UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

M.Sc. Computer Science – Semester I

NoSQL Technologies

JOURNAL

2023-2024

Seat No. _____

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the work entered in this journal was done in the University Department of Computer Science laboratory by Mr./Ms._____Seat No. _____ for the course of M.Sc. Computer Science - Semester I (CBCS) (Revised) during the academic year 2023- 2024 in a satisfactory manner.

_____ _____

Subject In-charge **Head of Department**

_____

External Examiner

INDEX

Practical - 01

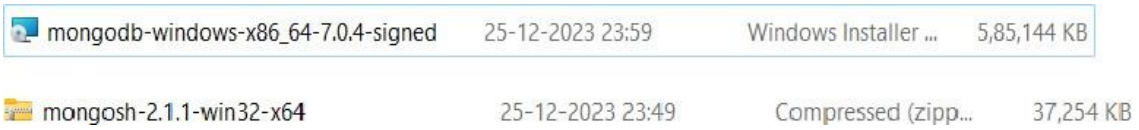
Aim: - Lab Exercise: Setting up and Exploring MongoDB

a) Install MongoDB on your local machine or lab server.
b) Create a new MongoDB database and collection.
c) Insert sample data into the collection.
d) Retrieve and display data from the collection using MongoDB queries.

Description: - MongoDB is a source-available, cross-platform, document-oriented database program. MongoDB is a NoSQL database product and uses JSON-like documents with optional schemas. It is non-relational and suitable for hierarchical data storage. It has a dynamic schema. In terms of performance, it is much faster than RDBMS.

a) Install MongoDB on your local machine or lab server.

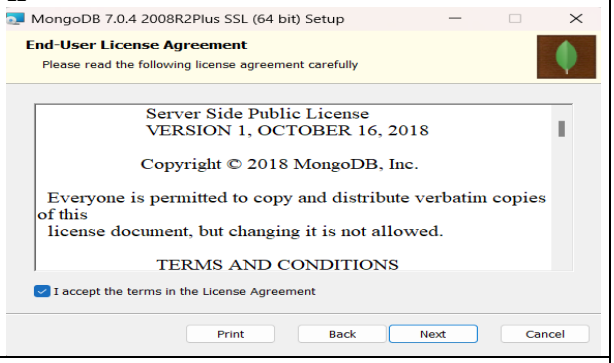
Step 1: - Download a MongoDB community server & MongoDB shell in your system.



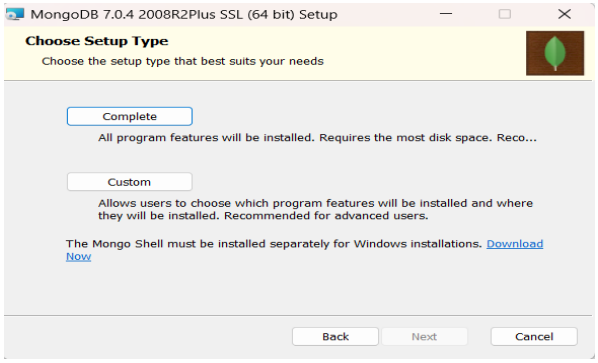
Step 2: - Set the MongoDB environment in your pc

I

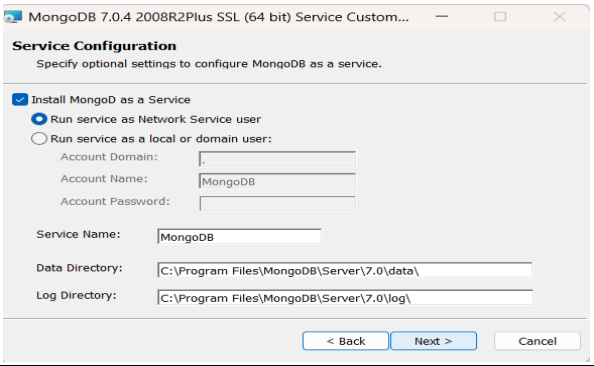
II



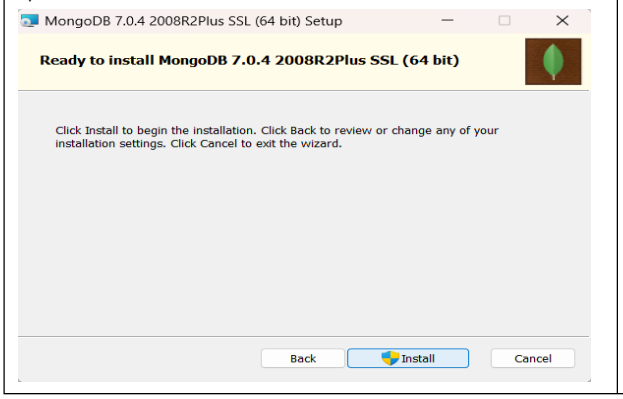
III



IV



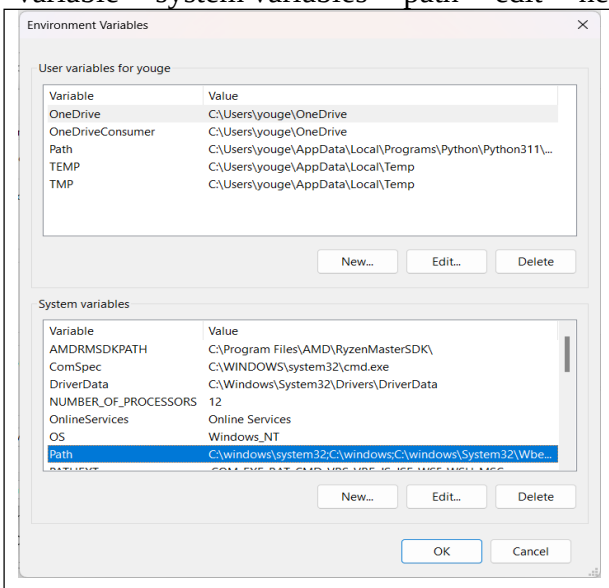
V

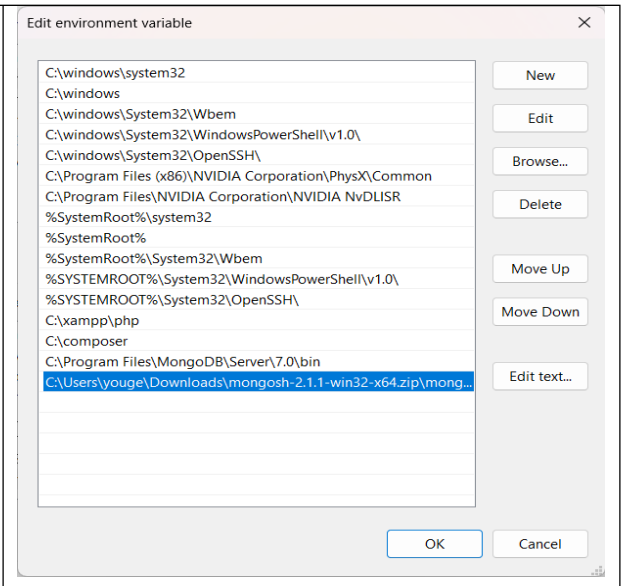


VI

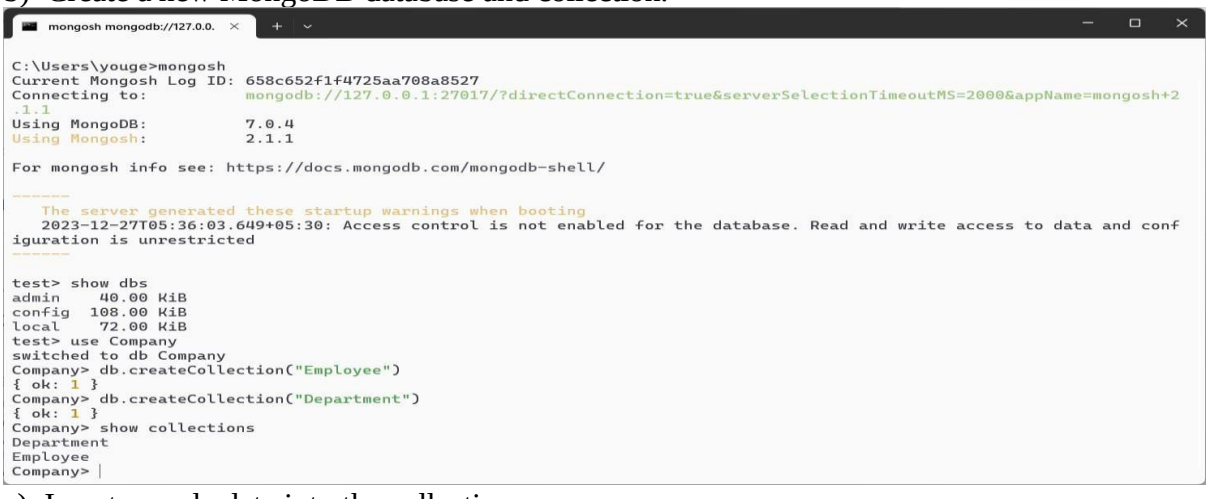
Step 3: -setting the path for the MongoDB

Step 4: - Copy the path of the MongoDB shell path and MongoDB server path and paste them into the environment variable>>system variables>>path>>edit>>new.





b) Create a new MongoDB database and collection.



c) Insert sample data into the collection.

```
Company> db.Department.insertMany([{ dept_id: 1, department: "IT", location: "Bangalore" },
... { dept_id: 2, department: "HR", location: "New York" },
... { dept_id: 3, department: "management", location: "Delhi" },
... { dept_id: 4, department: "Finance", location: "Mumbai" },
... { dept_id: 5, department: "sales", location: "Surat" }]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('658c714a1f4725aa708a8528'),
    '1': ObjectId('658c714a1f4725aa708a8529'),
    '2': ObjectId('658c714a1f4725aa708a852a'),
    '3': ObjectId('658c714a1f4725aa708a852b'),
    '4': ObjectId('658c714a1f4725aa708a852c')
  }
}
Company>
```

D) Retrieve and display data from the collection using MongoDB queries.

```
Company> db.Department.find()
[
  {
    _id: ObjectId('658c714a1f4725aa708a8528'),
    dept_id: 1,
    department: 'IT',
    location: 'Bangalore'
  },
  {
    _id: ObjectId('658c714a1f4725aa708a8529'),
    dept_id: 2,
    department: 'HR',
    location: 'New York'
  },
  {
    _id: ObjectId('658c714a1f4725aa708a852a'),
    dept_id: 3,
    department: 'management',
    location: 'Delhi'
  },
  {
    _id: ObjectId('658c714a1f4725aa708a852b'),
    dept_id: 4,
    department: 'Finance',
    location: 'Mumbai'
  },
  {
    _id: ObjectId('658c714a1f4725aa708a852c'),
    dept_id: 5,
    department: 'sales',
    location: 'Surat'
  }
]
Company>
```

Conclusion: - The Above Practical Setting up and Exploring MongoDB has been performed successfully.

Practical – 02

Aim: - Interacting With Redis

a) Install Redis on your lab server or local machine.
b) Store and retrieve data in Redis using various data structures like strings, lists, and sets.
c) Implement basic Redis commands for data manipulation and retrieval

Description: - Redis is an open-source in-memory storage, used as a distributed, in-memory key–valuedatabase, cache and message broker, with optional durability. Because it holds all data in memory andbecause of its design, Redis offers low-latency reads and writes, making it particularly suitable for use cases that require a cache.

