CHEMBUR TROMBAY EDUCATION SOCITY'S

N. G. ACHARYA & D. K. MARATHE COLLEGE

OF ARTS, SCIENCE & COMMERCE

Shri. N.G. ACHARYA MARG, CHEMBUR, MUMBAI-71

NAAC ACCREDIATED "A" GRADE

CERTIFICATE

This is to certify that **Mr. Prathamesh Prakash Sath** of M.Sc. Computer Science Part I, Sem-I, **Seat No: - 1290083** has successfully completed the necessary courses of experiments in the **NoSQL Technologies** during the academic year of 2024-2025.

Faculty In-Charge	External Examiner	Head of the Department-CS
(Prof. Anju Manral)		(Prof. Archana Jadhav)

College Seal

NoSQL Technologies

Name :- Prathamesh Prakash Sathe

College Name:- N.G Acharya and D.K Marathe College

Roll No :- 05

Class :- MSC CS

Part :- 1

Index

SR.No	Practical Name	Signature
1	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.	
2	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	
3	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.	
4	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.	
5	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.	
6	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.	
7	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.	
8	Using Google App Engine Data Store a) Create a project in Google App Engine and set up the Data Store. b) Store and retrieve data from the Data Store using the provided API.	

Practical 1

Lab Exercise: Setting up and Exploring MongoDB

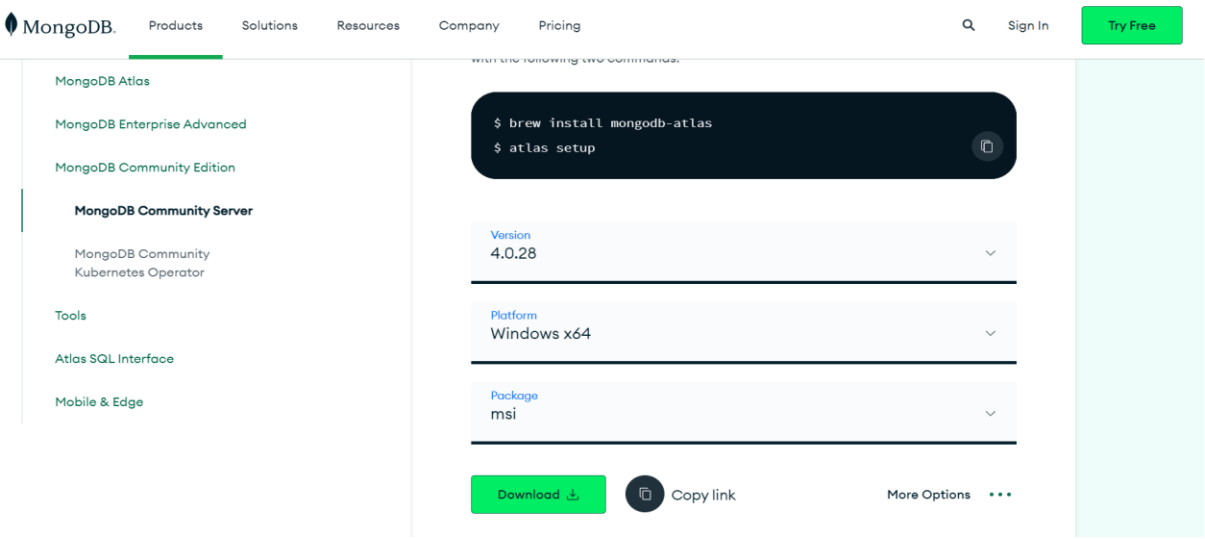
a) Install MongoDB on your local machine or lab server.

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

For Downloading Mongodb->

Goto browser-> mongodb free download

->mongodb community server download->select following



->download the msi and install the file

mongodb-win32-x86_64-2008plus-ssl-4.0.28-signed

☐ Go to cmd - run as admin

c:\program files\mongodb\4.0\bin

- mongod→(To start mongo database)
- mongo -(To start mongo shell)

- Create database orders in mongodb.

  use>Use orders
- Create two collections products and order details in your database.

> db

orders

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

```
student  0.000GB
> db.createCollection('Product')
{ "ok" : 1 }
> db.createCollection('order')
```

b) Create a new MongoDB database and collection.

```
db.createCollection('Product')
db.createCollection('Order')
```

c) Insert sample data into the collection.

- Add 5 product details in products collection. Add following information for product. pid, ProdDescr, MFR, Category & Uprice.

```
> db.Product.insert({ pid : 1, descr:"Mobile",
mfr:"Samsung",category:"Electronic",uprice:30000});
WriteResult({ "nInserted" : 1 })
> db.Product.insert({ pid : 2, descr:"TAB",
mfr:"Samsung",category:"Electronic",uprice:50000});
WriteResult({ "nInserted" : 1 })
> db.Product.insert({ pid : 3, descr:"Refrigerator",
mfr:"LG",category:"Electronic",uprice:45000});
WriteResult({ "nInserted" : 1 })
> db.Product.insert([{ pid : 4, descr:"Cricket Kit",
mfr:"Adidas",category:"Sports",uprice:5000},{ pid : 5, descr:"Shoes",
mfr:"Nike",category:"Sports",uprice:450}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
```

```
        "upserted" : [ ]
    })
```

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

- List all the documents of both the collections in JSON format.
  > db.Product.find().pretty()

```
{
        "_id" : ObjectId("61de99f80e531b700abf0fe2"),
        "pid" : 1,
        "descr" : "Mobile",
        "mfr" : "Samsung",
        "category" : "Electronic",
        "uprice" : 30000
}
{
        "_id" : ObjectId("61de9a1d0e531b700abf0fe3"),
        "pid" : 2,
        "descr" : "TAB",
        "mfr" : "Samsung",
        "category" : "Electronic",
        "uprice" : 50000
}
{
        "_id" : ObjectId("61de9a560e531b700abf0fe4"),
        "pid" : 3,
        "descr" : "Refrigerator",
        "mfr" : "LG",
```

```
        "category" : "Electronic",
        "uprice" : 45000
}
{
        "_id" : ObjectId("61de9af90e531b700abf0fe5"),
        "pid" : 4,
        "descr" : "Cricket Kit",
        "mfr" : "Adidas",
        "category" : "Sports",
        "uprice" : 5000
}
{
        "_id" : ObjectId("61de9af90e531b700abf0fe6"),
        "pid" : 5,
        "descr" : "Shoes",
        "mfr" : "Nike",
        "category" : "Sports",
        "uprice" : 450
}
> db.order.find().pretty()
{
        "_id" : ObjectId("61de9b9f0e531b700abf0fe7"),
        "ono" : 101,
        "odate" : "1-jan-22",
        "cust" : "abc",
        "pid" : 1,
        "qty" : 1,
        "oamt" : 29000
}
{
```

```
        "_id" : ObjectId("61de9bbb0e531b700abf0fe8"),
        "ono" : 102,
        "odate" : "1-jan-22",
        "cust" : "pqr",
        "pid" : 1,
        "qty" : 2,
        "oamt" : 55000
}
{
        "_id" : ObjectId("61de9bd70e531b700abf0fe9"),
        "ono" : 103,
        "odate" : "1-jan-22",
        "cust" : "pqr",
        "pid" : 5,
        "qty" : 2,
        "oamt" : 750
}
{
        "_id" : ObjectId("61de9bf40e531b700abf0fea"),
        "ono" : 104,
        "odate" : "11-jan-22",
        "cust" : "xyz",
        "pid" : 4,
        "qty" : 1,
        "oamt" : 1750
}
{
        "_id" : ObjectId("61de9c180e531b700abf0feb"),
        "ono" : 105,
        "odate" : "5-jan-22",
```

```
        "cust" : "xyz",
        "pid" : 3,
        "qty" : 1,
        "oamt" : 40050
}
```

