME	D:\Shashank\Study\hadoop-2.9.2
HIVE_HOME	D:\Shashank\Study\Hive-3.1.2
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_221
OneDrive	C:\Users\shash\OneDrive
OneDriveConsumer	C:\Users\shash\OneDrive

System variables

Variable	Value
PROCESSOR_REVISION	8e0c
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules; C:\WINDOW...
RegionCode	APJ
TEMP	C:\WINDOWS\TEMP
TMP	C:\WINDOWS\TEMP
USERNAME	SYSTEM
windir	C:\WINDOWS

Fig. 9:- Adding Environment Variable

On clicking "New", we get below screen.



New User Variable

Variable name: HADOOP_HOME

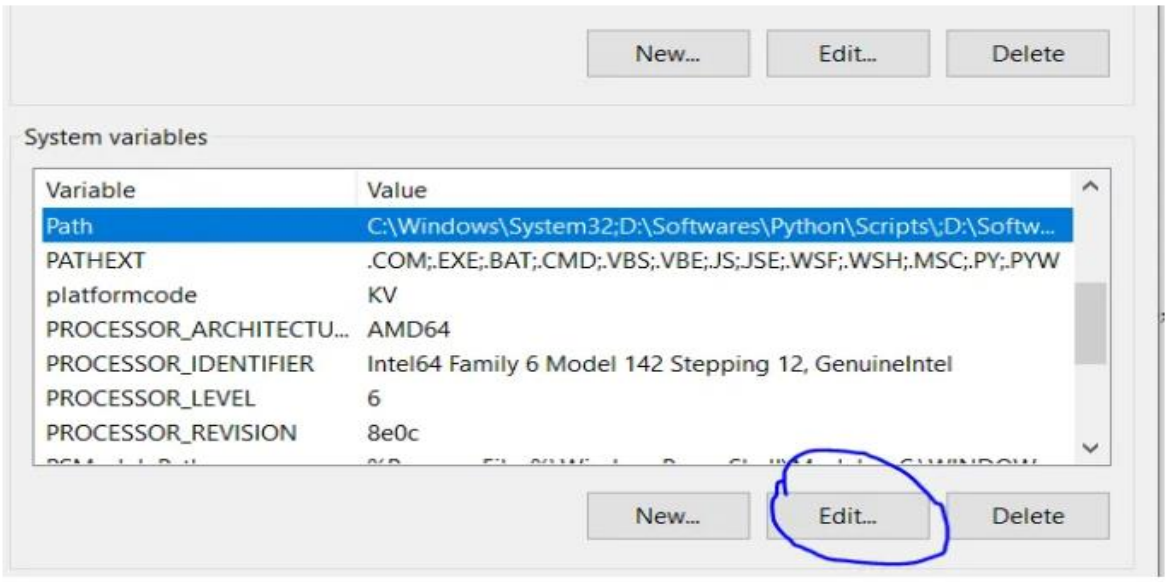
Variable value: D:\Shashank\Study\hadoop-2.9.2

Fig. 10:- Adding HADOOP_HOME

Now as shown, add HADOOP_HOME in variable name and path of Hadoop folder in Variable Value.
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

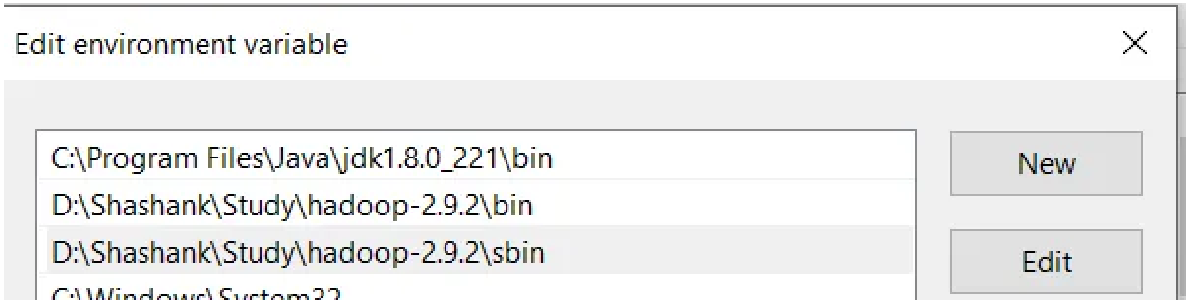
Last step in setting Environment variable is setting Path in System Variable.



Variable	Value
Path	C:\Windows\System32;D:\Softwares\Python\Scripts\;D:\Softw...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
platformcode	KV
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0c

Fig. 11:- Setting Path Variable

Select Path variable in the system variables and click on “Edit”.



```
C:\Program Files\Java\jdk1.8.0_221\bin
D:\Shashank\Study\hadoop-2.9.2\bin
D:\Shashank\Study\hadoop-2.9.2\sbin
```

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

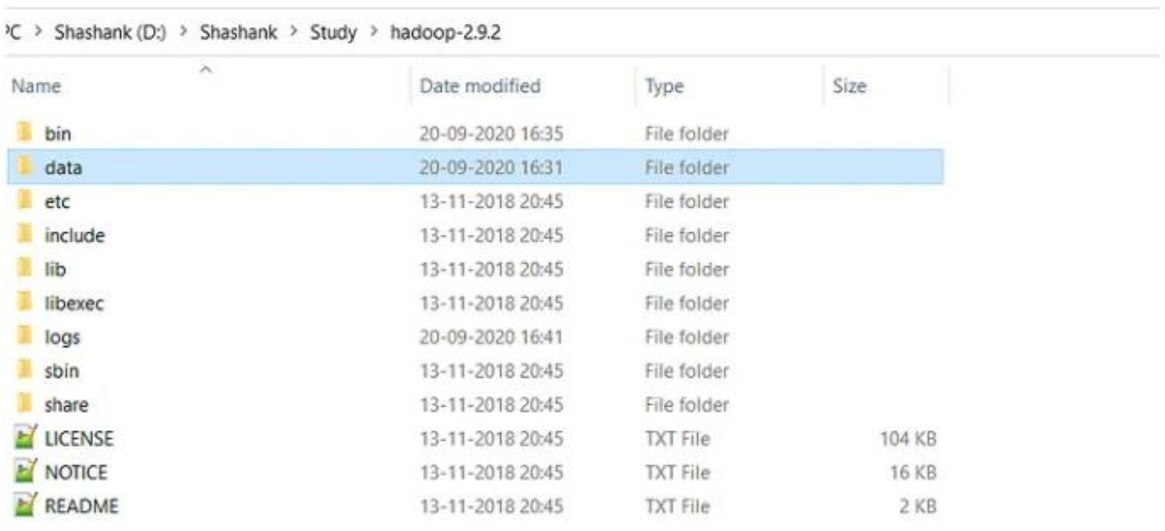
```
echo %JAVA_HOME%
echo %HADOOP_HOME%
echo %PATH%
```

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

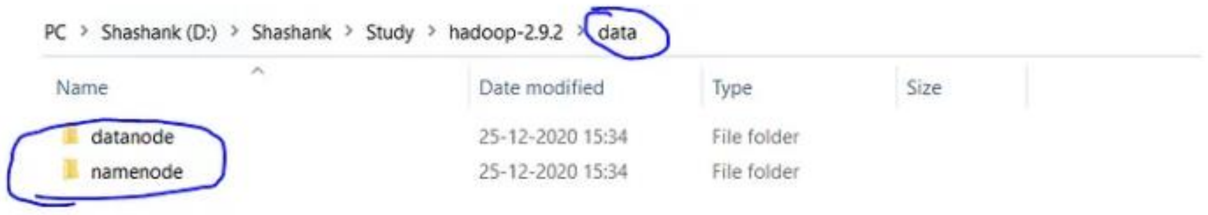
We need to create a folder data in the hadoop directory, and 2 sub folders namenode and datanode



```
PC > Shashank (D:) > Shashank > Study > hadoop-2.9.2

Name        Date modified       Type         Size
bin         20-09-2020 16:35    File folder
data        20-09-2020 16:31    File folder
etc         13-11-2018 20:45    File folder
include     13-11-2018 20:45    File folder
lib         13-11-2018 20:45    File folder
libexec     13-11-2018 20:45    File folder
logs        20-09-2020 16:41    File folder
sbin        13-11-2018 20:45    File folder
share       13-11-2018 20:45    File folder
LICENSE     13-11-2018 20:45    TXT File     104 KB
NOTICE      13-11-2018 20:45    TXT File      16 KB
README      13-11-2018 20:45    TXT File       2 KB
```

Fig. 13:- Creating Data Folder

Create DATA folder in the Hadoop directory



```
PC > Shashank (D:) > Shashank > Study > hadoop-2.9.2 > data

Name        Date modified       Type         Size
datanode    25-12-2020 15:34    File folder
namenode    25-12-2020 15:34    File folder
```

Fig. 14:- Creating Sub-folders

Once DATA folder is created, we need to create 2 new folders namely, namenode and datanode inside 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..!!!!!