Execution:

1. Installing Redis

- sudo apt-get update

```
keval@keval-VirtualBox:~$ sudo apt-get update
[sudo] password for keval:
Sorry, try again.
[sudo] password for keval:
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
keval@keval-VirtualBox:~$
```

- sudo apt-get install redis-server

```
keval@keval-VirtualBox:~$ sudo apt-get install redis-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libatomic1 amd64 12.3.0-1ubuntu1~22.04 [10.4 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libjemalloc2 amd64 5.2.1-4ubuntu1 [240 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 liblua5.1-0 amd64 5.1.5-8.1build4 [99.9 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 liblzf1 amd64 3.6-3 [7,444 B]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 lua-bitop amd64 1.0.2-5 [6,680 B]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 lua-cjson amd64 2.1.0+dfsg-2.1 [17.4 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 redis-tools amd64 5:6.0.16-1ubuntu1 [856 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 redis-server amd64 5:6.0.16-1ubuntu1 [45.9 kB]
Fetched 1.283 kB in 45s (28.4 kB/s)
```

2. Connecting Redis

- Redis-cli

```
keval@keval-VirtualBox:~$ redis-cli
127.0.0.1:6379> ping hi
"hi"
127.0.0.1:6379>
```

3. Performing CRUD operations using Strings, List and Sets.

- Creating Strings

```
127.0.0.1:6379> SET mykey "Redis connected !"
OK
127.0.0.1:6379> GET mykey
"Redis connected !"
```

- Working with Lists

```
127.0.0.1:6379> LPUSH mylist element1 element2 element3 element3
(integer) 4
127.0.0.1:6379> LRANGE mylist 0 -1
1) "element3"
2) "element3"
3) "element2"
4) "element1"
```

- Working with Sets

```
127.0.0.1:6379> SADD myset 1 2 3 4 4 5 6
(integer) 6
```

```
127.0.0.1:6379> SMEMBERS myset
1) "1"
2) "2"
3) "3"
4) "4"
5) "5"
6) "6"
```

- Basic Redis

  Commands# Check if

```
127.0.0.1:6379> Exists mykey
(integer) 1
```

a key exists

Delete a key

```
127.0.0.1:6379> DEL mykey
(integer) 1
```

Increment a key

```
127.0.0.1:6379> SET mynumber 10
OK
127.0.0.1:6379> INCR mynumber
(integer) 11
127.0.0.1:6379> GET mynumber
"11"
```

Decrement a key

```
127.0.0.1:6379> DECR mynumber
(integer) 10
127.0.0.1:6379> GET mynumber
"10"
```

Expire a key after a specific time (in seconds)

```
127.0.0.1:6379> EXPIRE mynumber 60
(integer) 1
127.0.0.1:6379> TTL mynumber
(integer) 50
127.0.0.1:6379> TTL mynumber
(integer) 48
```

Conclusion: - The Above Practical Interacting with Redis has been performed successfully

Practical - 03

Aim: - Working with HBase

a) Set up an HBase cluster in a lab environment.
b) Create an HBase table and define column families.
c) Insert sample data into the table.
d) Perform CRUD operations and retrieval of data in HBase

Description: - HBase is most effectively used to store non-relational data, accessed via the HBase API. Apache Phoenix is commonly used as a SQL layer on top of HBase allowing you to use familiar SQL syntax to insert, delete,and query data stored in HBase.

Execution:

1. Installing JAVA

```
keval@keval-VirtualBox:~$ sudo apt install openjdk-8-jdk-headless
[sudo] password for keval:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 java-common all 0.72build2 [6,782 B]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-8-jre-headless amd64 8u392-ga-1~22.04 [30.8 MB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ca-certificates-java all 20190909ubuntu1.2 [12.1 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-8-jdk-headless amd64 8u392-ga-1~22.04 [8,863 kB]
Fetched 39.7 MB in 18s (2,239 kB/s)
```

```
keval@keval-VirtualBox:~$ java -version
openjdk version "1.8.0_392"
OpenJDK Runtime Environment (build 1.8.0_392-8u392-ga-1~22.04-b08)
OpenJDK 64-Bit Server VM (build 25.392-b08, mixed mode)
keval@keval-VirtualBox:~$
```

Verifying the installed version of Java.

2. Create Hadoop User and Configure Password-less SSH

- Adding a new user Hadoop.

```
keval@keval-VirtualBox:~$ sudo adduser hadoop
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []: hadoop
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
keval@keval-VirtualBox:~$
```

- Adding the Hadoop user to the sudo group.

```
keval@keval-VirtualBox:~$ sudo usermod -aG sudo hadoop
```

- Installing the OpenSSH server and client.

```
keval@keval-VirtualBox:~$ sudo apt install openssh-server openssh-client -y
Need to get 1,657 kB of archives.
After this operation, 6,063 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-client amd64 1:8.9p1-3ubuntu0.5 [906 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-sftp-server amd64 1:8.9p1-3ubuntu0.5 [38.7 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-server amd64 1:8.9p1-3ubuntu0.5 [435 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-term all 6.3-2ubuntu0.1 [267 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 ssh-import-id all 5.11-0ubuntu1 [10.1 kB]
Fetched 1,657 kB in 13s (133 kB/s)
```

- Logging in with Hadoop user.

```
keval@keval-VirtualBox:~$ sudo su hadoop
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

hadoop@keval-VirtualBox:/home/keval$ ls
```

- Generating public and private key pairs.

```
hadoop@keval-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/o29v4A+JbmtWorNNRN+ynmqq94uhWdKgEkBhZ8H0rQ hadoop@keval-VirtualBox
The key's randomart image is:
+---[RSA 3072]----+
|.*+.             |
|o +.             |
| +E=             |
|  = o            |
|   . . .S ..     |
|      o.+.oo.    |
|     . =. B=o    |
|      o= BoXo.   |
|     .++X+X=+oo. |
+----[SHA256]-----+
```

- Adding the generated public key from id_rsa.pub to authorized keys.

```
hadoop@keval-VirtualBox:~$ sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
[sudo] password for hadoop:
hadoop@keval-VirtualBox:~$
```

- Changing the permissions of the authorized keys file.

```
hadoop@keval-VirtualBox:~$ sudo chmod 640 ~/.ssh/authorized_keys
hadoop@keval-VirtualBox:~$
```

- Verifying if the password-less SSH is functional.

```
hadoop@keval-VirtualBox:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:WedThkhDc83fRllxxVn4oRLZRyT48m1Y3VU6vyLAYBY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 6.2.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

443 updates can be applied immediately.
281 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Install Apache Hadoop

Download the latest stable version of Hadoop. To get the latest version, go to Apache Hadoop official download page.

wget https://archive.apache.org/dist/hadoop/core/hadoop-3.1.1/hadoop-3.1.1.tar.gz

- Downloading Hadoop

```
hadoop@keval-VirtualBox:~$ wget https://archive.apache.org/dist/hadoop/core/hadoop-3.1.1/hadoop-3.1.1.tar.gz
--2023-12-23 13:20:32--  https://archive.apache.org/dist/hadoop/core/hadoop-3.1.1/hadoop-3.1.1.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 334559382 (319M) [application/x-gzip]
Saving to: ‘hadoop-3.1.1.tar.gz’

hadoop-3.1.1.tar.gz                 100%[===================================================================================>] 319.06M
9m 29s

2023-12-23 13:40:02 (279 KB/s) - ‘hadoop-3.1.1.tar.gz’ saved [334559382/334559382]

hadoop@keval-VirtualBox:~$
```

- Extracting the downloaded file.

```
hadoop@keval-VirtualBox:~$ tar -xvzf hadoop-3.1.1.tar.gz
```

- Moving the extracted directory to the /usr/local/ directory.

```
hadoop@keval-VirtualBox:~$ sudo mv hadoop-3.1.1 /usr/local/hadoop
hadoop@keval-VirtualBox:~$
```

- Creating directory to store system logs.

```
hadoop@keval-VirtualBox:~$ sudo mkdir /usr/local/hadoop/logs
```

- Change the ownership of the Hadoop directory.

```
hadoop@keval-VirtualBox:~$ sudo chown -R hadoop:hadoop /usr/local/hadoop
hadoop@keval-VirtualBox:~$
```

3. Configure Hadoop

- Editing file ~/.bashrc to configure the Hadoop environment variables.

```
hadoop@keval-VirtualBox:~$ sudo nano ~/.bashrc
hadoop@keval-VirtualBox:~$
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy       ^Q Where Was
```

- Activating the environment variables.

```
hadoop@keval-VirtualBox:~$ source ~/.bashrc
hadoop@keval-VirtualBox:~$
```

- Finding the Java path.

```
hadoop@keval-VirtualBox:~$ which javac
/usr/bin/javac
hadoop@keval-VirtualBox:~$
```

- Finding the OpenJDK directory.

```
hadoop@keval-VirtualBox:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
hadoop@keval-VirtualBox:~$
```

- Editing the hadoop-env.sh file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd6
export HADOOP_CLASSPATH+=" $HADOOP_HOME/lib/*.jar"

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy       ^Q Where Was
```

- Browse to the Hadoop lib directory.

```
hadoop@keval-VirtualBox:~$ cd /usr/local/hadoop/lib
hadoop@keval-VirtualBox:/usr/local/hadoop/lib$
```

- Downloading the Javax activation file.

```
hadoop@keval-VirtualBox:/usr/local/hadoop/lib$ sudo wget https://jcenter.bintray.com/javax/activation/javax.activation-api/1.2.0/javax.activation-api-1.2.0.jar
--2023-12-23 15:16:28--  https://jcenter.bintray.com/javax/activation/javax.activation-api/1.2.0/javax.activation-api-1.2.0.jar
Resolving jcenter.bintray.com (jcenter.bintray.com)... 34.95.74.180
Connecting to jcenter.bintray.com (jcenter.bintray.com)|34.95.74.180|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 56674 (55K) [application/java-archive]
Saving to: 'javax.activation-api-1.2.0.jar'

javax.activation-api-1.2.0.jar       100%[=========================================================================>]  55.35K  --.-KB/s    in 0.03s

2023-12-23 15:16:29 (1.82 MB/s) - 'javax.activation-api-1.2.0.jar' saved [56674/56674]
```

- Coming back to home folder of Hadoop user

```
hadoop@keval-VirtualBox:/usr/local/hadoop/lib$ cd /home/hadoop
hadoop@keval-VirtualBox:~$
```

- Verifying the Hadoop version.

```
hadoop@keval-VirtualBox:~$ hadoop version
Hadoop 3.1.1
Source code repository https://github.com/apache/hadoop -r 2b9a8c1d3a2caf1e733d57f346af3ff0d5ba529c
Compiled by leftnoteasy on 2018-08-02T04:26Z
Compiled with protoc 2.5.0
From source with checksum f76ac55e5b5ff0382a9f7df36a3ca5a0
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.1.1.jar
hadoop@keval-VirtualBox:~$
```