Practical 2

Interacting with Redis

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

Link to download Redis on windows

=> search for - redis for windows 3.0.504 download

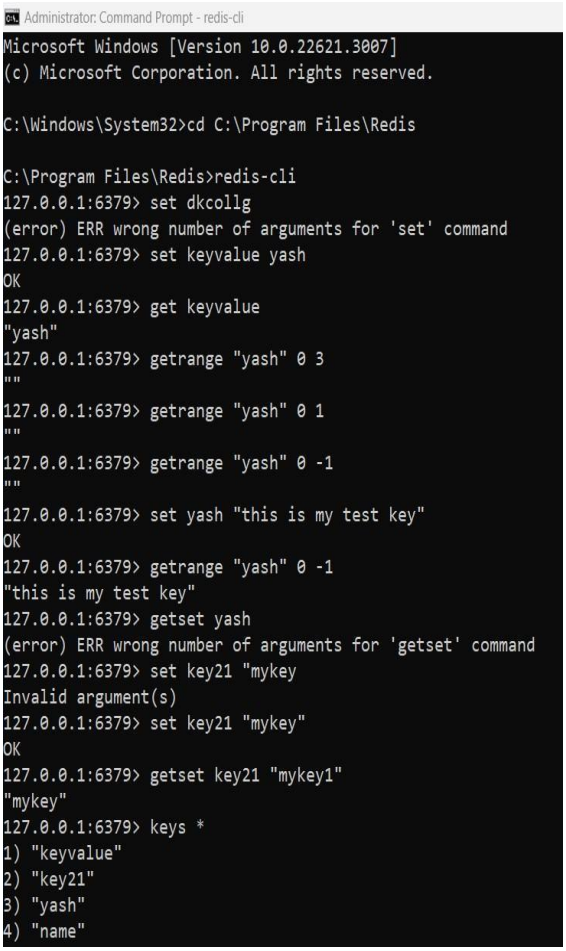
or

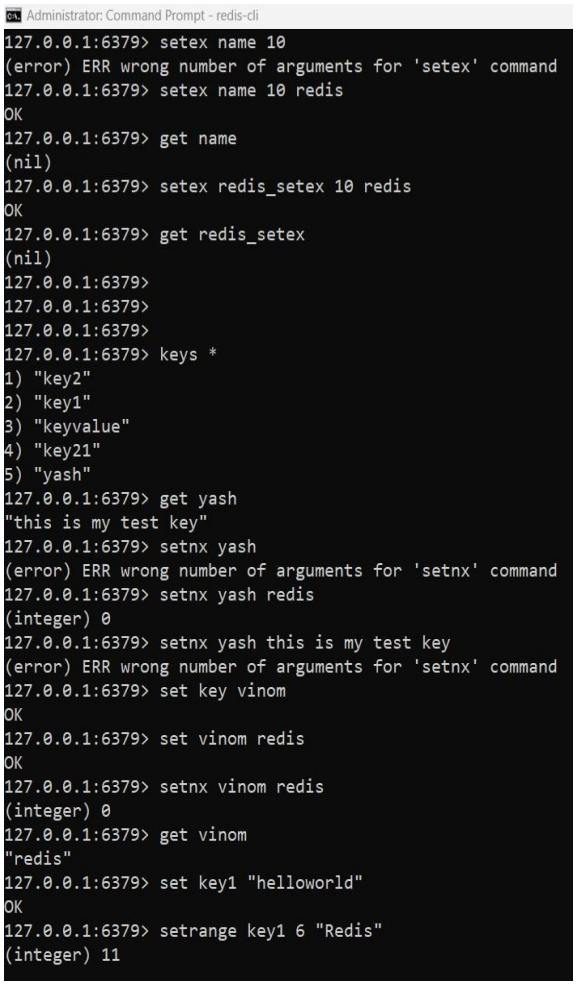
download from

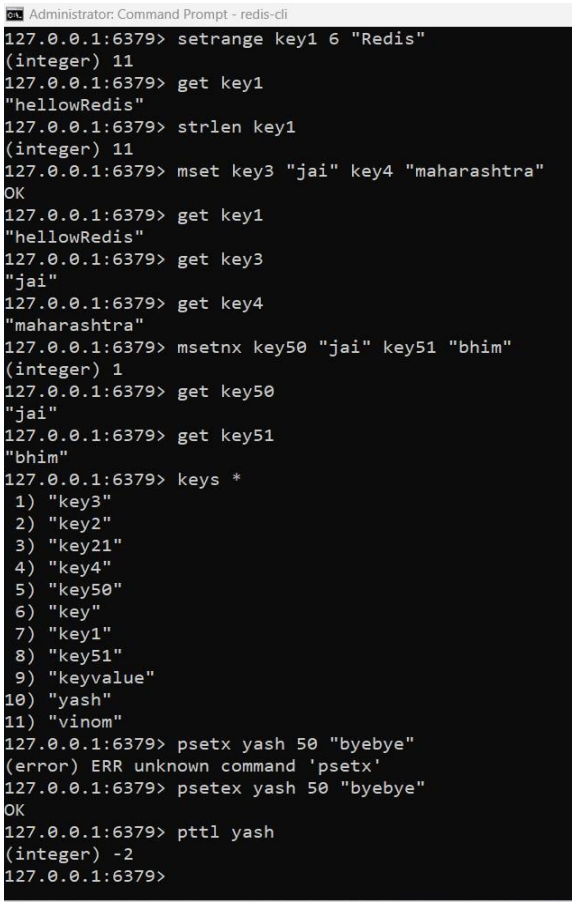
https://github.com/microsoftarchive/redis/releases
download =>`Redis-x64-3.0.504.msi`
Install the redis msi

b) Store and retrieve data in Redis using various data structures like Strings, Lists, and Sets.