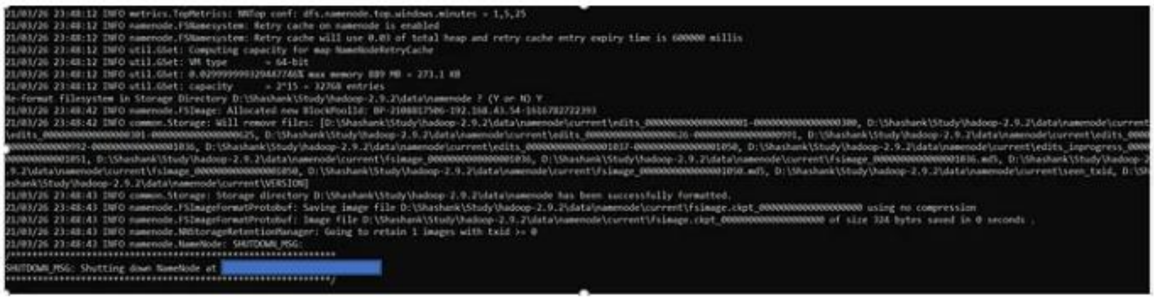
We are done with the setting up the Hadoop in our System.

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 Command Prompt and run below command

hadoop namenode –format



```
21/03/26 23:48:12 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
21/03/26 23:48:12 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
21/03/26 23:48:12 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
21/03/26 23:48:12 INFO util.GSet: Computing capacity for map NameNodeRetryCache
21/03/26 23:48:12 INFO util.GSet: VM type       = 64-bit
21/03/26 23:48:12 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
21/03/26 23:48:12 INFO util.GSet: capacity      = 2^15 = 32768 entries
Re-format filesystem in Storage Directory D:\Shashank\Study\hadoop-2.9.2\data\namenode ? (Y or N) Y
21/03/26 23:48:42 INFO namenode.FSImage: Allocated new BlockPoolId: BP-2108817506-192.168.43.54-1616782722393
21/03/26 23:48:42 INFO common.Storage: Will remove files: [D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\edits_0000000000000000001-0000000000000000300, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current
\edits_0000000000000000301-0000000000000000625, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\edits_0000000000000000626-0000000000000000991, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\edits_0000
000000000000992-0000000000000001036, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\edits_0000000000000001037-0000000000000001050, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\edits_inprogress_0000
000000000001051, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\fsimage_0000000000000001036, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\fsimage_0000000000000001036.md5, D:\Shashank\Study\hadoop-2
.9.2\data\namenode\current\fsimage_0000000000000001050, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\fsimage_0000000000000001050.md5, D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\seen_txid, D:\Sh
ashank\Study\hadoop-2.9.2\data\namenode\current\VERSION]
21/03/26 23:48:43 INFO common.Storage: Storage directory D:\Shashank\Study\hadoop-2.9.2\data\namenode has been successfully formatted.
21/03/26 23:48:43 INFO namenode.FSImageFormatProtobuf: Saving image file D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\fsimage.ckpt_0000000000000000000 using no compression
21/03/26 23:48:43 INFO namenode.FSImageFormatProtobuf: Image file D:\Shashank\Study\hadoop-2.9.2\data\namenode\current\fsimage.ckpt_0000000000000000000 of size 324 bytes saved in 0 seconds .
21/03/26 23:48:43 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
21/03/26 23:48:43 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at [redacted]
************************************************************/
```

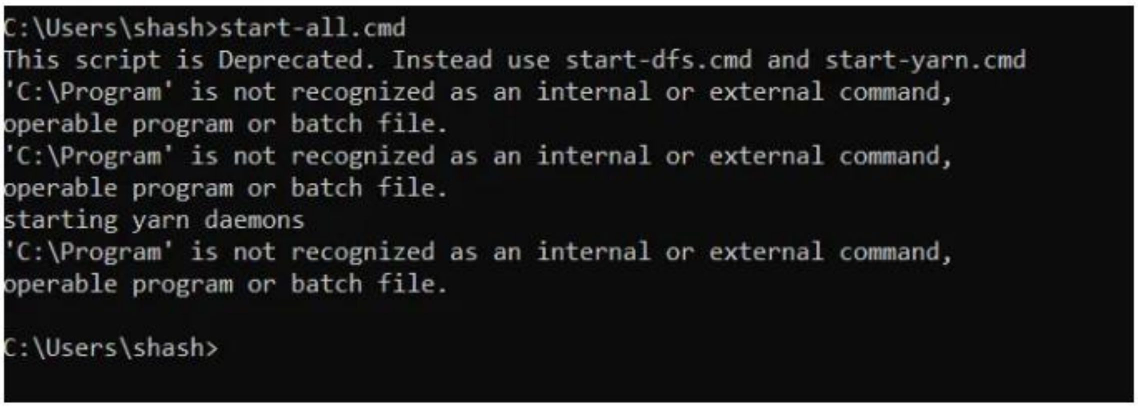
Fig. 15:- Formatting Namenode

Note:- This command formats all the data in namenode. So, its advisable to use only at the start and do 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 avoid permission issues and execute below commands

start-all.cmd

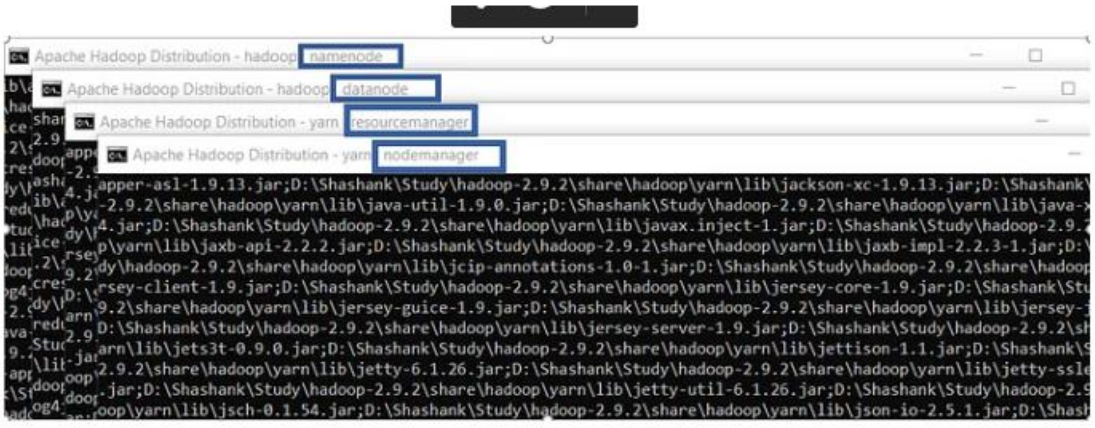


```
C:\Users\shash>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
starting yarn daemons
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\shash>
```