- Editing the core-site.xml configuration file to specify the URL for your NameNode.

```
hadoop@keval-VirtualBox:~$ sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml
hadoop@keval-VirtualBox:~$
```

```
<property>
<name>fs.default.name</name>
<value>hdfs://0.0.0.0:9000</value>
<description>The default file system URI</description>
</property>


<configuration>
</configuration>
```

- Create a directory for storing node metadata and change the ownership to hadoop.

```
hadoop@keval-VirtualBox:~$ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
hadoop@keval-VirtualBox:~$ sudo chown -R hadoop:hadoop /home/hadoop/hdfs
hadoop@keval-VirtualBox:~$
```

- Editing hdfs-site.xml configuration file to define the location for storing node metadata, fs-image file.

```
hadoop@keval-VirtualBox:~$ sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
hadoop@keval-VirtualBox:~$
```

```
<configuration>

<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>file:///home/hadoop/hdfs/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>file:///home/hadoop/hdfs/datanode</value>
</property>
<property>
<name>dfs.permissions.enabled</name>
<value>false</value>
</property>

</configuration>
```

- Editing mapred-site.xml configuration file to define MapReduce values.

```
hadoop@keval-VirtualBox:~$ sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
hadoop@keval-VirtualBox:~$
```

```
<!-- Put site-specific property overrides in this file. -->


<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>yarn.app.mapreduce.am.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
<description>Change this to your hadoop location.</description>
</property>
<property>
<name>mapreduce.map.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
<description>Change this to your hadoop location.</description>
</property>
<property>
<name>mapreduce.reduce.env</name>
<value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
<description>Change this to your hadoop location.</description>
</property>
</configuration>
```

- Editing the yarn-site.xml configuration file and define YARN-related settings.

```
hadoop@keval-VirtualBox:~$ sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
hadoop@keval-VirtualBox:~$
```

```
-->
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>


</configuration>
```

- Validate the Hadoop configuration and format the HDFS NameNode.

```
hadoop@keval-VirtualBox:~$ hdfs namenode -format
2023-12-23 15:37:53,871 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = keval-VirtualBox/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.1.1
```

4. Start the Apache Hadoop Cluster

- Starting the NameNode and DataNode.

```
hadoop@keval-VirtualBox:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ED25519) to the list of known hosts.
Starting datanodes
Starting secondary namenodes [keval-VirtualBox]
hadoop@keval-VirtualBox:~$
```

- Startin the YARN resource and node managers.

```
hadoop@keval-VirtualBox:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@keval-VirtualBox:~$
```

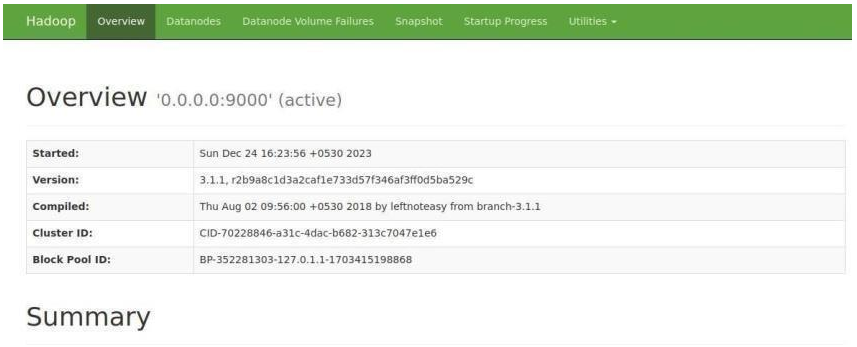
- Verifying all the running components.

```
hadoop@keval-VirtualBox:~$ jps
15616 NodeManager
15125 DataNode
14968 NameNode
15928 Jps
15305 SecondaryNameNode
15484 ResourceManager
hadoop@keval-VirtualBox:~$
```

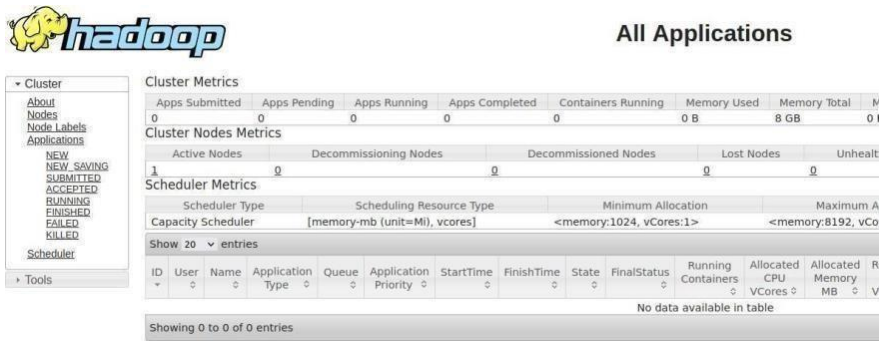
- Accessing the Hadoop NameNode and DataNode on browse

Access Apache Hadoop Web Interface

browser via http://localhost:9870



http://localhost:8088



5. Install HBase cluster using

   HadoopDownload Hbase

   wget https://archive.apache.org/dist/hbase/2.4.1/hbase-2.4.1-bin.tar.gz

- Downloading Hbase

```
hadoop@keval-VirtualBox:~$ wget https://archive.apache.org/dist/hbase/2.4.1/hbase-2.4.1-bin.tar.gz
```

- Extracting File.

```
hadoop@keval-VirtualBox:~$ tar -xvf hbase-2.4.1-bin.tar.gz
```

- Rename directory to hbase

```
hadoop@keval-VirtualBox:~$ mv hbase-2.4.1 hbase
hadoop@keval-VirtualBox:~$
```

- hbase-env.sh in hbase/conf and assign the JAVA_HOME path

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

^G Help          ^O Write Out     ^W Where Is      ^K Cut
^X Exit          ^R Read File     ^\ Replace       ^U Pas
```

- Editing the .bashrc file

```
hadoop@keval-VirtualBox:~$ sudo nano ~/.bashrc
hadoop@keval-VirtualBox:~$
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"


export HBASE_HOME=/home/hadoop/hbase
export PATH=$PATH:$HBASE_HOME/bin
```

- Reading the edited bashrc file to the running memory.

```
hadoop@keval-VirtualBox:~$ source ~/.bashrc
hadoop@keval-VirtualBox:~$
```

- Opening HBase-site.xml and editing properties in the file.

```
hadoop@keval-VirtualBox:~$ sudo nano hbase/conf/hbase-site.xml
hadoop@keval-VirtualBox:~$
```

```
<property>
<name>hbase.rootdir</name>
<value>hdfs://localhost:9000/hbase</value>
</property>
<property>
<name>hbase.cluster.distributed</name>
<value>true</value>
</property>
<property>
<name>hbase.zookeeper.quorum</name>
<value>localhost</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>hbase.zookeeper.property.clientPort</name>
<value>2181</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/hadoop/hbase/zookeeper</value>
</property>


</configuration>
```

Properties Explanation:

1. Setting up Hbase root directory in this property, for distributed set up we have to set this property
2. ZooKeeper quorum property should be set up here
3. Replication set up done in this property. By default we are placing replication as 1.In the fully distributed mode, multiple data nodes present so we can increase replication by placing more than 1 value in the dfs.replication property
4. Client port should be mentioned in this property
5. ZooKeeper data directory can be mentioned in this property

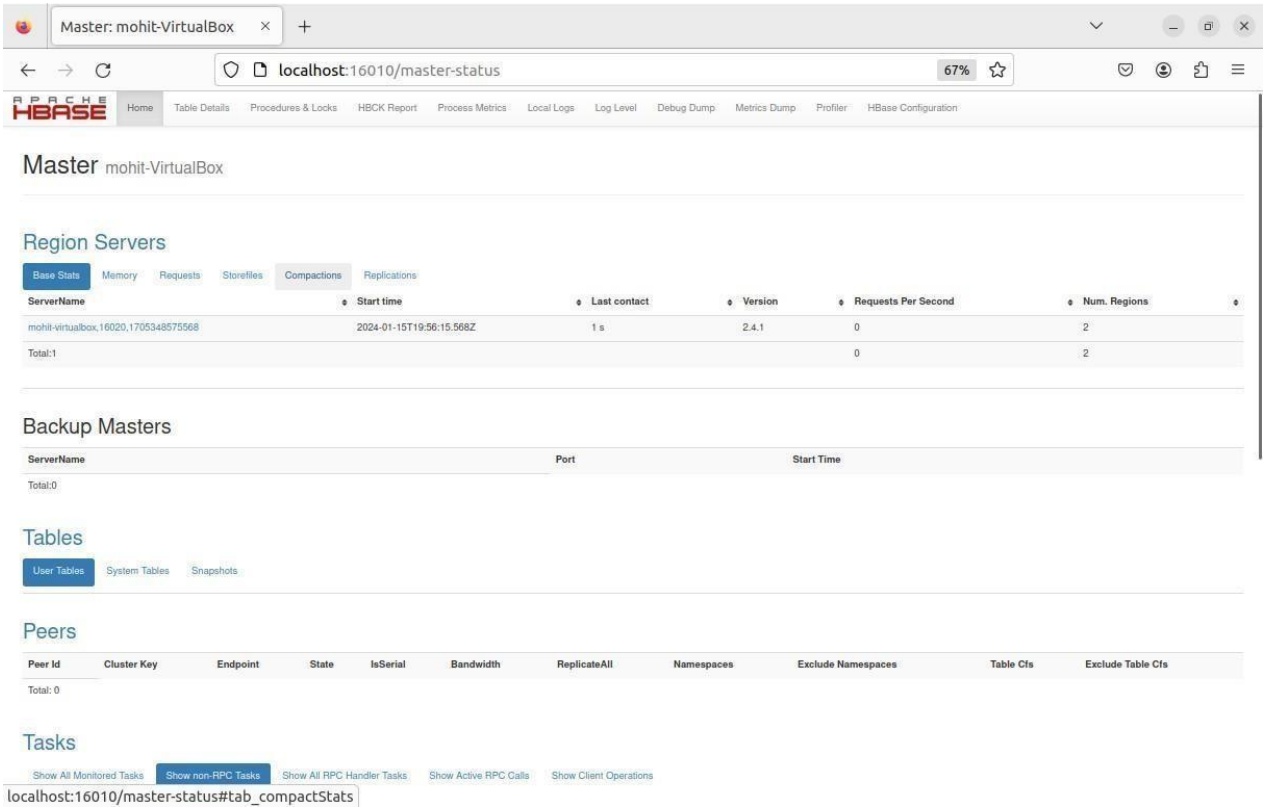
Start Hadoop daemons first and after that start HBase daemons as shown below:
start-dfs.sh
start-yarn.sh
start-hbase.sh

- Starting Hadoop and Hbase

```
hadoop@keval-VirtualBox:~$ start-hbase.sh
```

```
hadoop@keval-VirtualBox:~$ jps
15616 NodeManager
15125 DataNode
14968 NameNode
15305 SecondaryNameNode
23451 HQuorumPeer
24667 Jps
15484 ResourceManager
23709 HRegionServer
23550 HMaster
hadoop@keval-VirtualBox:~$
```

- You can access the HBase on your browser via http://localhost:16010.



- hbase shell