1) Redis String Commands

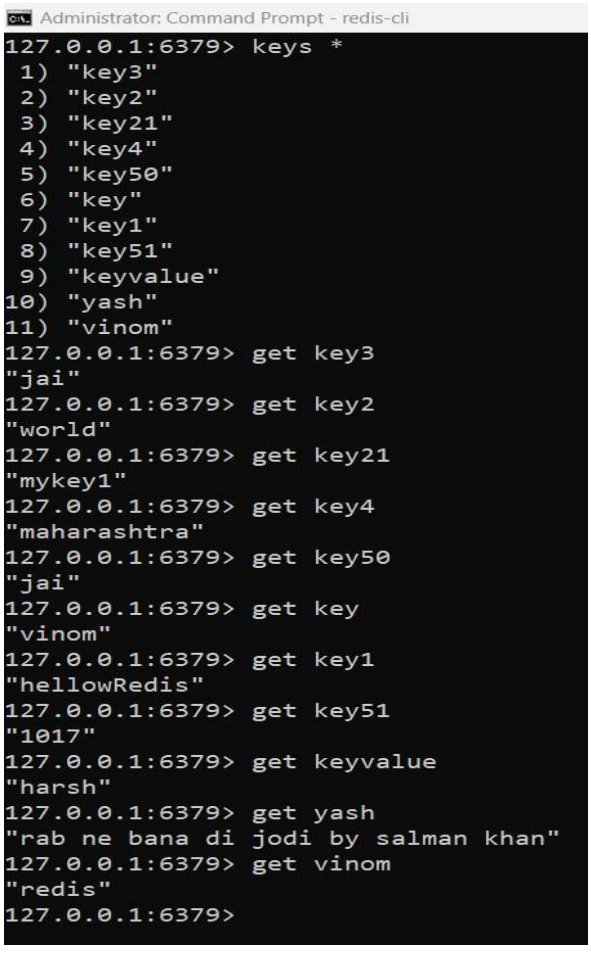

Administrator: Command Prompt - redis-cli
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\System32>cd C:\Program Files\Redis
C:\Program Files\Redis>redis-cli
127.0.0.1:6379> set dkcollg
(error) ERR wrong number of arguments for 'set' command
127.0.0.1:6379> set keyvalue yash
OK
127.0.0.1:6379> get keyvalue
"yash"
127.0.0.1:6379> getrange "yash" 0 3
""
127.0.0.1:6379> getrange "yash" 0 1
""
127.0.0.1:6379> getrange "yash" 0 -1
""
127.0.0.1:6379> set yash "this is my test key"
OK
127.0.0.1:6379> getrange "yash" 0 -1
"this is my test key"
127.0.0.1:6379> getset yash
(error) ERR wrong number of arguments for 'getset' command
127.0.0.1:6379> set key21 "mykey
Invalid argument(s)
127.0.0.1:6379> set key21 "mykey"
OK
127.0.0.1:6379> getset key21 "mykey1"
"mykey"
127.0.0.1:6379> keys *
1) "keyvalue"
2) "key21"
3) "yash"
4) "name"


Administrator: Command Prompt - redis-cli
127.0.0.1:6379> setex name 10
(error) ERR wrong number of arguments for 'setex' command
127.0.0.1:6379> setex name 10 redis
OK
127.0.0.1:6379> get name
(nil)
127.0.0.1:6379> setex redis_setex 10 redis
OK
127.0.0.1:6379> get redis_setex
(nil)
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> keys *
1) "key2"
2) "key1"
3) "keyvalue"
4) "key21"
5) "yash"
127.0.0.1:6379> get yash
"this is my test key"
127.0.0.1:6379> setnx yash
(error) ERR wrong number of arguments for 'setnx' command
127.0.0.1:6379> setnx yash redis
(integer) 0
127.0.0.1:6379> setnx yash this is my test key
(error) ERR wrong number of arguments for 'setnx' command
127.0.0.1:6379> set key vinom
OK
127.0.0.1:6379> set vinom redis
OK
127.0.0.1:6379> setnx vinom redis
(integer) 0
127.0.0.1:6379> get vinom
"redis"
127.0.0.1:6379> set key1 "helloworld"
OK
127.0.0.1:6379> setrange key1 6 "Redis"
(integer) 11

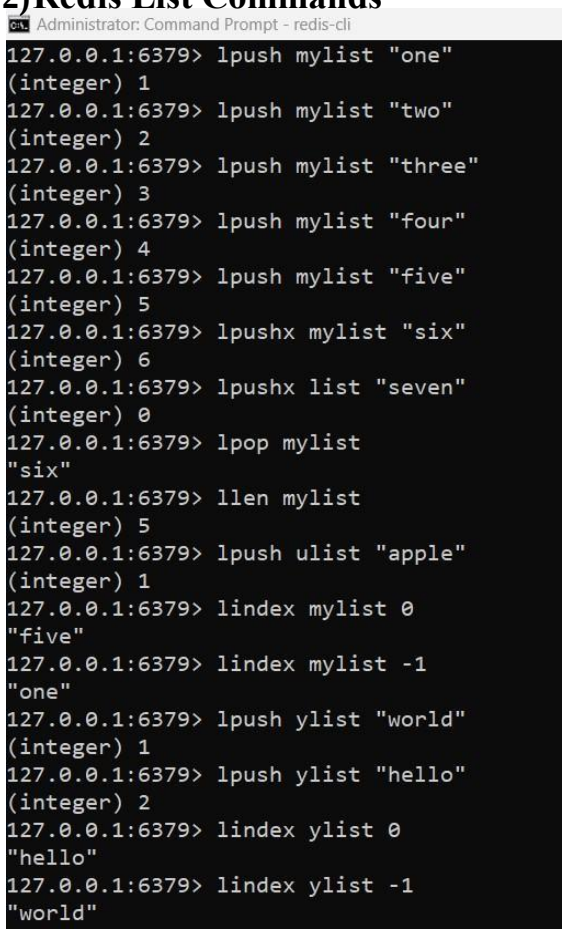

Administrator: Command Prompt - redis-cli
127.0.0.1:6379> setrange key1 6 "Redis"
(integer) 11
127.0.0.1:6379> get key1
"hellowRedis"
127.0.0.1:6379> strlen key1
(integer) 11
127.0.0.1:6379> mset key3 "jai" key4 "maharashtra"
OK
127.0.0.1:6379> get key1
"hellowRedis"
127.0.0.1:6379> get key3
"jai"
127.0.0.1:6379> get key4
"maharashtra"
127.0.0.1:6379> msetnx key50 "jai" key51 "bhim"
(integer) 1
127.0.0.1:6379> get key50
"jai"
127.0.0.1:6379> get key51
"bhim"
127.0.0.1:6379> keys *
1) "key3"
2) "key2"
3) "key21"
4) "key4"
5) "key50"
6) "key"
7) "key1"
8) "key51"
9) "keyvalue"
10) "yash"
11) "vinom"
127.0.0.1:6379> psetx yash 50 "byebye"
(error) ERR unknown command 'psetx'
127.0.0.1:6379> psetex yash 50 "byebye"
OK
127.0.0.1:6379> pttl yash
(integer) -2
127.0.0.1:6379>