Fig. 16:- start-all.cmd

This will open 4 new cmd windows running 4 different Daemons of hadoop:-

* Namenode
* Datanode
* Resourcemanager
* Nodemanager



Apache Hadoop Distribution - hadoop namenode

Apache Hadoop Distribution - hadoop datanode

Apache Hadoop Distribution - yarn resourcemanager

Apache Hadoop Distribution - yarn nodemanager

```
apper-asl-1.9.13.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jackson-xc-1.9.13.jar;D:\Shashank\
-2.9.2\share\hadoop\yarn\lib\java-util-1.9.0.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\java-x
4.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\javax.inject-1.jar;D:\Shashank\Study\hadoop-2.9.
p\yarn\lib\jaxb-api-2.2.2.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jaxb-impl-2.2.3-1.jar;D:\
dy\hadoop-2.9.2\share\hadoop\yarn\lib\jcip-annotations-1.0-1.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop
rsey-client-1.9.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jersey-core-1.9.jar;D:\Shashank\Stu
9.2\share\hadoop\yarn\lib\jersey-guice-1.9.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jersey-
D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jersey-server-1.9.jar;D:\Shashank\Study\hadoop-2.9.2\sh
arn\lib\jets3t-0.9.0.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jettison-1.1.jar;D:\Shashank\S
2.9.2\share\hadoop\yarn\lib\jetty-6.1.26.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jetty-ssle
.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\jetty-util-6.1.26.jar;D:\Shashank\Study\hadoop-2.9
oop\yarn\lib\jsch-0.1.54.jar;D:\Shashank\Study\hadoop-2.9.2\share\hadoop\yarn\lib\json-io-2.5.1.jar;D:\Shash
```

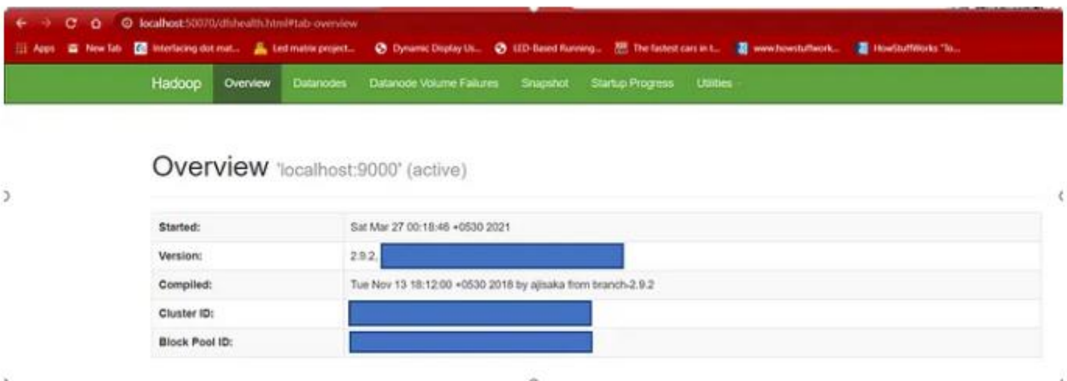
Fig. 17:- Hadoop Deamons

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.



localhost:50070/dfshealth.html#tab-overview

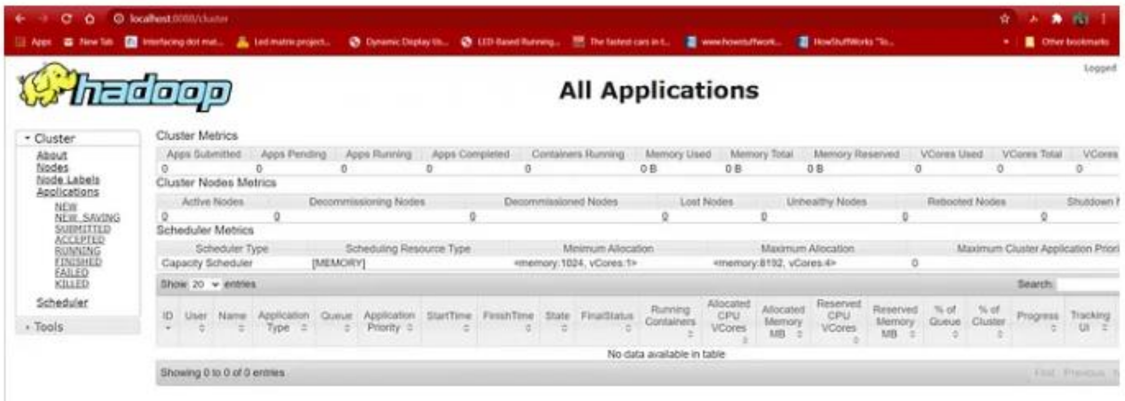
Overview 'localhost:9000' (active)

Started:	Sat Mar 27 00:18:46 +0530 2021
Version:	2.9.2, [redacted]
Compiled:	Tue Nov 13 18:12:00 +0530 2018 by ajisaka from branch-2.9.2
Cluster ID:	[redacted]
Block Pool ID:	[redacted]

Fig. 18:- Namenode Web UI

6.2 Resourcemanger

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



localhost:8088/cluster

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores
0	0	0	0	0	0 B	0 B	0 B	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown N
0	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priori
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

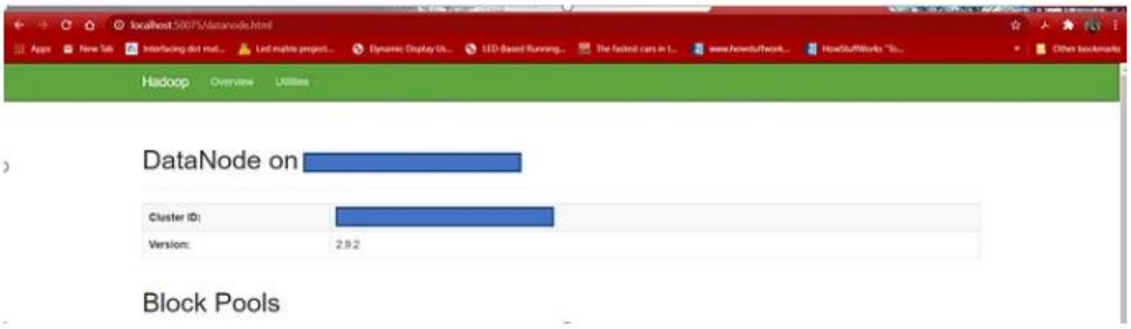
ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI
No data available in table																		

Showing 0 to 0 of 0 entries

Fig. 19:- Resourcemanager Web UI

6.3 Datanode

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



DataNode on [redacted]

Cluster ID:	[redacted]
Version:	2.9.2

Block Pools

Fig. 20:- Datanode Web UI

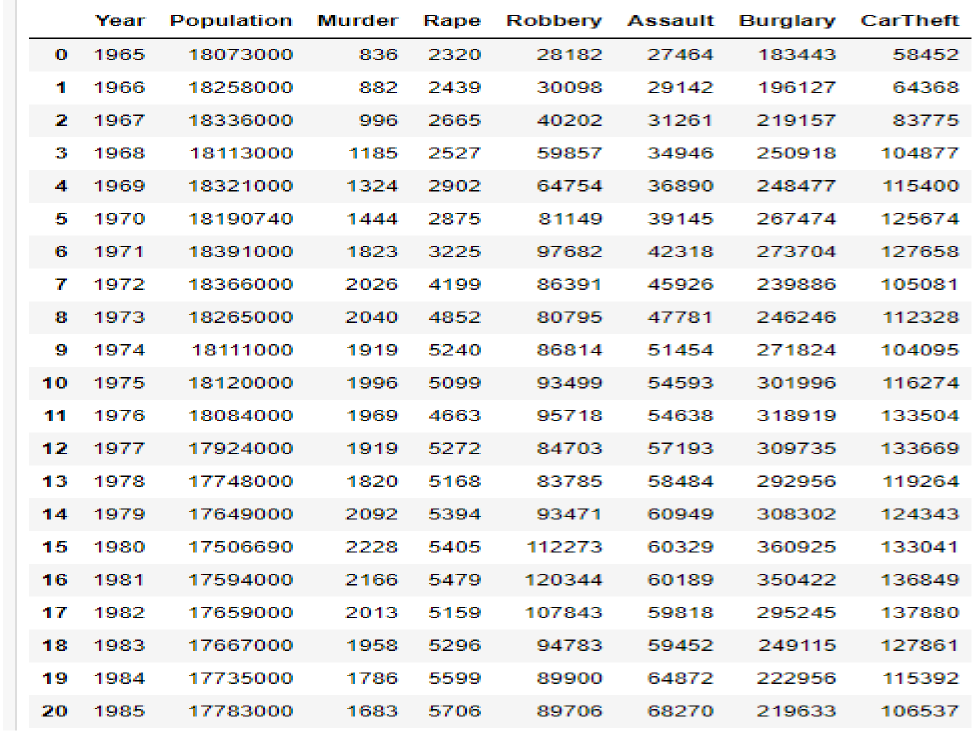
EXPERIMENT: 3

Visualize Data Using Basic Plotting Techniques In Python.