```
hadoop@mohit-VirtualBox:~$ hbase shell
/usr/local/hadoop/libexec/hadoop-functions.sh: line 2358: HADOOP_ORG.APACHE.HADO
OP.HBASE.UTIL.GETJAVAPROPERTY_USER: invalid variable name
/usr/local/hadoop/libexec/hadoop-functions.sh: line 2453: HADOOP_ORG.APACHE.HADO
OP.HBASE.UTIL.GETJAVAPROPERTY_OPTS: invalid variable name
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hbase/lib/client-facing-thirdpart
y/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.1, rb4d9639f66fccdb45fea0244202ffbd755341260, Fri Jan 15 10:58:57 PS
T 2021
Took 0.0014 seconds
hbase:001:0>
hbase:002:0>
hbase:003:0>
```

Create an HBase table and define column families

- create 'employe', 'pri_data', 'pro_data'

```
Took 0.4787 seconds
hbase:005:0> drop 'employe'
Took 0.4209 seconds
hbase:006:0>
hbase:007:0> create 'employe', 'pri_data', 'pro_data'
```

Insert sample data into the table

```
hbase:009:0> put 'employe', '1', 'pri_data:name', 'Andy'
Took 0.2212 seconds
hbase:010:0> put 'employe', '1', 'pri_data:age', '22'
Took 0.0209 seconds
hbase:021:0> put 'employe','1','pri_data:city','Indore'
Took 0.0342 seconds
hbase:012:0> put 'employe' ,'1', 'pro_data:post', 'asst. manager'
Took 0.0321 seconds
hbase:013:0> put 'employe' ,'1', 'pro_data:salary', '40k'
Took 0.0302 seconds
hbase:014:0> put 'employe' ,'2', 'pri_data:name', 'Icarus'
Took 0.0277 seconds
hbase:015:0> put 'employe' ,'2', 'pri_data:age', '22'
```

```
hbase:017:0> put 'employe' ,'2', 'pro_data:post', 'sales manager'
Took 0.0184 seconds
hbase:018:0> put 'employe' ,'2', 'pro_data:salary', '35k'
Took 0.0266 seconds
hbase:020:0> put 'employe','2','pro_data:city','Mumbai'
Took 0.0229 seconds
```

Perform CRUD operations and retrieval of data in HBase.

- **GET Operations:**

```
hbase:003:0> get 'employe','1'
COLUMN                  CELL
 pri_data:age           timestamp=2024-01-16T02:45:12.643, value=22
 pri_data:city          timestamp=2024-01-16T02:47:23.531, value=Indore
 pri_data:name          timestamp=2024-01-16T02:44:27.388, value=Andy
 pro_data:post          timestamp=2024-01-16T02:45:12.901, value=asst. manager
 pro_data:salary        timestamp=2024-01-16T02:45:13, value=40k
1 row(s)
Took 1.0035 seconds
hbase:004:0> get 'employe','2'
COLUMN                  CELL
 pri_data:age           timestamp=2024-01-16T02:45:13.235, value=22
 pri_data:name          timestamp=2024-01-16T02:45:13.098, value=Icarus
 pro_data:city          timestamp=2024-01-16T02:46:46.355, value=Mumbai
 pro_data:post          timestamp=2024-01-16T02:45:13.468, value=sales manager
 pro_data:salary        timestamp=2024-01-16T02:45:37.453, value=35k
1 row(s)
Took 0.0823 seconds
```

- **Delete Operations:**

```
hbase:006:0> delete 'employe','1','pri_data:city'
Took 0.0693 seconds
hbase:007:0> get 'employe','1'
COLUMN                  CELL
 pri_data:age           timestamp=2024-01-16T02:45:12.643, value=22
 pri_data:name          timestamp=2024-01-16T02:44:27.388, value=Andy
 pro_data:post          timestamp=2024-01-16T02:45:12.901, value=asst. manager
 pro_data:salary        timestamp=2024-01-16T02:45:13, value=40k
1 row(s)
Took 0.0819 seconds
```

```
hbase:008:0> deleteall 'employe','1'
Took 0.0201 seconds
hbase:009:0> get 'employe','1'
COLUMN                  CELL
0 row(s)
Took 0.0582 seconds
```

- **Scan:**

```
hbase:010:0> scan 'employe'
ROW                     COLUMN+CELL
 2                      column=pri_data:age, timestamp=2024-01-16T02:45:13.235, value=2
                        2
 2                      column=pri_data:name, timestamp=2024-01-16T02:45:13.098, value=
                        Icarus
 2                      column=pro_data:city, timestamp=2024-01-16T02:46:46.355, value=
                        Mumbai
 2                      column=pro_data:post, timestamp=2024-01-16T02:45:13.468, value=
                        sales manager
 2                      column=pro_data:salary, timestamp=2024-01-16T02:45:37.453, valu
                        e=35k
1 row(s)
Took 0.1414 seconds
```

- **List Tables**:

```
hbase:011:0> list
TABLE
emp
employe
2 row(s)
Took 0.2369 seconds
=> ["emp", "employe"]
```

Practical – 04

Aim: Apache Cassandra Operations

a) Install and configure Apache Cassandra in a lab environment.
b) Create a keyspace and define a table schema.
c) Insert data into the table.
d) Perform CRUD operations and query data from Apache Cassandra.

Description: - Apache Cassandra is a highly scalable and distributed NoSQL database system designed to handle large amounts of data across multiple commodity servers without a single pointof failure. Developed by Facebook and later open- sourced, Cassandra is renowned for its high availability, fault tolerance, and linear scalability, making it a popular choice for organizations dealing with vast amounts of data.

Execution
:

a) Install and configure Apache Cassandra in a lab environment.

Check if you have java installed

```
snehanaik@snehanaik-VirtualBox:~$ java -version
openjdk version "1.8.0_392"
OpenJDK Runtime Environment (build 1.8.0_392-8u392-ga-1~22.04-b08)
OpenJDK 64-Bit Server VM (build 25.392-b08, mixed mode)
snehanaik@snehanaik-VirtualBox:~$
```

A) install java

1. Refresh:

```
snehanaik@snehanaik-VirtualBox:~$ sudo apt-get update
[sudo] password for snehanaik:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:3 https://apache.jfrog.io/artifactory/cassandra-deb 41x InRelease
Fetched 229 kB in 2s (104 kB/s)
Reading package lists... Done
W: https://debian.cassandra.apache.org/dists/41x/InRelease: Key is stored in leg
acy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in a
pt-key(8) for details.
snehanaik@snehanaik-VirtualBox:~$
```

sudo apt-get update

Addthe Cassandra Repository File

Add the Cassandra repository to the system's repository file.

echo"deb http://www.apache.org/dist/cassandra/debian 40x main" | sudo tee - a

/etc/apt/sources.list.d/cassandra.sources.list deb http://www.apache.org/dist/cassandra/debian 41x main

```
snehanaik@snehanaik-VirtualBox:~$ echo "deb https://debian.cassandra.apache.org
41x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
deb https://debian.cassandra.apache.org 41x main
snehanaik@snehanaik-VirtualBox:~$
```

- install curl

```
snehanaik@snehanaik-VirtualBox:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.15).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
  intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
  libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
  libcodec2-1.0 libdav1d5 libflite1 libgme0 libgsm1
  libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0 libllvm15 libmfx1
  libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4
  librubberband2 libserd-0-0 libshine3 libsnappy1v5 libsord-0-0 libsratom-0-0
  libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0
  libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1
  libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
  mesa-va-drivers mesa-vdpau-drivers pocketsphinx-en-us systemd-hwe-hwdb
  va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
```

To avoid issues with the repository, add the GPG key and increase the security ofthe repository.

- curl - https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -

```
snehanaik@snehanaik-VirtualBox:~$ curl https://downloads.apache.org/cassandra/KEYS
| sudo apt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0Warni
ng: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-k
ey(8)).
100  249k  100  249k    0     0   240k      0  0:00:01  0:00:01 --:--:--  240k
OK
snehanaik@snehanaik-VirtualBox:~$
```

- update and install cassandra

```
snehanaik@snehanaik-VirtualBox:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:2 https://apache.jfrog.io/artifactory/cassandra-deb 41x InRelease
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 229 kB in 2s (103 kB/s)
Reading package lists... Done
W: https://debian.cassandra.apache.org/dists/41x/InRelease: Key is stored in legacy trusted.g
pg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
W: Target Packages (main/binary-amd64/Packages) is configured multiple times in /etc/apt/sour
ces.list.d/cassandra.sources.list:1 and /etc/apt/sources.list.d/cassandra.sources.list:2
W: Target Packages (main/binary-i386/Packages) is configured multiple times in /etc/apt/sourc
es.list.d/cassandra.sources.list:1 and /etc/apt/sources.list.d/cassandra.sources.list:2
W: Target Packages (main/binary-all/Packages) is configured multiple times in /etc/apt/source
s.list.d/cassandra.sources.list:1 and /etc/apt/sources.list.d/cassandra.sources.list:2
W: Target Translations (main/i18n/Translation-en_IN) is configured multiple times in /etc/apt
snehanaik@snehanaik-VirtualBox:~$ sudo apt-get install cassandra
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cassandra is already the newest version (4.1.3).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
  intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
  libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
  libcodec2-1.0 libdav1d5 libflite1 libgme0 libgsm1 libgstreamer-plugins-bad1.0-0
  libigdgmm12 liblilv-0-0 libllvm15 libmfx1 libmysofa1 libnorm1 libopenmpt0
  libpgm-5.3-0 libpostproc55 librabbitmq4 librubberband2 libserd-0-0 libshine3
  libsnappy1v5 libsord-0-0 libsratom-0-0 libsrt1.4-gnutls libssh-gcrypt-4
  libswresample3 libswscale5 libudfread0 libva-drm2 libva-wayland2 libva-x11-2
  libva2 libvdpau1 libvidstab1.1 libx265-199 libxvidcore4 libzimg2 libzmq5
  libzvbi-common libzvbi0 mesa-va-drivers mesa-vdpau-drivers pocketsphinx-en-us
  systemd-hwe-hwdb va-driver-all vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 516 not upgraded.
snehanaik@snehanaik-VirtualBox:~$
```

- check nodetool status

```
snehanaik@snehanaik-VirtualBox:~$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load        Tokens  Owns (effective)  Host ID
          Rack
UN  127.0.0.1  139.49 KiB  16      100.0%            825f7984-25d3-40ce-8466-75d
381406f26  rack1
```

- start cassandra

```
snehanaik@snehanaik-VirtualBox:~$ sudo systemctl start cassandra
snehanaik@snehanaik-VirtualBox:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.3 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh>
```

b) Create a keyspace and define a table schema.

Create a KEYSPACE

```
cqlsh> CREATE KEYSPACE user WITH replication ={'class':'SimpleStrategy','replication_factor':'1'};
cqlsh> USE user
   ... ;
cqlsh:user>
```

Create a table