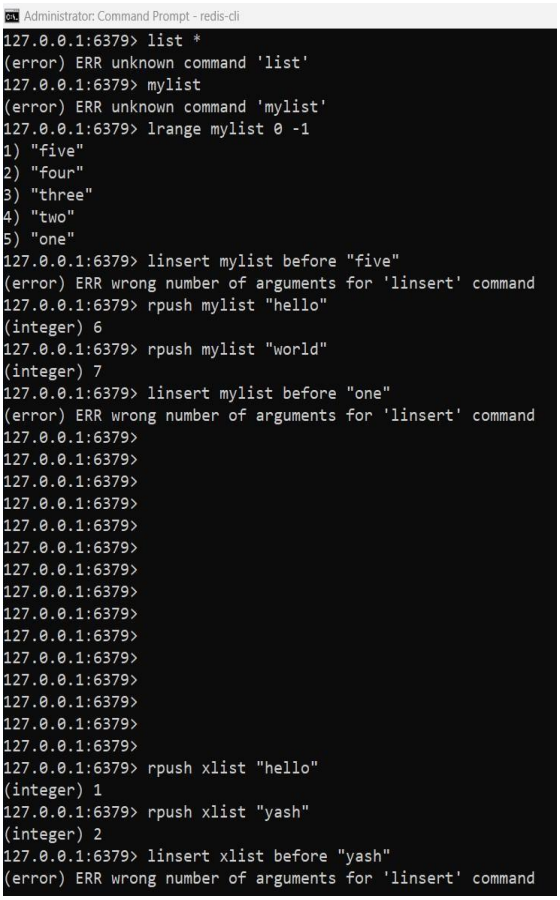
Administrator: Command Prompt - redis-cli

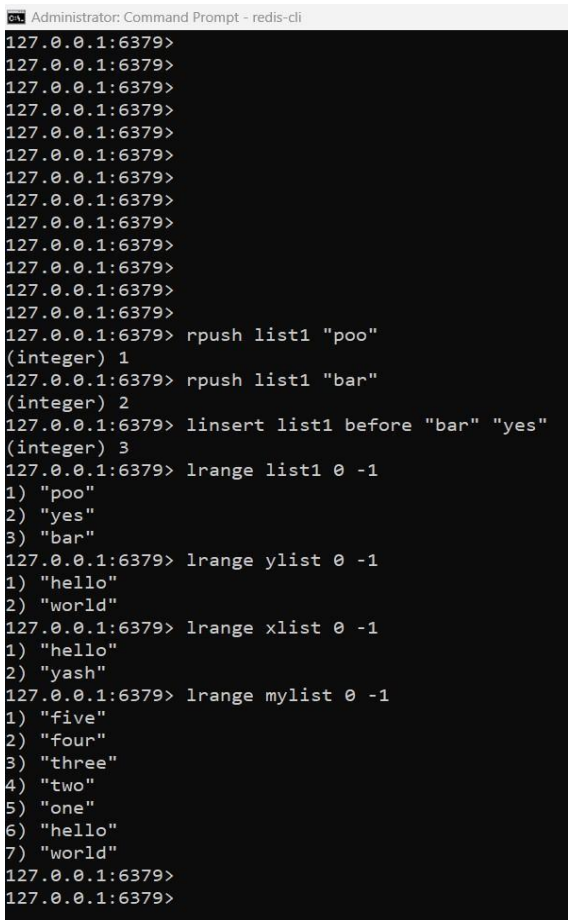
```
127.0.0.1:6379> pttl yash
(integer) -2
127.0.0.1:6379> pttl yash
(integer) -2
127.0.0.1:6379> get yash
(nil)
127.0.0.1:6379> keys *
 1) "key3"
 2) "key2"
 3) "key21"
 4) "key4"
 5) "key50"
 6) "key"
 7) "key1"
 8) "key51"
 9) "keyvalue"
10) "vinom"
127.0.0.1:6379> set key51 1000
OK
127.0.0.1:6379> incr key51
(integer) 1001
127.0.0.1:6379> incrby 5000
(error) ERR wrong number of arguments for 'incrby' command
127.0.0.1:6379> incrby key51 26
(integer) 1027
127.0.0.1:6379> get key51
"1027"
127.0.0.1:6379> decr key51 10
(error) ERR wrong number of arguments for 'decr' command
127.0.0.1:6379> decrby key51 10
(integer) 1017
127.0.0.1:6379> get yash
(nil)
127.0.0.1:6379> set yash "rab ne bana di jodi"
OK
127.0.0.1:6379> append yash " by salman khan"
(integer) 34
127.0.0.1:6379> get yash
"rab ne bana di jodi by salman khan"
```