PROGRAM:

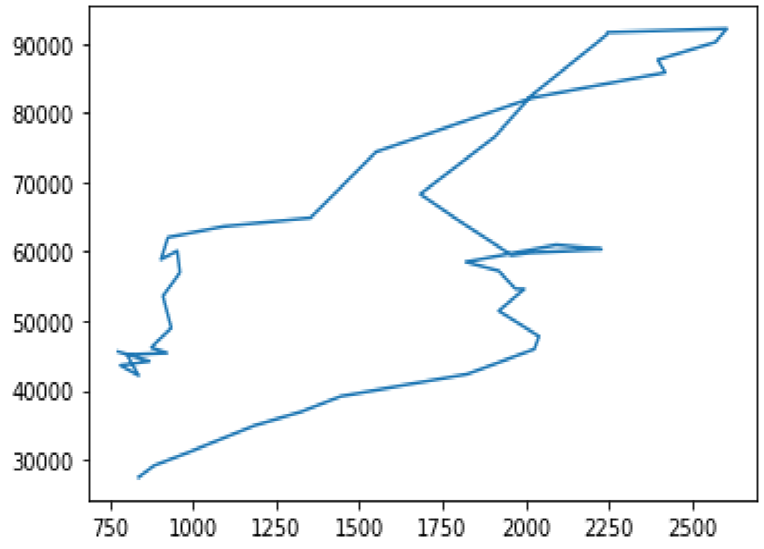
AIM: To create an application that takes the Visualize Data Using Basic Plotting Techniques.

```
import pandas as pb
import matplotlib.pyplot as plt
import seaborn as sns
crime=pb.read_csv('crime.csv')
crime
```



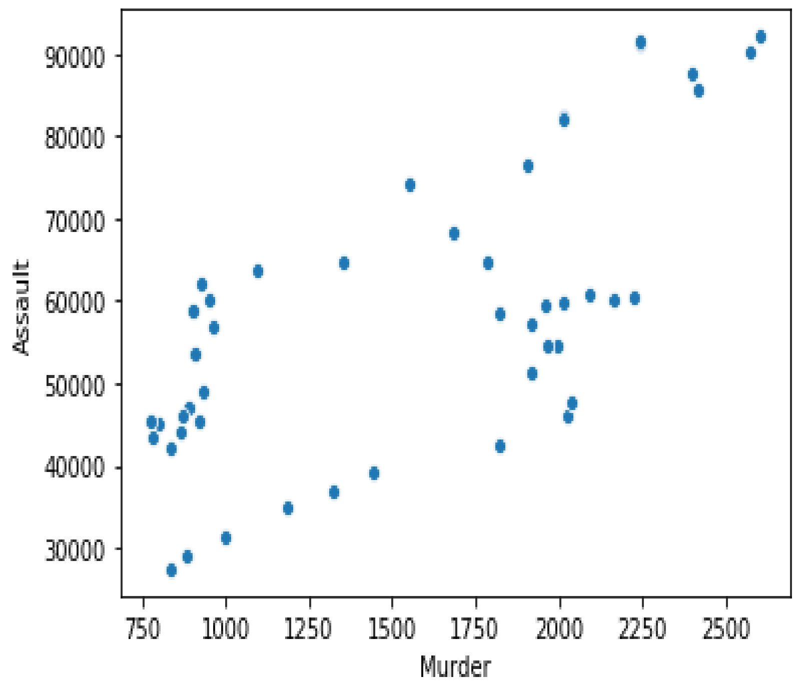
```
    Year  Population  Murder  Rape  Robbery  Assault  Burglary  CarTheft
0   1965    18073000     836  2320    28182    27464    183443     58452
1   1966    18258000     882  2439    30098    29142    196127     64368
2   1967    18336000     996  2665    40202    31261    219157     83775
3   1968    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
```

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

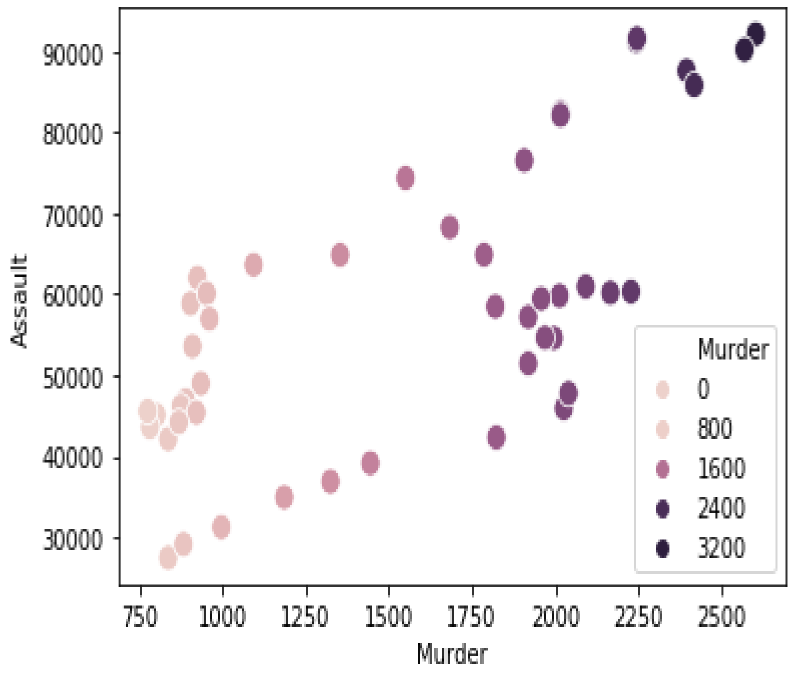


```
import seaborn as sns
```

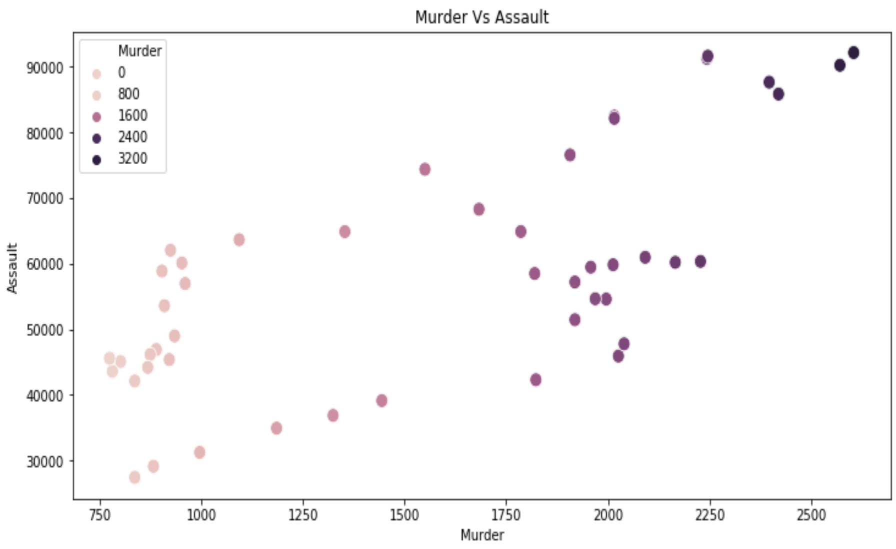
```
sns.scatterplot(crime.Murder,crime.Assault);
```