```
cqlsh:user> CREATE TABLE user(id UUID PRIMARY KEY,name TEXT,email TEXT);
cqlsh:user> SELECT * FROM USER
        ... ;

 id | email | name
----+-------+------

(0 rows)
```

c) Insert data into the table.

Add values in said table

```
cqlsh:user> INSERT INTO USER(id,name,email)values(uuid(),'kanye','vul@gmail.com');
cqlsh:user> SELECT * FROM USER ;

 id                                   | email         | name
--------------------------------------+---------------+--------
 c0dfcb3c-b036-41b6-a82d-eea104e5e719 | rep@gmail.com | taylor
 d0b9f4db-5cdd-4c33-bee3-090a879d00c6 | lil@gmail.com |  wayne
 44f5a159-f943-4a4e-9b4e-d45d2eec544e | bcd@gmail.com |  nicki
 985844eb-e459-44a3-b00f-6e3d4681fbe7 | abc@gmail.com |  sneha
 199b017d-6b87-472c-808b-c5a96eab63f7 | vul@gmail.com |  kanye

(5 rows)
```

Alter table

```
cqlsh:user> ALTER TABLE USER add salary INT;
cqlsh:user> SELECT * FROM USER ;

 id                                   | email         | name   | salary
--------------------------------------+---------------+--------+--------
 c0dfcb3c-b036-41b6-a82d-eea104e5e719 | rep@gmail.com | taylor |   null
 d0b9f4db-5cdd-4c33-bee3-090a879d00c6 | lil@gmail.com |  wayne |   null
 44f5a159-f943-4a4e-9b4e-d45d2eec544e | bcd@gmail.com |  nicki |   null
 985844eb-e459-44a3-b00f-6e3d4681fbe7 | abc@gmail.com |  sneha |   null
 199b017d-6b87-472c-808b-c5a96eab63f7 | vul@gmail.com |  kanye |   null
```

a) Perform CRUD operations and query data from Apache Cassandra.

Deletedata from table

```
cqlsh:user> TRUNCATE USER;
cqlsh:user> SELECT * FROM USER ;

 id | email | name | salary
----+-------+------+--------

(0 rows)
```

Drop/delete table

```
cqlsh:user> DROP TABLE USER
        ... ;
cqlsh:user> SELECT * FROM USER ;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table user does not exist"
cqlsh:user>
```

Conclusion: - The Above Practical Apache Cassandra Operations has been performed Successfully.

Practical -05

Aim: Querying MongoDB and HBase
a) Write and execute MongoDB queries to retrieve specific data from a collection.
b) Perform queries on HBase tables using HBase shell commands

Execution:
Q1. Create collection employee and department in mongodb using company database.Insert Sample Data (At least 10rows in employee collection and 5 rows in department. Make sure you use the same dept_id mentions by you in department table). Make sure you insert data in order to get output for the queries below:

```
company> db.department.insertMany([
"Operations", location: "Dallas" }
...   { _id: 1, name: "IT", location: "New York" },
...   { _id: 2, name: "HR", location: "Chicago" },
...   { _id: 3, name: "Finance", location: "Los Angeles" },
...   { _id: 4, name: "Marketing", location: "San Francisco" },
...   { _id: 5, name: "Operations", location: "Dallas" }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
company> db.createCollection("employee");
{ ok: 1 }
```

```
company> db.employee.insertMany([
... { name: "Alice Johnson", age: 28, dept_id: 1 },
... { name: "Bob Smith", age: 32, dept_id: 2 },
... { name: "Charlie Brown", age: 29, dept_id: 1 },
... { name: "David Miller", age: 35, dept_id: 3 },
... { name: "Emma Davis", age: 31, dept_id: 4 },
... { name: "Frank Wilson", age: 40, dept_id: 2 },
... { name: "Grace Taylor", age: 27, dept_id: 5 },
... { name: "Harry Johnson", age: 33, dept_id: 1 },
... { name: "Ivy Martinez", age: 36, dept_id: 3 },
... { name: "Jack Anderson", age: 30, dept_id: 4 },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65a6173f0756174d3330a4f8'),
    '1': ObjectId('65a6173f0756174d3330a4f9'),
    '2': ObjectId('65a6173f0756174d3330a4fa'),
    '3': ObjectId('65a6173f0756174d3330a4fb'),
    '4': ObjectId('65a6173f0756174d3330a4fc'),
    '5': ObjectId('65a6173f0756174d3330a4fd'),
    '6': ObjectId('65a6173f0756174d3330a4fe'),
    '7': ObjectId('65a6173f0756174d3330a4ff'),
    '8': ObjectId('65a6173f0756174d3330a500'),
    '9': ObjectId('65a6173f0756174d3330a501')
  }
}
company>
```

1. Retrieve all employees

```
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f8'),
    name: 'Alice Johnson',
    age: 28,
    dept_id: 1
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4f9'),
    name: 'Bob Smith',
    age: 32,
    dept_id: 2
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 29,
    dept_id: 1
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 35,
    dept_id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 31,
    dept_id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fd'),
    name: 'Frank Wilson',
    age: 40,
    dept_id: 2
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 27,
    dept_id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 33,
    dept_id: 1
  },
  {
    _id: ObjectId('65a6173f0756174d3330a500'),
    name: 'Ivy Martinez',
    age: 36,
    dept_id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a501'),
    name: 'Jack Anderson',
    age: 30,
    dept_id: 4
  }
]
```

2. Retrieve employees in a specific department (e.g., IT department)

```
company> db.employee.find({ dept_id: 1 });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 30,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7
  }
]
```

3. Retrieve all departments

```
company> db.department.find()
[
  { _id: 1, name: 'IT', location: 'New York' },
  { _id: 2, name: 'HR', location: 'Chicago' },
  { _id: 3, name: 'Finance', location: 'Los Angeles' },
  { _id: 4, name: 'Marketing', location: 'San Francisco' },
  { _id: 5, name: 'Operations', location: 'Dallas' }
]
```

4. Retrieve employees older than 30 years

```
company> db.employee.find({ age: { $gt: 30 } })
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 36,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 32,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7
  }
]
```

5. Retrieve employees sorted by age in ascending order

```
company> db.employee.find().sort({ age: 1 });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 28,
    dept_id: 5,
    id: 6
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 30,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 32,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 36,
    dept_id: 3,
    id: 4
  }
]
```

6. Retrieve employees belonging to a specific department with their department details (using $lookup)

```
company> db.employee.aggregate([{ $lookup: { from: "department", localField: "dept_id", foreignField: "_id", as: "department" } }]);
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 30,
    dept_id: 1,
    id: 3,
    department: [
      {
        _id: 1,
        name: 'Information Technology',
        location: 'San Francisco',
        id: 1
      }
    ]
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 36,
    dept_id: 3,
    id: 4,
    department: [ { _id: 3, name: 'Finance', location: 'Los Angeles', id: 3 } ]
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 32,
    dept_id: 4,
    id: 5,
    department: [
      {
        _id: 4,
        name: 'Marketing',
        location: 'San Francisco',
        id: null
      }
    ]
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 28,
    dept_id: 5,
    id: 6,
    department: [ { _id: 5, name: 'Operations', location: 'Dallas', id: 5 } ]
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7,
    department: [
      {
        _id: 1,
        name: 'Information Technology',
        location: 'San Francisco',
        id: 1
      }
    ]
  }
]
```

7. Retrieve employees sorted by department and age

```
company> db.employee.find().sort({ dept_id: 1, age: 1 });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 30,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 36,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 32,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 28,
    dept_id: 5,
    id: 6
  }
]
```

8. Increment the age of all employees by 1 year

```
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 32,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 38,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 34,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 30,
    dept_id: 5,
    id: 6
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 36,
    dept_id: 1,
    id: 7
  }
]
company> db.employee.updateMany({}, { $inc: { age: 1 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
```

```
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 33,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 39,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 35,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 31,
    dept_id: 5,
    id: 6
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 37,
    dept_id: 1,
    id: 7
  }
]
```

9. Rename a department (e.g., rename "IT" department to "Information Technology")

```
company> db.department.update({ name: "IT" }, { $set: { name: "Information Technology" } });
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
company> db.department.find()
[
  { _id: 1, name: 'Information Technology', location: 'New York' },
  { _id: 2, name: 'HR', location: 'Chicago' },
  { _id: 3, name: 'Finance', location: 'Los Angeles' },
  { _id: 4, name: 'Marketing', location: 'San Francisco' },
  { _id: 5, name: 'Operations', location: 'Dallas' }
]
```

10. Update an employee's information (e.g., update Bob Smith's age)

```
company> db.employee.update({ name: "Bob Smith" }, { $set: { age: 33 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
company> db.employee.find({name:'Bob Smith'});
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f9'),
    name: 'Bob Smith',
    age: 33,
    dept_id: 2
  }
]
```

11. Update a department's location (e.g., update IT department's location to "San Francisco")

```
company> db.department.find();
[
  { _id: 2, name: 'HR', location: 'Chicago' },
  { _id: 3, name: 'Finance', location: 'Los Angeles' },
  { _id: 4, name: 'Marketing', location: 'San Francisco', id: null },
  { _id: 5, name: 'Operations', location: 'Dallas' },
  { _id: 1, name: 'IT', location: 'New York' }
]
company> db.department.update({ name: "IT" }, { $set: { name: "Information Technology" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
company> db.department.update({ name: "Information Technology" }, { $set: { location: "San Francisco" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
company> db.department.find();
[
  { _id: 2, name: 'HR', location: 'Chicago' },
  { _id: 3, name: 'Finance', location: 'Los Angeles' },
  { _id: 4, name: 'Marketing', location: 'San Francisco', id: null },
  { _id: 5, name: 'Operations', location: 'Dallas' },
  { _id: 1, name: 'Information Technology', location: 'San Francisco' }
]
```

12. Delete a specific employee by their ID

```
company> db.department.deleteOne({ _id: ObjectId('65a62699bb66225202cbbfcc') });
{ acknowledged: true, deletedCount: 1 }
```

13. Delete a specific department by its ID

```
company> db.department.find();
[
  { _id: 3, name: 'Finance', location: 'Los Angeles', id: 3 },
  { _id: 4, name: 'Marketing', location: 'San Francisco', id: null },
  { _id: 5, name: 'Operations', location: 'Dallas', id: 5 },
  {
    _id: 1,
    name: 'Information Technology',
    location: 'San Francisco',
    id: 1
  }
]
company> db.department.deleteOne({_id:1});
{ acknowledged: true, deletedCount: 1 }
company> db.department.find();
[
  { _id: 3, name: 'Finance', location: 'Los Angeles', id: 3 },
  { _id: 4, name: 'Marketing', location: 'San Francisco', id: null },
  { _id: 5, name: 'Operations', location: 'Dallas', id: 5 }
]
```

14. Delete employees older than 40 years

```
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 37,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 43,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 39,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 35,
    dept_id: 5,
    id: 6
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 41,
    dept_id: 1,
    id: 7
  }
]
company> db.employee.deleteMany({ age: { $gt: 40 } });
{ acknowledged: true, deletedCount: 2 }
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 37,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 39,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 35,
    dept_id: 5,
    id: 6
  }
]
```

15. Delete employees not belonging to any department (where departmentId is null)

```
company> cursor.forEach(function(doc) {
...     var newValue = i;
...     db.employee.update(
...         { _id: doc._id },
...         { $set: { id: newValue } }
...     );
...     i++;});
```