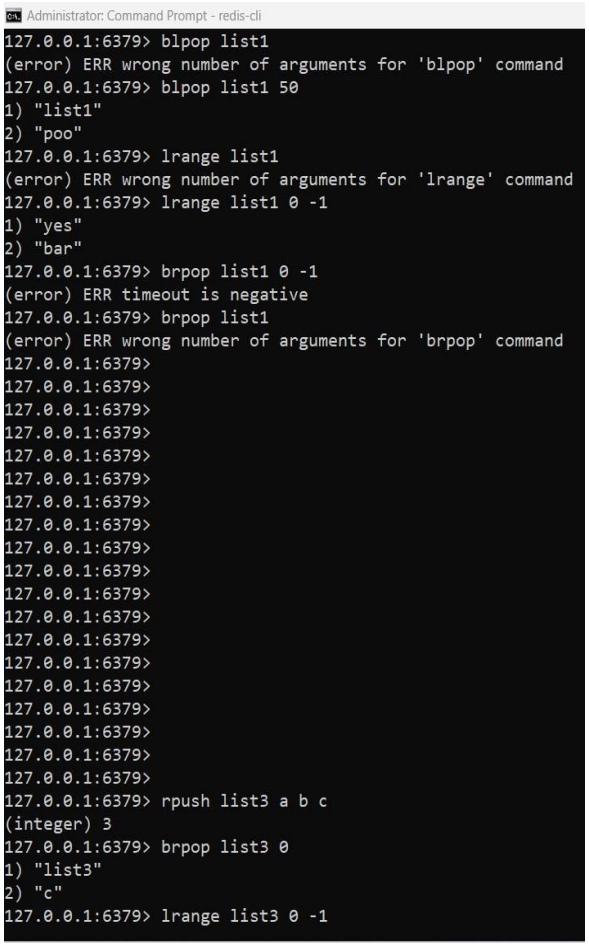


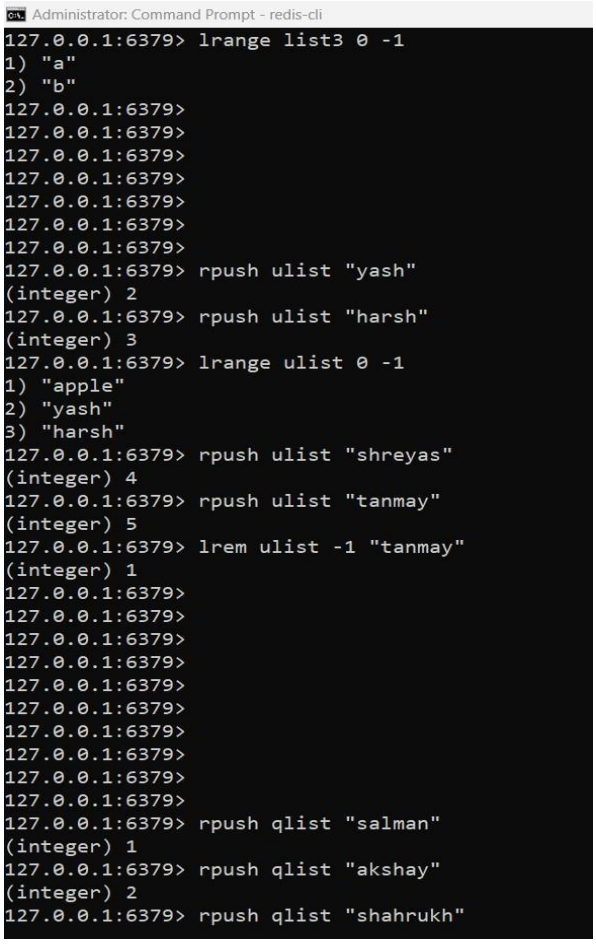
2)Redis List Commands

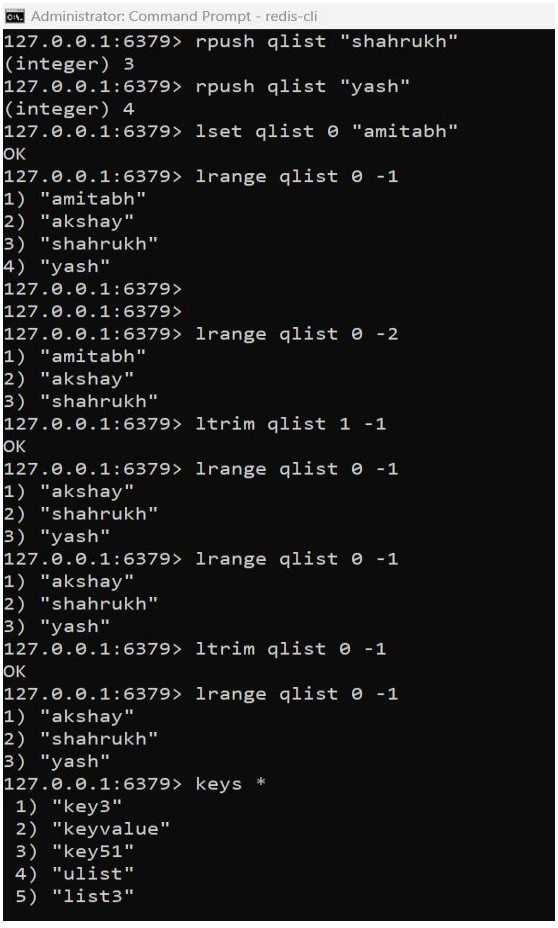




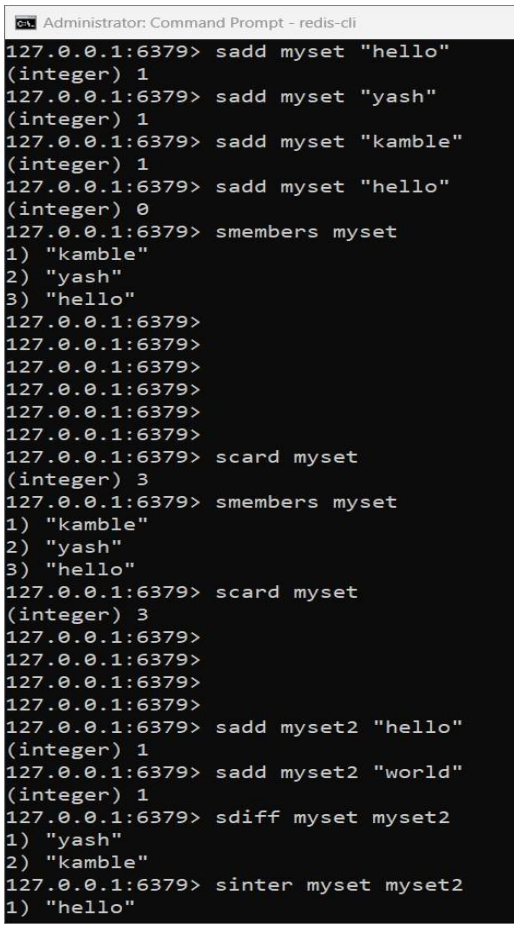

Administrator: Command Prompt - redis-cli
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> rpush list1 "poo"
(integer) 1
127.0.0.1:6379> rpush list1 "bar"
(integer) 2
127.0.0.1:6379> linsert list1 before "bar" "yes"
(integer) 3
127.0.0.1:6379> lrange list1 0 -1
1) "poo"
2) "yes"
3) "bar"
127.0.0.1:6379> lrange ylist 0 -1
1) "hello"
2) "world"
127.0.0.1:6379> lrange xlist 0 -1
1) "hello"
2) "yash"
127.0.0.1:6379> lrange mylist 0 -1
1) "five"
2) "four"
3) "three"
4) "two"
5) "one"
6) "hello"
7) "world"
127.0.0.1:6379>
127.0.0.1:6379>

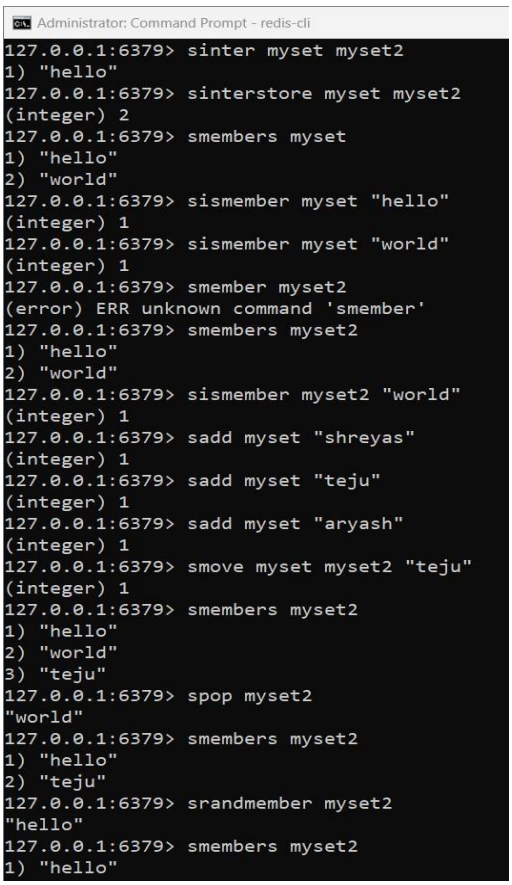

Administrator: Command Prompt - redis-cli
127.0.0.1:6379> blpop list1
(error) ERR wrong number of arguments for 'blpop' command
127.0.0.1:6379> blpop list1 50
1) "list1"
2) "poo"
127.0.0.1:6379> lrange list1
(error) ERR wrong number of arguments for 'lrange' command
127.0.0.1:6379> lrange list1 0 -1
1) "yes"
2) "bar"
127.0.0.1:6379> brpop list1 0 -1
(error) ERR timeout is negative
127.0.0.1:6379> brpop list1
(error) ERR wrong number of arguments for 'brpop' command
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> rpush list3 a b c
(integer) 3
127.0.0.1:6379> brpop list3 0
1) "list3"
2) "c"
127.0.0.1:6379> lrange list3 0 -1