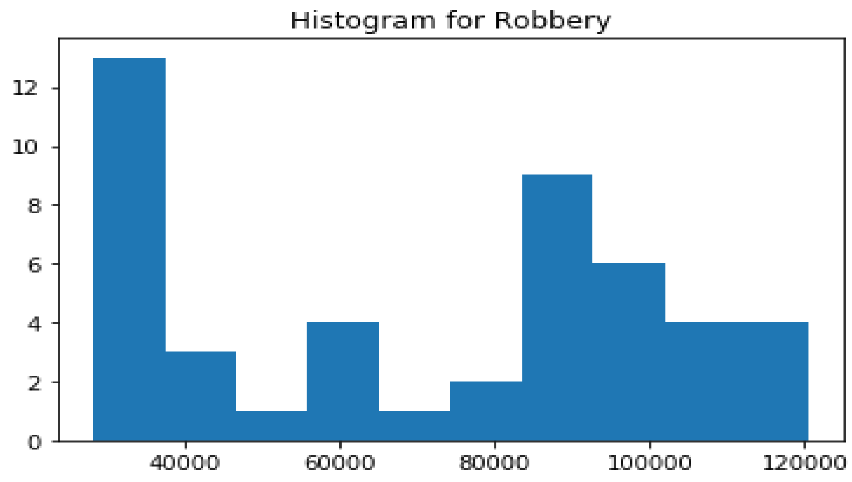
```
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



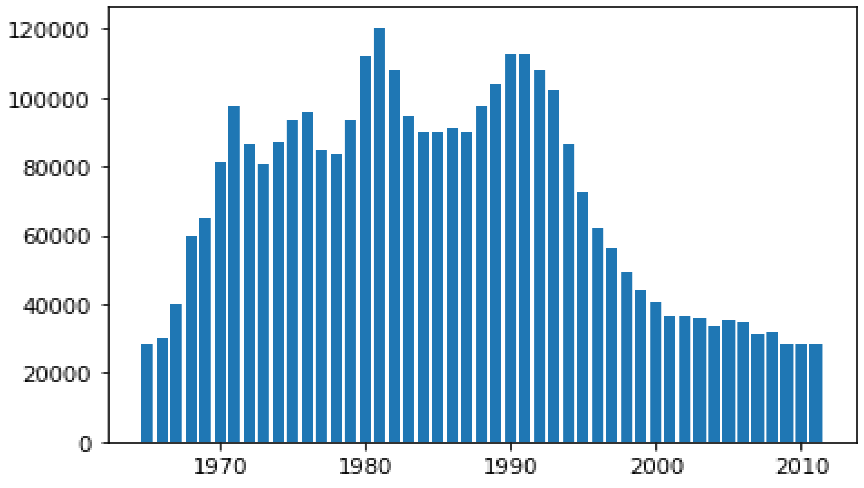
```
plt.figure(figsize=(12,6))
plt.title('Murder Vs Assault')
sns.scatterplot(crime.Murder,crime.Assault,hue=crime.Murder,s=100);
```



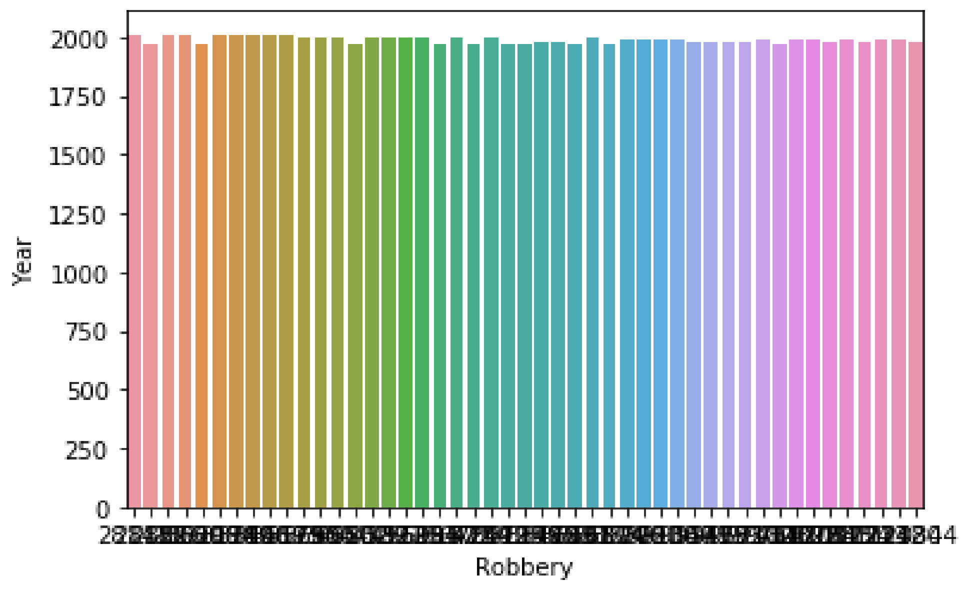
plt.title('Histogram for Robbery')
plt.hist(crime.Robbery);



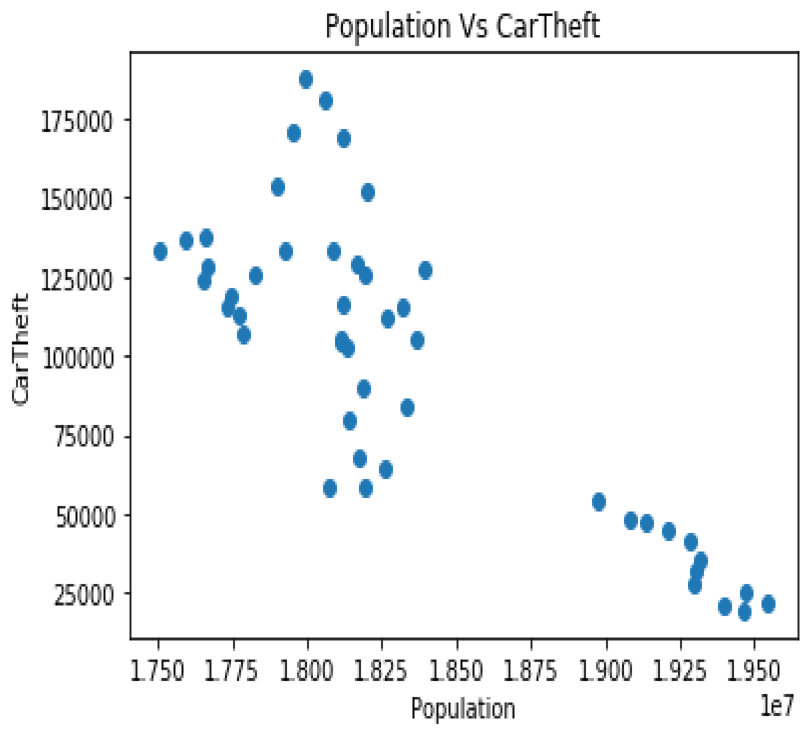
plt.bar(crime_bar.index,crime_bar.Robbery);



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();
```



EXPERIMENT: 4

Implement no sql Database Operations: Crud Operations, Arrays Using MONGODB.

PROGRAM:

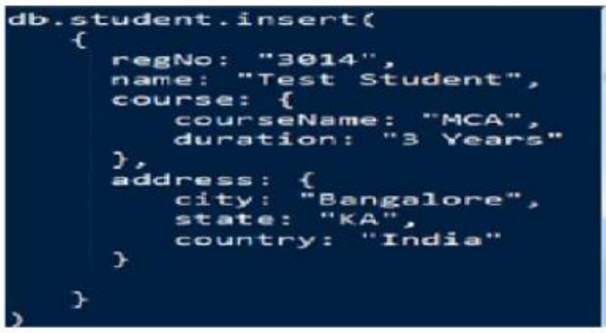
AIM: To Create a operations for crud and arrays without no sql datasbase.