```
company> db.employee.find()
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f8'),
    name: 'Alice Johnson',
    age: 29,
    dept_id: 1,
    id: 1
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4f9'),
    name: 'Bob Smith',
    age: 33,
    dept_id: 2,
    id: 2
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fa'),
    name: 'Charlie Brown',
    age: 30,
    dept_id: 1,
    id: 3
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fb'),
    name: 'David Miller',
    age: 36,
    dept_id: 3,
    id: 4
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fc'),
    name: 'Emma Davis',
    age: 32,
    dept_id: 4,
    id: 5
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4fe'),
    name: 'Grace Taylor',
    age: 28,
    dept_id: 5,
    id: 6
  },
  {
    _id: ObjectId('65a6173f0756174d3330a4ff'),
    name: 'Harry Johnson',
    age: 34,
    dept_id: 1,
    id: 7
  },
  {
    _id: ObjectId('65a6173f0756174d3330a500'),
    name: 'Ivy Martinez',
    age: 37,
    dept_id: null,
    id: 8
  }
]
```

```
company> db.employee.find({dept_id:null})
[
  {
    _id: ObjectId('65a6173f0756174d3330a500'),
    name: 'Ivy Martinez',
    age: 37,
    dept_id: null,
    id: 8
  }
]
company> db.employee.deleteMany({ dept_id: null });
{ acknowledged: true, deletedCount: 1 }
company> db.employee.find({dept_id:null})
```

16. Delete an employee (e.g., delete Alice Johnson from the employees)

```
company> db.employee.find({ name: "Alice Johnson" });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f8'),
    name: 'Alice Johnson',
    age: 29,
    dept_id: 1,
    id: 1
  }
]
company> db.employee.deleteOne({ name: "Alice Johnson" });
{ acknowledged: true, deletedCount: 1 }
company> db.employee.find({ name: "Alice Johnson" });

company> db.employee.find({ dept_id: 2 });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f9'),
    name: 'Bob Smith',
    age: 33,
    dept_id: 2,
    id: 2
  }
]
```

17. Delete employees in a specific department (e.g., delete all employees in HR department)

```
company> db.employee.find({ dept_id: 2 });
[
  {
    _id: ObjectId('65a6173f0756174d3330a4f9'),
    name: 'Bob Smith',
    age: 33,
    dept_id: 2,
    id: 2
  }
]
company> db.employee.deleteMany({ dept_id: 2 });
{ acknowledged: true, deletedCount: 1 }
company> db.employee.find({ dept_id: 2 });
```

18. Delete a department (e.g., delete the HR department)

```
company> db.department.find({ name: "HR" });
[ { _id: 2, name: 'HR', location: 'Chicago', id: 2 } ]
company> db.department.deleteOne({ name: "HR" });
{ acknowledged: true, deletedCount: 1 }
company> db.department.find({ name: "HR" });
```

Hbase:

Q2. create the 'student' table with two column families ('info' and 'grades')

```
hbase:003:0> create 'student','info','grades'
Created table student
Took 2.1790 seconds
=> Hbase::Table - student
```

1. Insert Data:
Imagine you have a new student, John Doe. How would you use the HBase shell to insert his name ('John Doe') into the 'info:name' column for the corresponding row?

```
hbase:005:0> put 'student','1','info:name','John Doe'
Took 0.3133 seconds
```

```
hbase:010:0> put 'student','1','grades:English','B'
Took 0.0261 seconds
```

```
hbase:037:0> put 'student','1','grades:math','B'
Took 0.0222 seconds
```

2. Get Student Information:
If you want to retrieve all information about a student with ID '1' named John Doe,how would you use the HBase shell to fetch this data?

```
hbase:030:0> get 'student','1'
COLUMN                                CELL

 grades:English                       timestamp=2024-01-16T08:34:53.398, value=B

 grades:math                          timestamp=2024-01-16T08:40:01.563, value=c

 info:name                            timestamp=2024-01-16T08:33:01.902, value=John
 Doe
1 row(s)
Took 0.0519 seconds
```

3. Update Grade:

To update John Doe's math grade from 'B' to 'A' in the 'grades:math' column:3. Update Grade:
Suppose John Doe's math grade changes from 'B' to 'A'. How can you use the HBaseshell to update this specific grade in the 'grades:math' column?

```
hbase:037:0> put 'student','1','grades:math','B'
Took 0.0222 seconds

hbase:038:0>
hbase:039:0> get 'student','1'
COLUMN                                CELL

 grades:English                       timestamp=2024-01-16T08:34:53.398, value=B

 grades:math                          timestamp=2024-01-16T08:57:33.653, value=B

 info:name                            timestamp=2024-01-16T08:33:01.902, value=John
 Doe
1 row(s)
Took 0.0798 seconds
```

4. Get Specific Column:

If you are interested in fetching only the name of a student with ID '1', how would youmodify the HBase shell command to retrieve just this specific column?

```
hbase:042:0> get 'student', '1', {COLUMN => 'info:name'}
COLUMN                CELL
 info:name            timestamp=2024-01-16T08:33:01.902, value=John Doe
1 row(s)
Took 0.0415 seconds
```

5. Scan Table:

You need to view all students' information in the 'student' table. How would you usethe HBase shell to scan and display the entire table?

```
hbase:043:0> scan 'student'
ROW                   COLUMN+CELL
 1                    column=grades:English, timestamp=2024-01-16T08:34:53.398,
                      value=B
 1                    column=grades:math, timestamp=2024-01-16T08:58:04.359, val
                      ue=A
 1                    column=info:name, timestamp=2024-01-16T08:33:01.902, value
                      =John Doe
 2                    column=grades:English, timestamp=2024-01-16T08:38:33.636,
                      value=A
 2                    column=grades:math, timestamp=2024-01-16T08:38:45.027, val
                      ue=A
 2                    column=info:name, timestamp=2024-01-16T08:36:42.460, value
                      =keval saud
 3                    column=grades:English, timestamp=2024-01-16T08:41:44.997,
                      value=c
 3                    column=grades:math, timestamp=2024-01-16T08:41:28.269, val
                      ue=A
 3                    column=info:name, timestamp=2024-01-16T08:41:06.038, value
                      =vinay tawde
 4                    column=grades:English, timestamp=2024-01-16T08:43:34.286,
                      value=B+
 4                    column=grades:math, timestamp=2024-01-16T08:44:10.215, val
                      ue=B
 4                    column=info:name, timestamp=2024-01-16T08:43:07.250, value
                      =Kavita Singh
4 row(s)
Took 0.1213 seconds
```

6. Get Multiple Rows:

If you want to retrieve information for students with IDs '2' and '3', how would youstructure the HBase shell command to achieve this?

```
hbase:052:0> scan 'student', {STARTROW => '2', ENDROW => '3'}
ROW                                   COLUMN+CELL

 2                                    column=grades:English, timestamp=2024-01-16T0
8:38:33.636, value=A
 2                                    column=grades:math, timestamp=2024-01-16T08:3
8:45.027, value=A
 2                                    column=info:name, timestamp=2024-01-16T08:36:
42.460, value=keval saud
1 row(s)
Took 0.0464 seconds
```

7. Delete Data:

Imagine you need to remove John Doe's name from the 'info:name' column.
Howwould you use the HBase shell to delete this specific cell?

```
hbase:055:0> delete 'student', '1', 'info:name'
Took 0.0688 seconds
```

8. Get All Columns in a Row:
If you want to retrieve all columns for a specific student (e.g., with ID '1'), how wouldyou structure the HBase shell command?

```
hbase:057:0> get 'student', '1'
COLUMN                CELL
 grades:English       timestamp=2024-01-16T08:34:53.398, value=B
 grades:math          timestamp=2024-01-16T08:58:04.359, value=A
1 row(s)
Took 0.0481 seconds
```

9. Filter Data by Grade:

Suppose you want to find all students who scored an 'A' in any subject. How wouldyou construct the HBase shell command to achieve this?

```
hbase:058:0> scan 'student', {FILTER => "ValueFilter(=, 'binary:A')"}
ROW                   COLUMN+CELL
 1                    column=grades:math, timestamp=2024-01-16T08:58:04.359, val
                      ue=A
 2                    column=grades:English, timestamp=2024-01-16T08:38:33.636,
                      value=A
 2                    column=grades:math, timestamp=2024-01-16T08:38:45.027, val
                      ue=A
 3                    column=grades:math, timestamp=2024-01-16T08:41:28.269, val
                      ue=A
3 row(s)
Took 0.1509 seconds
```

10. Count Rows:

You are tasked with determining the total number of students in the 'student' table.How would you use the HBase shell to count the rows?

```
hbase:059:0> count 'student'
4 row(s)
Took 0.1063 seconds
=> 4
```

11. Get Latest Version of a Cell:

To retrieve the most recent version of John Doe's name from the 'info:name' column:

```
hbase:061:0> get 'student', '1', {COLUMN => 'info:name', VERSIONS => 1}
COLUMN                CELL
0 row(s)
Took 0.0640 seconds
```

12. Scan with Row Filter:

You want to view information for students whose IDs start with '2'. How would youmodify the HBase shell command to scan rows based on this criterion?

```
hbase:061:0> get 'student', '1', {COLUMN => 'info:name', VERSIONS => 1}
COLUMN                CELL
0 row(s)
Took 0.0640 seconds
hbase:062:0> scan 'student', {FILTER => "RowFilter(=, 'regexstring:^2')"}
ROW                   COLUMN+CELL
 2                    column=grades:English, timestamp=2024-01-16T08:38:33.636,
                      value=A
 2                    column=grades:math, timestamp=2024-01-16T08:38:45.027, val
                      ue=A
 2                    column=info:name, timestamp=2024-01-16T08:36:42.460, value
                      =keval saud
1 row(s)
Took 0.1003 seconds
```

13. Delete Entire Row:

Suppose you need to remove all information for a student with ID '2'. How would youuse the HBase shell to delete the entire row?

```
hbase:063:0> deleteall 'student', '2'
Took 0.0329 seconds
```

14. Get Row with Maximum Version:

To retrieve the student's information with ID '1' limited to the last two versions:

```
hbase:059:0> count 'student'
4 row(s)
Took 0.1063 seconds
=> 4
```

15. Scan with Column Range:

You are interested in viewing only the 'grades' and 'info' columns for all students.How would you modify the HBase shell command to achieve this?

```
hbase:066:0> scan 'student', {COLUMNS => ['info', 'grades']}
ROW                   COLUMN+CELL
 1                    column=grades:English, timestamp=2024-01-16T08:34:53.398,
                      value=B
 1                    column=grades:math, timestamp=2024-01-16T08:58:04.359, val
                      ue=A
 3                    column=grades:English, timestamp=2024-01-16T08:41:44.997,
                      value=c
 3                    column=grades:math, timestamp=2024-01-16T08:41:28.269, val
                      ue=A
 3                    column=info:name, timestamp=2024-01-16T08:41:06.038, value
                      =vinay tawde
 4                    column=grades:English, timestamp=2024-01-16T08:43:34.286,
                      value=B+
 4                    column=grades:math, timestamp=2024-01-16T08:44:10.215, val
                      ue=B
 4                    column=info:name, timestamp=2024-01-16T08:43:07.250, value
                      =Kavita Singh
3 row(s)
Took 0.0704 seconds
```

16. Filter Data by Name Prefix:

How would you use the HBase shell to find all students whose names start with'John'?