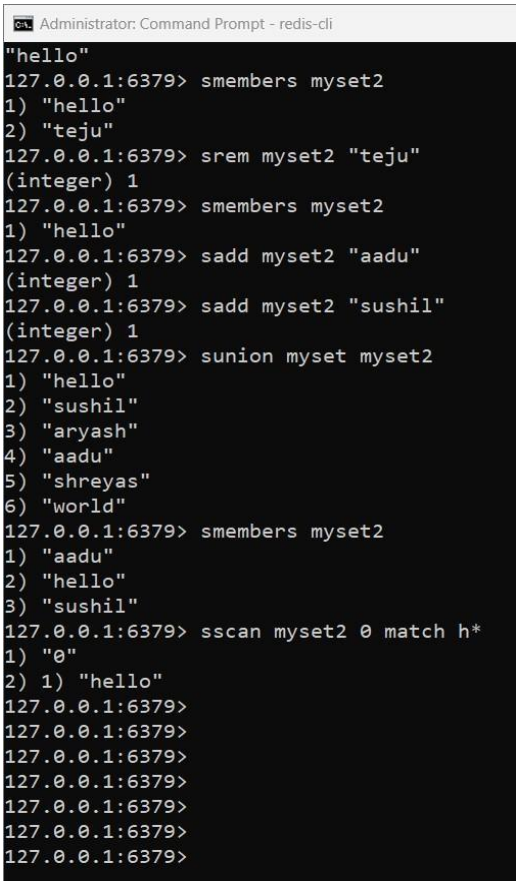

Administrator: Command Prompt - redis-cli
127.0.0.1:6379> lrange list3 0 -1
1) "a"
2) "b"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> rpush ulist "yash"
(integer) 2
127.0.0.1:6379> rpush ulist "harsh"
(integer) 3
127.0.0.1:6379> lrange ulist 0 -1
1) "apple"
2) "yash"
3) "harsh"
127.0.0.1:6379> rpush ulist "shreyas"
(integer) 4
127.0.0.1:6379> rpush ulist "tanmay"
(integer) 5
127.0.0.1:6379> lrem ulist -1 "tanmay"
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> rpush qlist "salman"
(integer) 1
127.0.0.1:6379> rpush qlist "akshay"
(integer) 2
127.0.0.1:6379> rpush qlist "shahrukh"


Administrator: Command Prompt - redis-cli
127.0.0.1:6379> rpush qlist "shahrukh"
(integer) 3
127.0.0.1:6379> rpush qlist "yash"
(integer) 4
127.0.0.1:6379> lset qlist 0 "amitabh"
OK
127.0.0.1:6379> lrange qlist 0 -1
1) "amitabh"
2) "akshay"
3) "shahrukh"
4) "yash"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> lrange qlist 0 -2
1) "amitabh"
2) "akshay"
3) "shahrukh"
127.0.0.1:6379> ltrim qlist 1 -1
OK
127.0.0.1:6379> lrange qlist 0 -1
1) "akshay"
2) "shahrukh"
3) "yash"
127.0.0.1:6379> lrange qlist 0 -1
1) "akshay"
2) "shahrukh"
3) "yash"
127.0.0.1:6379> ltrim qlist 0 -1
OK
127.0.0.1:6379> lrange qlist 0 -1
1) "akshay"
2) "shahrukh"
3) "yash"
127.0.0.1:6379> keys *
 1) "key3"
 2) "keyvalue"
 3) "key51"
 4) "ulist"
 5) "list3"