TITLE: Basic CRUD operations in MongoDB.

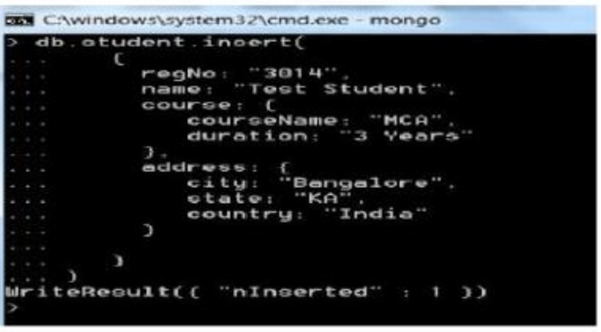
CRUD operations refer to the basic Insert, Read, Update and Delete operations.
Inserting a document into a collection (Create)
➢ The command db.collection.insert()will perform an insert operation into a collection of a document. ➢ Let us insert a document to a student collection. You must be connected to a database for doing any insert. It is done as follows:

```
db.student.insert({
regNo: "3014",
 name: "Test Student",
course: { courseName: "MCA", duration: "3 Years" },
address: {
city: "Bangalore",
state: "KA",
country: "India" } })
```

An entry has been made into the collection called student.

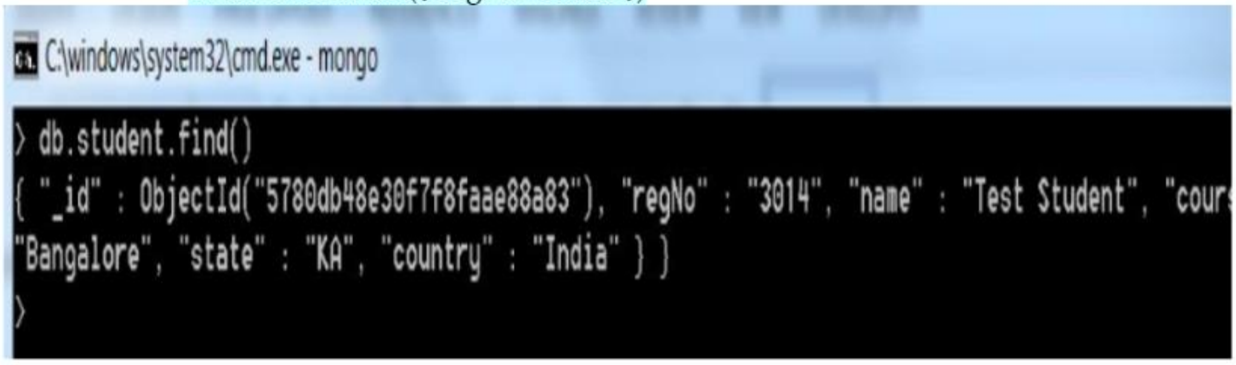


```
db.student.insert(
    {
        regNo: "3014",
        name: "Test Student",
        course: {
            courseName: "MCA",
            duration: "3 Years"
        },
        address: {
            city: "Bangalore",
            state: "KA",
            country: "India"
        }

    }
)
```



```
C:\windows\system32\cmd.exe - mongo
> db.student.insert(
...     {
...         regNo: "3014",
...         name: "Test Student",
...         course: {
...             courseName: "MCA",
...             duration: "3 Years"
...         },
...         address: {
...             city: "Bangalore",
...             state: "KA",
...             country: "India"
...         }
...
...     }
... )
WriteResult({ "nInserted" : 1 })
>
```

Querying a document from a collection (Read)
To retrieve (Select) the inserted document, run the below command. The find() command will retrieve all the documents of the given collection.
db.collection_name.find()
➢ If a record is to be retrieved based on some criteria, the find() method should be called passing parameters, then the record will be retrieved based on the attributes specified.
db.collection_name.find({"fieldname":"value"})
➢ For Example: Let us retrieve the record from the student collection where the attribute regNo is 3014and the query for the same is as shown below:
db.students.find({"regNo":"3014"})



```
C:\windows\system32\cmd.exe - mongo
> db.student.find()
{ "_id" : ObjectId("5780db48e30f7f8faae88a83"), "regNo" : "3014", "name" : "Test Student", "cour
"Bangalore", "state" : "KA", "country" : "India" } }
>
```

Updating a document in a collection (Update) In order to update specific field values of a collection in MongoDB, run the below query. db.collection_name.update()

➢ update() method specified above will take the fieldname and the new value as argument to update a document.

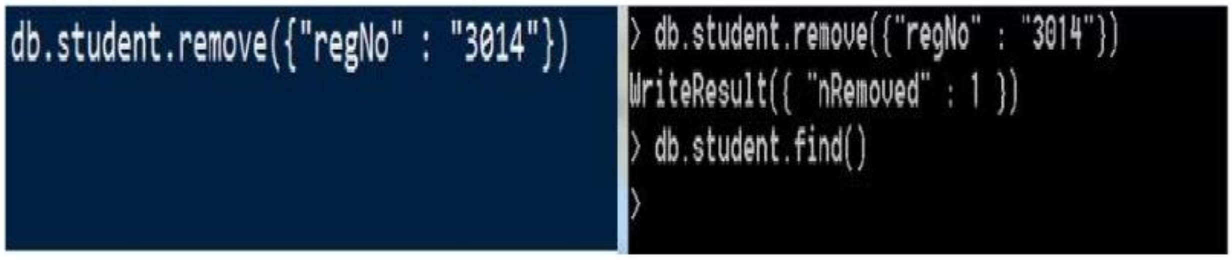
➢ Let us update the attribute name of the collection student for the document with regNo 3014.

```
db.student.update({
 "regNo": "3014"
 },
$set:
{
 "name": "Viraj"
})
```

Removing an entry from the collection (Delete)

➢ Let us now look into the deleting an entry from a collection. In order to delete an entry from a collection, run the command as shown below : db.collection_name.remove({"fieldname":"value"})

➢ For Example : db.student.remove({"regNo":"3014"})



```
db.student.remove({"regNo" : "3014"})
```

```
> db.student.remove({"regNo" : "3014"})
WriteResult({ "nRemoved" : 1 })
> db.student.find()
>
```

Note that after running the remove() method, the entry has been deleted from the student collection.

Working with Arrays in MongoDB

1. Introduction

In a MongoDB database, data is stored in collections and a collection has documents. A document has fields and values, like in a JSON. The field types include scalar types (string, number, date, etc.) and composite types (arrays and objects). In this article we will look at an example of using the array field type.

The example is an application where users create blog posts and write comments for the posts. The relationship between the posts and comments is One-to-Many; i.e., a post can have many comments. We will consider a collection of blog posts with their comments. That is a post document will also store the related comments. In MongoDB's document model, a 1:N relationship data can be stored within a collection; this is a de-normalized form of data. The related data is stored together and can be accessed (and updated) together. The comments are stored as an array; an array of comment objects.

A sample document of the blog posts with comments:

```
{
        "_id" : ObjectId("5ec55af811ac5e2e2aafb2b9"),
        "name" : "Working with Arrays",
        "user" : "Database Rebel",
        "desc" : "Maintaining an array of objects in a document",
        "content" : "some content ...",
        "created" : ISODate("2020-05-20T16:28:55.468Z"),
        "updated" : ISODate("2020-05-20T16:28:55.468Z"),
        "tags" : [ "mongodb", "arrays" ],
        "comments" : [
                {
                        "user" : "DB Learner",
                "content" : "Nice post.",
                "updated" : ISODate("2020-05-20T16:35:57.461Z")
            }
        ]
}
```

In an application, a blog post is created, comments are added, queried, modified or deleted by users. In the example, we will write code to create a blog post document, and do some CRUD operations with comments for the post.