```
hbase:067:0> scan 'student', {FILTER => "PrefixFilter('John')"}
ROW                   COLUMN+CELL
0 row(s)
Took 0.0467 seconds
```

17. Get Timestamp of a Cell:

If you need to retrieve the timestamp of John Doe's math grade ('grades:math') when itwas last updated, how would you structure the HBase shell command?

```
hbase:071:0> get 'student', '1', {COLUMN => 'grades:math', TIMERANGE => [0, 9227
5678357907]}
COLUMN                CELL
 grades:math          timestamp=2024-01-16T08:58:04.359, value=A
1 row(s)
Took 0.0584 seconds
```

18. Disable Table:

Your task is to perform maintenance on the 'student' table. How would you use theHBase shell to temporarily disable the table?

```
hbase:072:0> disable 'student'
Took 0.8130 seconds
```

19. Enable Table:

After completing maintenance, how would you use the HBase shell to re-enable the'student' table?

```
hbase:075:0> enable 'student'
Took 0.8482 seconds
```

20. Drop Table:

Imagine you no longer need the 'student' table. How would you use the HBase shell topermanently delete the table and all its data?

```
hbase:076:0> disable 'student'
Took 0.3832 seconds
hbase:077:0> drop 'student'
Took 0.4282 seconds
hbase:078:0> list
TABLE
emp
employe
2 row(s)
Took 0.0185 seconds
=> ["emp", "employe"]
```

Practical - 06

Aim: Redis Data Manipulation

a) Use Redis commands to manipulate and modify data stored in different data structures.
b) Retrieve specific data using Redis query operations.

Description:

Redis, a high-performance key-value store, boasts in-memory data storage for swift read and write operations. Versatile in handling various data structures like strings, lists, sets, and hashes, Redis excels in atomic operations, ensuring data integrity.

Used widely for caching, real-time analytics, and more, Redis supports persistence to disk and offers publish/subscribe messaging for real-time communication. With master-slave replication and partitioning capabilities, Redis scales seamlessly, making it ideal for diverse workloads.

Redis data manipulation involves interacting with the various data structures provided by Redis. Here are the main topics for Redis data manipulation

Execution:

1) Set a string value as your name

```
127.0.0.1:6379> SET mystr "Myname"
OK
```

2) Get the string value which is your name

```
127.0.0.1:6379> GET mystr
"Myname"
```

3) Set multiple fields in a hash i.e. set your age and email id

```
127.0.0.1:6379> HMSET myhasht Name "Keval" Email "test@gmail.com" Age 22
OK
```

4) Get specific field from a hash (get your name)

```
127.0.0.1:6379> HMGET myhasht Name
1) "Keval"
```

5) Add an element to the end of a list (add project as a test in your profile)

```
127.0.0.1:6379> LPUSH Profile "Email:test@gmail.com" "Name:Keval"
(integer) 2
127.0.0.1:6379> RPUSH Profile "Project:test"
(integer) 3
```

6) Get all elements in a list

```
127.0.0.1:6379> LRANGE Profile  0 -1
1) "Name:Keval"
2) "Email:test@gmail.com"
3) "Project:test"
127.0.0.1:6379>
```

7) Add elements to a set

```
127.0.0.1:6379> SADD myset "element1" "element2" "element3" "element4"
(integer) 4
```

8) Get all elements in a set

```
127.0.0.1:6379> SMEMBERS myset
1) "element4"
2) "element2"
3) "element1"
4) "element3"
127.0.0.1:6379>
```

9) Add elements with scores to a sorted set

```
127.0.0.1:6379> ZADD sortedset 1 English 2 CS 3 Maths 4 IT
(integer) 4
```

10) Get the rank of an element in a sorted set (In the case for subject English)

```
127.0.0.1:6379> ZRANK sortedset English
(integer) 0
```

11) Get a range of elements from a sorted set by score

```
127.0.0.1:6379> ZRANGE sortedset 0 -1 WITHSCORES
1) "English"
2) "1"
3) "CS"
4) "2"
5) "Maths"
6) "3"
7) "IT"
8) "4"
```

12) Get the score of a member in a sorted set

```
127.0.0.1:6379> zscore sortedset CS
"2"
```

13) Get an element from a list by index

```
127.0.0.1:6379> lindex Profile 2
"Project:test"
```

14) Get the number of members in a set

```
127.0.0.1:6379> scard myset
(integer) 4
```

Conclusion: - The Above Practical RedisDataManipulation has been performed successfully.

Practical - 07

Aim: Implementing Indexing in MongoDB

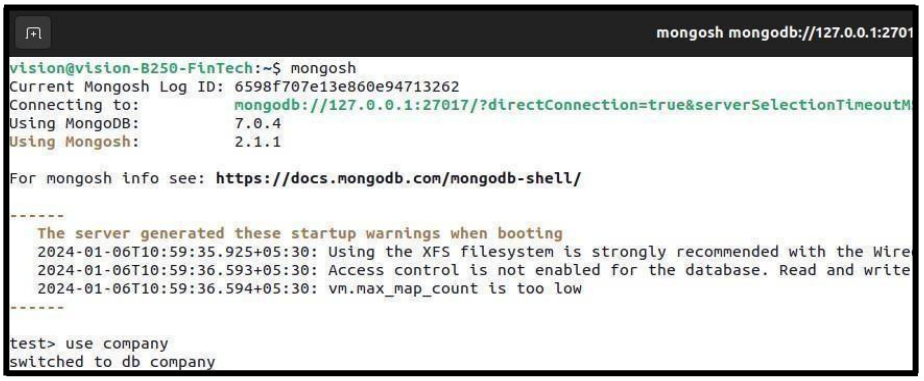
a) Create an index on a specific field in a MongoDB collection.
b) Measure the impact of indexing on query performance.

Description:

Indexing in MongoDB:
MongoDB uses indexing in order to make the query processing more efficient. If there isno indexing, then the MongoDB must scan every document in the collection and retrieveonly those documents that matchthe query. Indexes are special data structures that store some information related to the documents such that it becomes easy for MongoDB to find the right data file. The indexes are ordered by the value of the field specified in the index.

Execution:

1. Create a MongoDB database named ‘company’



2. Create an employee collection with following attributes: {name, salary,department, position}.

```
company> db.createCollection("employee")
{ ok: 1 }
company> show collections
employee
company> db.employee.insertMany([
...   {
...     name: "Deepak",
...     salary: 45000,
...     department: "Backend Developer",
...     position: "Tech Lead"
...   },
...   {
...     name: "Debashish",
...     salary: 52000,
...     department: "UI/UX Developer",
...     position: "Manager"
...   },
...   {
...     name: "Vinit",
...     salary: 68000,
...     department: "Data Analyst",
...     position: "Manager"
...   },
...   {
...     name: "Pooja",
...     salary: 48000,
...     department: "UI/UX Developer",
...     position: "Developer"
...   },
...   {
...     name: "Rohit",
...     salary: 50000,
...     department: "Data Analyst",
...     position: "Developer"
...   },
...   {
...     name: "Narayan",
...     salary: 70000,
...     department: "Data Analyst",
...     position: "Developer"
...   },
...   {
...     name: "Bijisha",
...     salary: 42000,
...     department: "Web Developer",
...     position: "Developer"
...   },
...   {
...     name: "Sunil",
```

3. Create an index on the salary field

```
company> db.employee.createIndex({salary:1});
salary_1
```

4. Search for employees with a salary greater than 50,000 without using indexes.

```
company> db.employee.find({salary:{$gt:50000}});
[
  {
    _id: ObjectId('6598fa7ee13e860e94713264'),
    name: 'Debashish',
    salary: 52000,
    department: 'UI/UX Developer',
    position: 'Manager'
  },
  {
    _id: ObjectId('6598fa7ee13e860e9471326b'),
    name: 'Ganesh',
    salary: 65000,
    department: 'Web Developer',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713265'),
    name: 'Vinit',
    salary: 68000,
    department: 'Data Analyst',
    position: 'Manager'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713268'),
    name: 'Narayan',
    salary: 70000,
    department: 'Data Analyst',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e9471326a'),
    name: 'Sunil',
    salary: 70000,
    department: 'Data Analyst',
    position: 'Developer'
  }
]
```

5. Search for employees with a salary greater than 50,000 using indexes (use of index using the hint() method. After executing these queries, examine the output's executionStats):

```
company> db.employee.find({salary:{$gt:50000}}).hint({salary:1}).explain("executionStats");
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'company.employee',
    indexFilterSet: false,
    parsedQuery: { salary: { '$gt': 50000 } },
    queryHash: '29984A79',
    planCacheKey: '9058732A',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 2,
        inputStage: {
          stage: 'IXSCAN',
          planNodeId: 1,
          keyPattern: { salary: 1 },
          indexName: 'salary_1',
          isMultiKey: false,
          multiKeyPaths: { salary: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { salary: [ '(50000, inf.0]' ] }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s11 env: { s3 = 1704524936669 (NOW), s2 = Nothing (SEARCH_META), s5 = KS(2D0186A0FE04), s10 = {"salary" :
33FFFFFFFFFFFFFFFFFE04) }',
        stages: '[2] nlj inner [] [s4, s7, s8, s9, s10] \n' +
          '    left \n' +
          '        [1] cfilter {(exists(s5) && exists(s6))} \n' +
          '        [1] ixseek s5 s6 s9 s4 s7 s8 [] @"d8ee9843-28df-43ba-853c-0f0426707559" @"salary_1" true \n' +
          '    right \n' +
          '        [2] limit 1 \n' +
          '        [2] seek s4 s11 s12 s7 s8 s9 s10 [] @"d8ee9843-28df-43ba-853c-0f0426707559" true false \n'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 5,
    executionTimeMillis: 0,
    totalKeysExamined: 5,
    totalDocsExamined: 5,
    executionStages: {
      stage: 'nlj',
      planNodeId: 2,
      nReturned: 5,
```

6. Query to Find Employees in a Specific Department (use of index using the hint() method. After executing these queries,examine the output's executionStats) :

```
company> db.employee.find({department: "Data Analyst"}).hint({salary:1}).explain("executionStats");
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'company.employee',
    indexFilterSet: false,
    parsedQuery: { department: { '$eq': 'Data Analyst' } },
    queryHash: '2DD6FB50',
    planCacheKey: '675C919B',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 2,
        filter: { department: { '$eq': 'Data Analyst' } },
        inputStage: {
          stage: 'IXSCAN',
          planNodeId: 1,
          keyPattern: { salary: 1 },
          indexName: 'salary_1',
          isMultiKey: false,
          multiKeyPaths: { salary: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { salary: [ '[MinKey, MaxKey]' ] }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s9 env: { s3 = 1704525172478 (NOW), s8 = {"salary" : 1}, s12 = "Data Analyst", s1 = TimeZoneDataba
        stages: '[2] filter {traverseF(s11, lambda(l1.0) { ((l1.0 == s12) ?: false) }, false)} \n' +
          '[2] nlj inner [] [s4, s5, s6, s7, s8] \n' +
          '    left \n' +
          '        [1] ixseek KS(0A0104) KS(F0FE04) s7 s4 s5 s6 lowPriority [] @"d8ee9843-28df-43ba-853c-0f0426707559" @"sa
          '    right \n' +
          '        [2] limit 1 \n' +
          '        [2] seek s4 s9 s10 s5 s6 s7 s8 [s11 = department] @"d8ee9843-28df-43ba-853c-0f0426707559" true false \n'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 9,
    totalDocsExamined: 9,
    executionStages: {
      stage: 'filter',
      planNodeId: 2,
      nReturned: 4,
```

7. Query to Find Employees with a Salary Range(use of index using the hint() method. After executing these queries, examine the output's executionStats) :