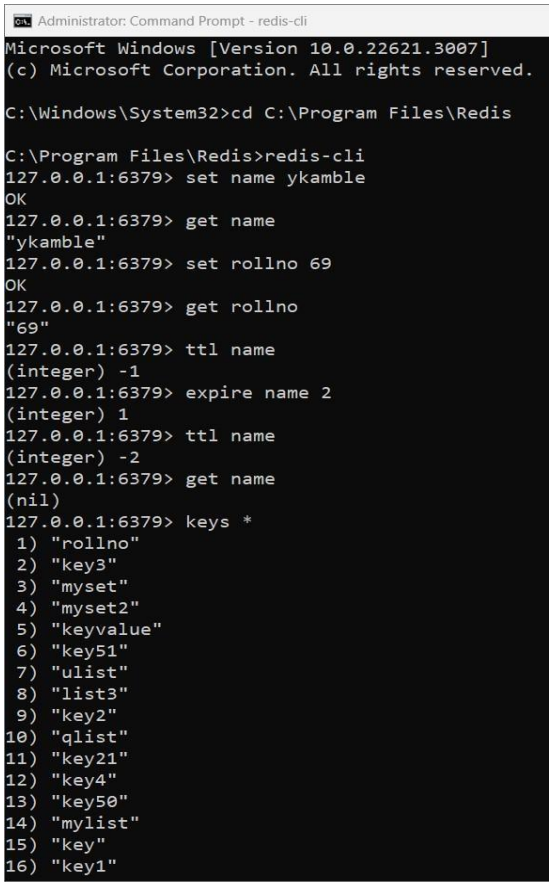
3)Redis Set Commands

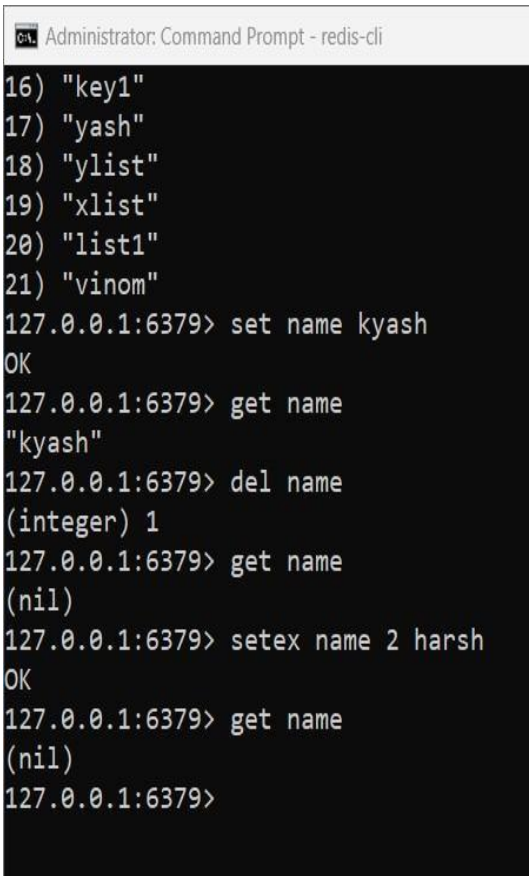





Administrator: Command Prompt - redis-cli
"hello"
127.0.0.1:6379> smembers myset2
1) "hello"
2) "teju"
127.0.0.1:6379> srem myset2 "teju"
(integer) 1
127.0.0.1:6379> smembers myset2
1) "hello"
127.0.0.1:6379> sadd myset2 "aadu"
(integer) 1
127.0.0.1:6379> sadd myset2 "sushil"
(integer) 1
127.0.0.1:6379> sunion myset myset2
1) "hello"
2) "sushil"
3) "aryash"
4) "aadu"
5) "shreyas"
6) "world"
127.0.0.1:6379> smembers myset2
1) "aadu"
2) "hello"
3) "sushil"
127.0.0.1:6379> sscan myset2 0 match h*
1) "0"
2) 1) "hello"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>

3) Basic Commands





Practical 3

Working with HBase

a) Set up an HBase cluster in a lab environment.

Hbase-Installation on Windows

Introduction

In earlier decades, produced data was organized and far less in quantity than now. These types of data were stored using RDMS. However, we now process a vast volume of semi-structured data such as emails, JSONs, XML,.csv files, and so on. HBase was introduced when RDMS failed to store and handle this data.
HBase is a data format equivalent to Google's big table that allows for speedy random access to massive volumes of structured data. HBase is a Java-based open-source, multidimensional, distributed, and scalable NoSQL database that works on top of HDFS (Hadoop Distributed File System). It is intended to hold a massive number of sparse data sets. It enables users to obtain information in real-time.HBase is a column-oriented database that stores data in tables. Only column families are defined in the HBase table structure. The HBase database is divided into families, and each family can contain an infinite number of columns. The column values are successively saved on a disc. Each table cell has a timestamp.
In this blog, we will learn how to set up Hbase in the window operating system.

Prerequisite

Install Java JDK - You can download it from this link. (https://www.oracle.com/java/technologies/downloads/)

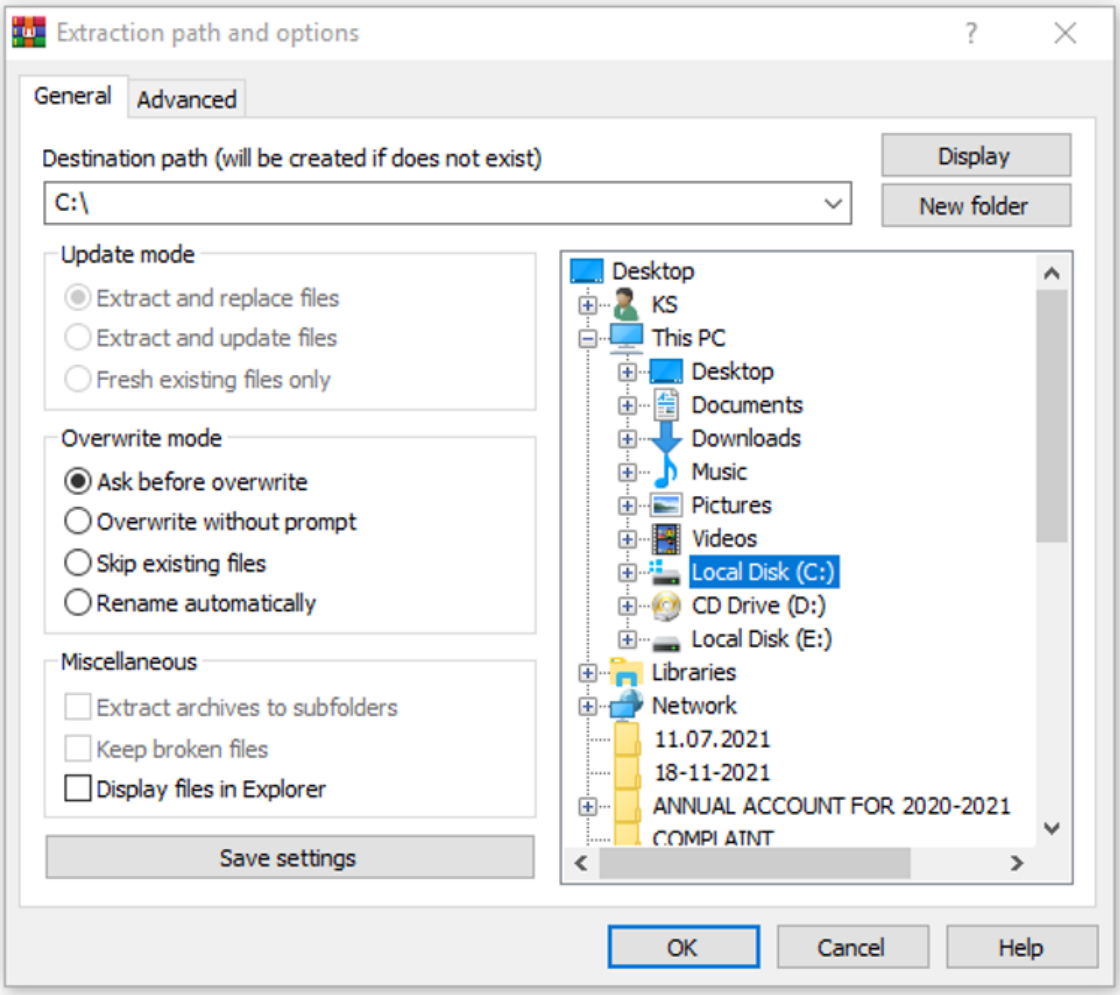
- The Java Development Kit (JDK) is a cross-platform software development environment that includes tools and libraries for creating Java-based software applications and applets.