2. Create and Query a Document

Let's create a blog post document. We will use a database called as blogs and a collection called as posts. The code is written in mongoshell (an interactive JavaScript interface to MongoDB). Mongo shell is started from the command line and is connected to the MongoDB server. From the shell:

```
use blogs
NEW_POST =
  {
    name: "Working with Arrays",
    user: "Database Rebel",
    desc: "Maintaining an array of objects in a document",
    content: "some content...",
    created: ISODate(),
    updated: ISODate(),
    tags: [ "mongodb", "arrays" ]
}
db.posts.insertOne(NEW_POST)
```

Returns a result { "acknowledged" : true, "insertedId" : ObjectId("5ec55af811ac5e2e2aafb2b9") } indicating that a new document is created. This is a common acknowledgement when you perform a write operation. When a document is inserted into a collection for the first time, the collection gets created (if it doesn't exist already). The insertOne method inserts a document into the collection.
Now, let's query the collection :

```
db.posts.findOne()
{
        "_id" : ObjectId("5ec55af811ac5e2e2aafb2b9"),
        "name" : "Working with Arrays",
        "user" : "Database Rebel",
        "desc" : "Maintaining an array of objects in a document",
        "content" : "some content...",
        "created" : ISODate("2020-05-20T16:28:55.468Z"),
        "updated" : ISODate("2020-05-20T16:28:55.468Z"),
        "tags" : [
                "mongodb",
                "arrays"
        ]
}
```

The findOne method retrieves one matching document from the collection. Note the scalar fields name (string type) and created (date type), and the array field tags. In the newly inserted document there are no comments, yet.

3. Add an Array Element

Let's add a comment for this post, by a user "DB Learner":

```
NEW_COMMENT = {
  user: "DB Learner",
  text: "Nice post, can I know more about the arrays in MongoDB?",
  updated: ISODate()
}
db.posts.updateOne(
  { _id : ObjectId("5ec55af811ac5e2e2aafb2b9") },
  { $push: { comments: NEW_COMMENT } }
)
```

Returns: { "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
The updateOne method updates a document's fields based upon the specified condition. $push is an array update operator which adds an element to an array. If the array doesn't exist, it creates an array field and then adds the element.
Let's query the collection and confirm the new comment visually, using the findOne method:

```
{
        "_id" : ObjectId("5ec55af811ac5e2e2aafb2b9"),
        "name" : "Working with Arrays",
        ...
        "comments" : [
                {
                        "user" : "DB Learner",
                        "text" : "Nice post, can I know more about the arrays in MongoDB?",
                        "updated" : ISODate("2020-05-20T16:35:57.461Z")
                }
        ]
}
```

Note the comments array field has comment objects as elements. Let's add one more comment using the same $push update operator. This new comment (by user "Database Rebel") is appended to the comments array:

```
"comments" : [
        {
           "user" : "DB Learner",
           "text" : "Nice post, can I know more about the arrays in MongoDB?",
           "updated" : ISODate("2020-05-20T16:35:57.461Z")
        },
        {
           "user" : "Database Rebel",
           "text" : "Thank you, please look for updates",
           "updated" : ISODate("2020-05-20T16:48:25.506Z")
        }
]
```

4. Update an Array Element

Let's update the comment posted by "Database Rebel" with modified text field :
NEW_CONTENT = "Thank you, please look for updates - updated the post".

```
db.posts.updateOne(
  { _id : ObjectId("5ec55af811ac5e2e2aafb2b9"), "comments.user": "Database Rebel" },
  { $set: { "comments.$.text": NEW_CONTENT } }
)
```

The $set update operator is used to change a field's value. The positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array. The first matching element is updated. The updated comment object:

```
"comments" : [
        {
            "user" : "Database Rebel",
            "text" : "Thank you, please look for updates - updated",
            "updated" : ISODate("2020-05-20T16:48:25.506Z")
        }
]
```

5. Delete an Array Element

The user changed his mind and wanted to delete the comment, and then add a new one.

```
db.posts.updateOne(
  { _id" : ObjectId("5ec55af811ac5e2e2aafb2b9") },
  { $pull: { comments: { user: "Database Rebel" } } }
)
```

The $pull update operator removes elements from an array which match the specified condition - in this case { comments: { user: "Database Rebel" } }.
A new comment is added to the array after the above delete operation, with the following text: "Thank you for your comment. I have updated the post with CRUD operations on an array field".

6. Add a New Field to all Objects in the Array

Let's add a new field likes for all the comments in the array.

```
db.posts.updateOne(
  { "_id : ObjectId("5ec55af811ac5e2e2aafb2b9") },
  { $set: { "comments.$[].likes": 0 } }
)
```

The all positional operator $[] specifies that the update operator $set should modify all elements in the specified array field. After the update, all comment objects have the likes field, for example:

```
{
   "user" : "DB Learner",
   "text" : "Nice post, can I know more about the arrays in MongoDB?",
   "updated" : ISODate("2020-05-20T16:35:57.461Z"),
   "likes" : 0
}
```

7. Update a Specific Array Element Based on a Condition

First, let's add another new comment using the $push update operator:

```
NEW_COMMENT = {
  user: "DB Learner",
  text: "Thanks for the updates!",
  updated: ISODate()
}
```

Note the likes field is missing in the input document. We will update this particular comment in the comments array with the condition that the likes field is missing.

```
db.posts.updateOne(
  { "_id" : ObjectId("5ec55af811ac5e2e2aafb2b9") },
  { $inc: { "comments.$[ele].likes": 1 } },
  { arrayFilters: [ { "ele.user": "DB Learner",  "ele.likes": { $exists: false } } ] }
)
```

The likes field is updated using the $inc update operator (this increments a field's value, or if not exists adds the field and then increments). The filtered positional operator $[<identifier>] identifies the array elements that match the arrayFilters conditions for an update operation.

EXPERIMENT: 5

Implement Functions: Count – Sort – Limit – Skip – Aggregate Using MONGODB.

PROGRAM:

AIM: To create function operations for sort, limit, skip and aggregate.

1. COUNT

How do you get the number of Debit and Credit transactions? One way to do it is by using count() function as below

```
> db.transactions.count({cr_dr : "D"});
                                    or

> db.transactions.find({cr_dr : "D"}).length();
```

But what if you do not know the possible values of cr_dr upfront. Here Aggregation framework comes to play. See the below Aggregate query.

```
> db.transactions.aggregate(
     [
        {
           $group : {
              _id : '$cr_dr', // group by type of transaction
            // Add 1 for each document to the count for this type of
transaction
              count : {$sum : 1}
           }
        }
     ]
  );
```

And the result is

```
{
   "_id" : "C",
   "count" : 3
}
{
   "_id" : "D",
   "count" : 5
}
```

2. SORT

Definition

$sort

Sorts all input documents and returns them to the pipeline in sorted order.

The

$sort

stage has the following prototype form:

```
{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
```

$sort
takes a document that specifies the field(s) to sort by and the respective sort order. <sort order> can have one of the following values:

Value	Description
1	Sort ascending.
-1	Sort descending.
{ $meta: "textScore" }	Sort by the computed textScore metadata in descending order. See Text Score Metadata Sort for an example.

If sorting on multiple fields, sort order is evaluated from left to right. For example, in the form above, documents are first sorted by <field1>. Then documents with the same <field1> values are further sorted by <field2>.

Behavior

Limits

You can sort on a maximum of 32 keys.

Sort Consistency

MongoDB does not store documents in a collection in a particular order. When sorting on a field which contains duplicate values, documents containing those values may be returned in any order.

If consistent sort order is desired, include at least one field in your sort that contains unique values. The easiest way to guarantee this is to include the _id field in your sort query.