```
}
company> db.employee.find({salary: {$gte: 50000, $lte: 67000}}).hint({salary:1});
[
  {
    _id: ObjectId('6598fa7ee13e860e94713267'),
    name: 'Rohit',
    salary: 50000,
    department: 'Data Analyst',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713264'),
    name: 'Debashish',
    salary: 52000,
    department: 'UI/UX Developer',
    position: 'Manager'
  },
  {
    _id: ObjectId('6598fa7ee13e860e9471326b'),
    name: 'Ganesh',
    salary: 65000,
    department: 'Web Developer',
    position: 'Developer'
  }
]
```

8. Query to Find Employees with a Specific Position(use of index using the hint() method. After executing these queries, examine the output's executionStats)

```
}
company> db.employee.find({position: "Developer"}).hint({salary:1});
[
  {
    _id: ObjectId('6598fa7ee13e860e94713269'),
    name: 'Bijisha',
    salary: 42000,
    department: 'Web Developer',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713266'),
    name: 'Pooja',
    salary: 48000,
    department: 'UI/UX Developer',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713267'),
    name: 'Rohit',
    salary: 50000,
    department: 'Data Analyst',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e9471326b'),
    name: 'Ganesh',
    salary: 65000,
    department: 'Web Developer',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e94713268'),
    name: 'Narayan',
    salary: 70000,
    department: 'Data Analyst',
    position: 'Developer'
  },
  {
    _id: ObjectId('6598fa7ee13e860e9471326a'),
    name: 'Sunil',
    salary: 70000,
    department: 'Data Analyst',
    position: 'Developer'
  }
]
```

9. Query to Find Employees with a Specific Position(use of index using the hint() method. After executing these queries, examine the output's executionStats)

```
company> db.employee.find({position: "Developer"}).hint({salary:1}).explain("executionStats");
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'company.employee',
    indexFilterSet: false,
    parsedQuery: { position: { '$eq': 'Developer' } },
    queryHash: '4AB7019E',
    planCacheKey: '058F94D0',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 2,
        filter: { position: { '$eq': 'Developer' } },
        inputStage: {
          stage: 'IXSCAN',
          planNodeId: 1,
          keyPattern: { salary: 1 },
          indexName: 'salary_1',
          isMultiKey: false,
          multiKeyPaths: { salary: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { salary: [ '[MinKey, MaxKey]' ] }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s9 env: { s2 = Nothing (SEARCH_META), s3 = 1704744551486 (NOW), s1 = TimeZone
        stages: '[2] filter {traverseF(s11, lambda(l1.0) { ((l1.0 == s12) ?: false) }, false)} \n' +
          '[2] nlj inner [] [s4, s5, s6, s7, s8] \n' +
          '    left \n' +
          '        [1] ixseek KS(0A0104) KS(F0FE04) s7 s4 s5 s6 lowPriority [] @"d8ee9843-28df-43ba-85
          '    right \n' +
          '        [2] limit 1 \n' +
          '        [2] seek s4 s9 s10 s5 s6 s7 s8 [s11 = position] @"d8ee9843-28df-43ba-853c-0f0426707
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 6,
    executionTimeMillis: 0,
    totalKeysExamined: 9,
    totalDocsExamined: 9,
    executionStages: {
      stage: 'filter',
      planNodeId: 2,
      nReturned: 6,
```

10. Query to Find Employees with a Salary Above a Threshold in a SpecificDepartment(use of indexusing the hint() method. After executing these queries, examine the output's executionStats).

```
]
company> db.employee.find({salary:{$gt:50000},position: "Developer"}).hint({salary:1}).explain("executionSt
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'company.employee',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [
        { position: { '$eq': 'Developer' } },
        { salary: { '$gt': 50000 } }
      ]
    },
    queryHash: 'A5F607E0',
    planCacheKey: '4EDD1B16',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 2,
        filter: { position: { '$eq': 'Developer' } },
        inputStage: {
          stage: 'IXSCAN',
          planNodeId: 1,
          keyPattern: { salary: 1 },
          indexName: 'salary_1',
          isMultiKey: false,
          multiKeyPaths: { salary: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { salary: [ '(50000, inf.0]' ] }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s11 env: { s3 = 1704744601659 (NOW), s2 = Nothing (SEARCH_META), s5 = KS(2D0186A0F
eveloper", s6 = KS(33FFFFFFFFFFFFFFFFE04) }',
        stages: '[2] filter {traverseF(s13, lambda(l1.0) { ((l1.0 == s14) ?: false) }, false)} \n' +
          '[2] nlj inner [] [s4, s7, s8, s9, s10] \n' +
          '    left \n' +
          '        [1] cfilter {(exists(s5) && exists(s6))} \n' +
          '        [1] ixseek s5 s6 s9 s4 s7 s8 [] @"d8ee9843-28df-43ba-853c-0f0426707559" @"salary_1" true
          '    right \n' +
          '        [2] limit 1 \n' +
          '        [2] seek s4 s11 s12 s7 s8 s9 s10 [s13 = position] @"d8ee9843-28df-43ba-853c-0f0426707559
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 3,
```

Practical – 08

Aim: Data Storage in Redis

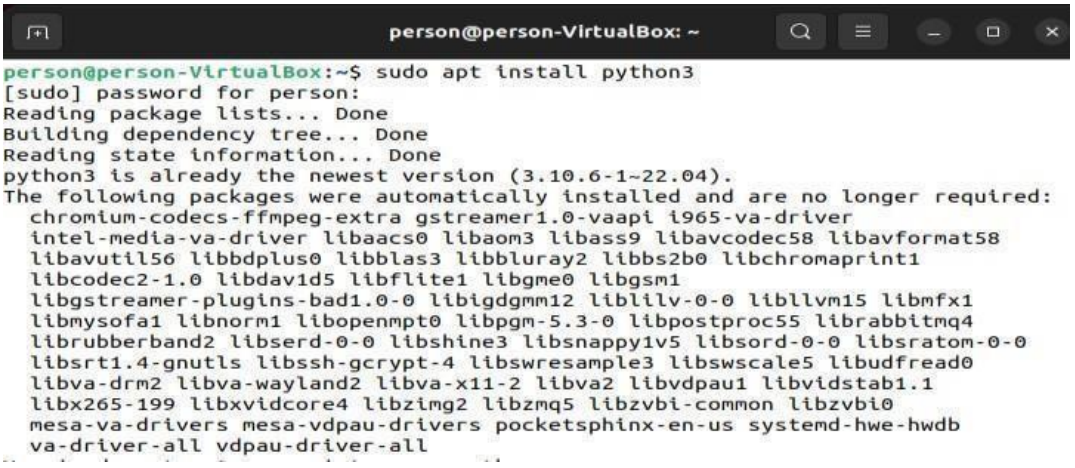
a) Implement caching functionality using Redis as a cache Store.

b) Store and retrieve data from Redis cache using appropriate commands.

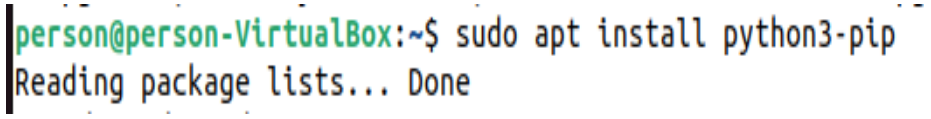
Description: - Implement caching functionality using Redis in a hypothetical programming language. Keep in mind that the exact implementation details may vary depending on the programming language you are using, as different languages may have different Redis client libraries.

Execution:

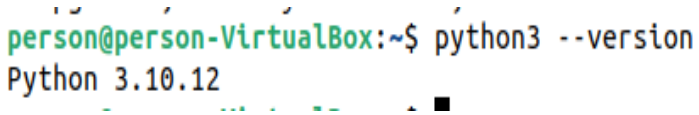
1) Implement caching functionality using Redis as a cache store using python.
2) To install Python on Ubuntu, you can use the following steps: -
   Sudo apt install python3



3) sudo apt install python3-pip



4) python3 –version



5) python3 -m pip –version.

```
person@person-VirtualBox:~$ python3 -m pip --version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
```

6) Use a text editor to write your Python script and save the file as redis-functionality.py

Open *redis-functionality.py Save

```
import redis

# Create a Connection with Redis Server
redis_client=redis.StrictRedis(host="localhost",port=6379,db=0)

# create a data into redis
def get_data_from_database(data_id):
        database_data={
                "1":"Data For ID 1",
                "2":"Data For ID 2",
                "3":"Data For ID 3"
        }
        return database_data.get(data_id,None)


# for fetch the data
def get_data_from_cache(data_id):
        cached_data=redis_client.get(data_id)

        if cached_data:
                return client_data.decode('utf-8')
                #convert into text
        else:
                database_data=get_data_from_database(data_id)

                if database_data:
                        redis_client.set(data_id,database_data)

        return database_data

data_id_to_fetch="1"
cached_data=get_data_from_cache(data_id_to_fetch)

if cached_data:
        print(f"Data found in cache: {cached_data}")
else:
        print("Data not found in cache , from database.")
```

Python 2 Tab Width: 8 Ln 37, Col 56 INS

7) To turn the file in the command line type python3 redis-functionality.py(make sure you are in the right folder where you have saved redis- functionality.py).

```
person@person-VirtualBox:~$ python3 redis-functionality.py
Data found in cache: Data For ID 1
```

8) Store and retrieve data from Redis cache using appropriate commands.

- Connecting to Redis

Open redis-practical.py

```
import redis

#step 1: to create Connection
redis_client=redis.StrictRedis(host="localhost",port=6379,db=0)

```

- Storing Data in Redis

```
6
7 #step 2: To store data into redis
8 data_id="1"
9 data_to_store="Pratical for Redis Store and Retrive"
10
11 redis_client.set(data_id,data_to_store)
12
```

- Retrieving Data from Redis

```
15 # step 3:  to fetch the data which is store
16
17 cached_data=redis_client.get(data_id)
18
19 if cached_data:
20         print(f"Data found in Redis: {cached_data.decode('utf-8')}")
21 else:
22         print("Data not found in Redis")
23
```

```
person@person-VirtualBox:~$ python3 redis-practical.py
Data found in Redis: Pratical for Redis Store and Retrive
```

- Checking if a Key Exists

```
24 #step 4 :Checking if key Exists or Not
25
26 key=redis_client.exists(data_id)
27
28 if key:
29         print("Key Exists in Redis")
30 else:
31         print("Key Does Not Exists")
32
```

```
person@person-VirtualBox:~$ python3 redis-practical.py
Key Exists in Redis
```

- Deleting a Key from Redis

```
33 #step 5:delete a key from Redis
34 delete_key=redis_client.delete(data_id)
35
36 if delete_key>0:
37         print(f"{data_id} deleted from Redis")
38 else:
39         print(f"{data_id} was not found in Redis")
```

```
person@person-VirtualBox:~$ python3 redis-practical.py
1 deleted from Redis
```

Conclusion: - The Above Practical Data Storage in Redis has been performed successfully.