Consider the following restaurant collection:

```
db.restaurants.insertMany( [
   { "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan"},
   { "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" :
"Queens"},
   { "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn"},
   { "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan"},
   { "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn"},
] )
```

The following command uses the
$sort
stage to sort on the borough field:

```
db.restaurants.aggregate(
   [
     { $sort : { borough : 1 } }
   ]
)
```

In this example, sort order may be inconsistent, since the borough field contains duplicate values for both Manhattan and Brooklyn. Documents are returned in alphabetical order by borough, but the order of those documents with duplicate values for borough might not the be the same across multiple executions of the same sort. For example, here are the results from two different executions of the above command:

```
{ "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn" }
{ "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn" }
{ "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan" }
{ "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan" }
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens"
}
{ "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn" }
{ "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn" }
{ "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan" }
{ "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan" }
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens"
}
```

While the values for borough are still sorted in alphabetical order, the order of the documents containing duplicate values for borough (i.e. Manhattan and Brooklyn) is not the same.

To achieve a consistent sort, add a field which contains exclusively unique values to the sort. The following command uses the
$sort
stage to sort on both the borough field and the _id field:

```
db.restaurants.aggregate(
   [
     { $sort : { borough : 1, _id: 1 } }
   ]
)
```

Since the _id field is always guaranteed to contain exclusively unique values, the returned sort order will always be the same across multiple executions of the same sort.

Examples
Ascending/Descending Sort
For the field or fields to sort by, set the sort order to 1 or -1 to specify an ascending or descending sort respectively, as in the following example:

```
db.users.aggregate(
   [
     { $sort : { age : -1, posts: 1 } }
   ]
)
```

This operation sorts the documents in the users collection, in descending order according by the age field and then in ascending order according to the value in the posts field.

3. LIMIT

$sort
Sorts all input documents and returns them to the pipeline in sorted order.
The $sort stage has the following prototype form:

```
        { $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
```

$sort takes a document that specifies the field(s) to sort by and the respective sort order. <sort order> can have one of the following values:

Value	Description
1	Sort ascending.
-1	Sort descending.
{ $meta: "textScore" }	Sort by the computed textScore metadata in descending order. See Text Score Metadata Sort for an example.

If sorting on multiple fields, sort order is evaluated from left to right. For example, in the form above, documents are first sorted by <field1>. Then documents with the same <field1> values are further sorted by <field2>.

Behavior

Limits
You can sort on a maximum of 32 keys.

Sort Consistency
MongoDB does not store documents in a collection in a particular order. When sorting on a field which contains duplicate values, documents containing those values may be returned in any order.

If consistent sort order is desired, include at least one field in your sort that contains unique values. The easiest way to guarantee this is to include the _id field in your sort query.

Consider the following restaurant collection:

```
db.restaurants.insertMany( [
{ "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan"},
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens"},
{ "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn"},
{ "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan"},
{ "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn"},
] )
```

The following command uses the $sort stage to sort on the borough field:

```
db.restaurants.aggregate(
[
{ $sort : { borough : 1 } }
]
)
```

In this example, sort order may be inconsistent, since the borough field contains duplicate values for both Manhattan and Brooklyn. Documents are returned in alphabetical order by borough, but the order of those documents with duplicate values for borough might not the be the same across multiple executions of the same sort. For example, here are the results from two different executions of the above command:

```
{ "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn" }
{ "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn" }
{ "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan" }
{ "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan" }
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens" }

{ "_id" : 5, "name" : "Jane's Deli", "borough" : "Brooklyn" }
{ "_id" : 3, "name" : "Empire State Pub", "borough" : "Brooklyn" }
{ "_id" : 4, "name" : "Stan's Pizzaria", "borough" : "Manhattan" }
{ "_id" : 1, "name" : "Central Park Cafe", "borough" : "Manhattan" }
{ "_id" : 2, "name" : "Rock A Feller Bar and Grill", "borough" : "Queens" }
```

While the values for borough are still sorted in alphabetical order, the order of the documents containing duplicate values for borough (i.e. Manhattan and Brooklyn) is not the same.

To achieve a *consistent sort,* add a field which contains exclusively unique values to the sort. The following command uses the $sort stage to sort on both the borough field and the _id field:

```
db.restaurants.aggregate(
[
{ $sort : { borough : 1, _id: 1 } }
]
)
```

Since the _id field is always guaranteed to contain exclusively unique values, the returned sort order will always be the same across multiple executions of the same sort.

Examples

Ascending/Descending Sort
For the field or fields to sort by, set the sort order to 1 or -1 to specify an ascending or descending sort respectively, as in the following example:

```
db.users.aggregate(
[
{ $sort : { age : -1, posts: 1 } }
]
)
```

4. SKIP
$skip
Skips over the specified number of documents that pass into the stage and passes the remaining documents to the next stage in the pipeline.

The
$skip
stage has the following prototype form:

```
{ $skip: <positive 64-bit integer> }
```

EXPERIMENT: 6

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 code

```
from collections import defaultdict

# Sample input data
data = [
    "Hello world",
    "MapReduce is a programming model",
    "Hello MapReduce",
    "MapReduce example"
]

# Initialize a dictionary to hold word counts
word_counts = defaultdict(int)

# Map Function
def map_function(data):
    for line in data:
        words = line.split()
        for word in words:
            yield (word.lower(), 1)

# Reduce Function
def reduce_function(mapped_data):
    for word, count in mapped_data:
        word_counts[word] += count

# Map phase
mapped_data = list(map_function(data))

# Shuffle and Sort (usually handled by the MapReduce framework)

# Reduce phase
reduce_function(mapped_data)

# Output the word counts
for word, count in word_counts.items():
    print(f"{word}: {count}")
```

OUTPUT:

EXPERIMENT: 7

Implement a MapReduce Program that process a dataset.

AIM:

To create process dataset using map reduce functions.

PROGRAM:

```
import csv
import pickle

# Read data from data.csv and perform mapping
mapped_data = []

with open('crime.csv', 'r') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        for word in row:  # Split each line into words
            mapped_data.append((word, 1))  # Emit (word, 1) for each word

# Store the mapped data in shuffled.pkl
with open('shuffled.pkl', 'wb') as output_file:
    pickle.dump(mapped_data, output_file)

import pickle
from collections import defaultdict

# Read the mapped data from shuffled.pkl
with open('shuffled.pkl', 'rb') as input_file:
    mapped_data = pickle.load(input_file)

# Perform reducing (word count) operation
word_counts = defaultdict(int)
for word, count in mapped_data:
    word_counts[word] += count

# Print the word counts
for word, count in word_counts.items():
    print(f"{word}: {count}")
```

EXPERIMENT: 8

Implement Clustering Techniques using SPARK.

AIM: To create a clustering using SPARK.

PROGRAM:

```
from pyspark.sql import SparkSession
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import ClusteringEvaluator

# Start Spark session
spark = SparkSession.builder.appName("IrisKMeans").getOrCreate()

# Load the dataset
data = spark.read.csv("iris.csv", header=True, inferSchema=True)

# Print schema to confirm structure
data.printSchema()
data.show(5)

# Prepare features by excluding the label column
feature_cols = data.columns
if "label" in feature_cols:
    feature_cols.remove("label")
elif "species" in feature_cols:  # Common case for Iris
    feature_cols.remove("species")

# Assemble features
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
assembled_data = assembler.transform(data)

# Train KMeans
kmeans = KMeans(k=3, seed=1)
model = kmeans.fit(assembled_data)

# Predictions
predictions = model.transform(assembled_data)
predictions.select("prediction", "features").show(5)

# Evaluate with Silhouette Score
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(predictions)
print(f"Silhouette Score = {silhouette}")

# Optional: Show cluster centers
centers = model.clusterCenters()
print("Cluster Centers:")
for i, center in enumerate(centers):
    print(f"Cluster {i}: {center}")
```

EXPERIMENT : 9

Implement an application that stores Big Data in MONGODB / PIG Using Hadoop / R.

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

PROGRAM:

```
pip install pymongo

import pymongo

# Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["bigdata_db"]
collection = db["bigdata_collection"]

# Simulated big data
big_data = [{"_id": i, "data": f"Data point {i}"} for i in range(1, 1000000)]

# Insert big data into MongoDB
collection.insert_many(big_data)

# Query data from MongoDB
result = collection.find_one({"_id": 1})
print("Retrieved data from MongoDB:")
print(result)

# Close MongoDB connection
client.close()
```

Signature of the Faculty