- Download Hbase - Download Apache Hbase from this link. (https://hbase.apache.org/downloads.html)
- **Hbase version -1.7.1**

Steps

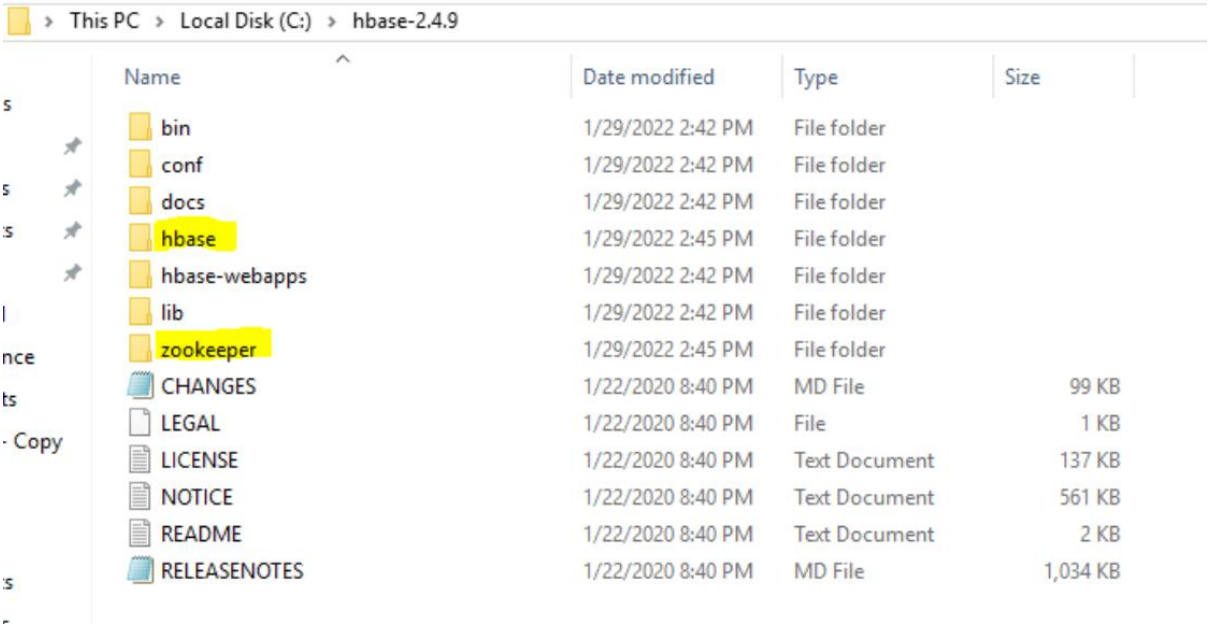
Step-1 (Extraction of files)

Extract all the files in C drive



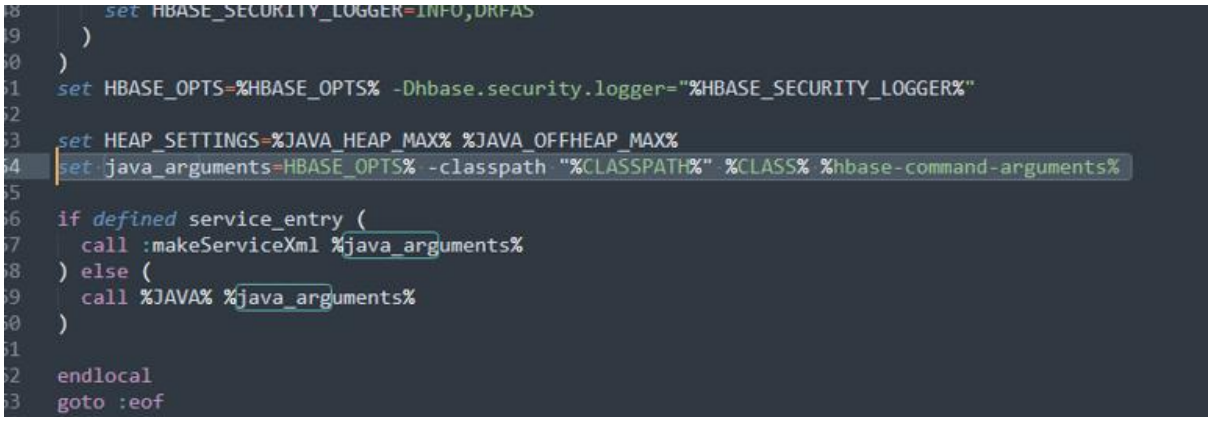
Step-2 (Creating Folder)

Create folders named "hbase" and "zookeeper."



Step-3 (Deleting line in HBase.cmd)

Open hbase.cmd in any text editor.
Search for line **%HEAP_SETTINGS% and remove it.**



Step-4 (Add lines in hbase-env.cmd)

C:\hbase\hbase-1.7.1\conf

Now open hbase-env.cmd, which is in the conf folder in any text editor.
Add the below lines in the file after the comment session.

```
set JAVA_HOME=%JAVA_HOME%
set HBASE_CLASSPATH=%HBASE_HOME%\lib\client-facing-thirdparty\*
set HBASE_HEAPSIZE=8000
set HBASE_OPTS="-XX:+UseConcMarkSweepGC" "-Djava.net.preferIPv4Stack=true"
set SERVER_GC_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-XX:+PrintGCDateStamps"
%HBASE_GC_OPTS%
set HBASE_USE_GC_LOGFILE=true

set HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false" ."-
Dcom.sun.management.jmxremote.authenticate=false"

set HBASE_MASTER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10101"
set HBASE_REGIONSERVER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10102"
set HBASE_THRIFT_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10103"
set HBASE_ZOOKEEPER_OPTS=%HBASE_JMX_BASE% -
Dcom.sun.management.jmxremote.port=10104"
set HBASE_REGIONSERVERS=%HBASE_HOME%\conf\regionservers
set HBASE_LOG_DIR=%HBASE_HOME%\logs
set HBASE_IDENT_STRING=%USERNAME%
set HBASE_MANAGES_ZK=true
```

```
@rem Set environment variables here.

@rem The java implementation to use.  Java 1.8+ required.
@rem set JAVA_HOME=c:\apps\java
set JAVA_HOME=%JAVA_HOME%
set HBASE_CLASSPATH=%HBASE_HOME%\lib\client-facing-thirdparty\*
set HBASE_HEAPSIZE=8000
set HBASE_OPTS="-XX:+UseConcMarkSweepGC" "-Djava.net.preferIPv4Stack=true"
set SERVER_GC_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-XX:+PrintGCDateStamps" %HBASE_GC_OPTS%
set HBASE_USE_GC_LOGFILE=true

set HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false" "-Dcom.sun.management.jmxremote.authenticate=false"

set HBASE_MASTER_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10101"
set HBASE_REGIONSERVER_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10102"
set HBASE_THRIFT_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10103"
set HBASE_ZOOKEEPER_OPTS=%HBASE_JMX_BASE% -Dcom.sun.management.jmxremote.port=10104"
set HBASE_REGIONSERVERS=%HBASE_HOME%\conf\regionservers
set HBASE_LOG_DIR=%HBASE_HOME%\logs
set HBASE_IDENT_STRING=%USERNAME%
set HBASE_MANAGES_ZK=true

@rem Extra Java CLASSPATH elements.  Optional.
@rem set HBASE_CLASSPATH=
```

set JAVA_HOME=”%JAVA_HOME%”

Step-5 (Add the line in Hbase-site.xml)

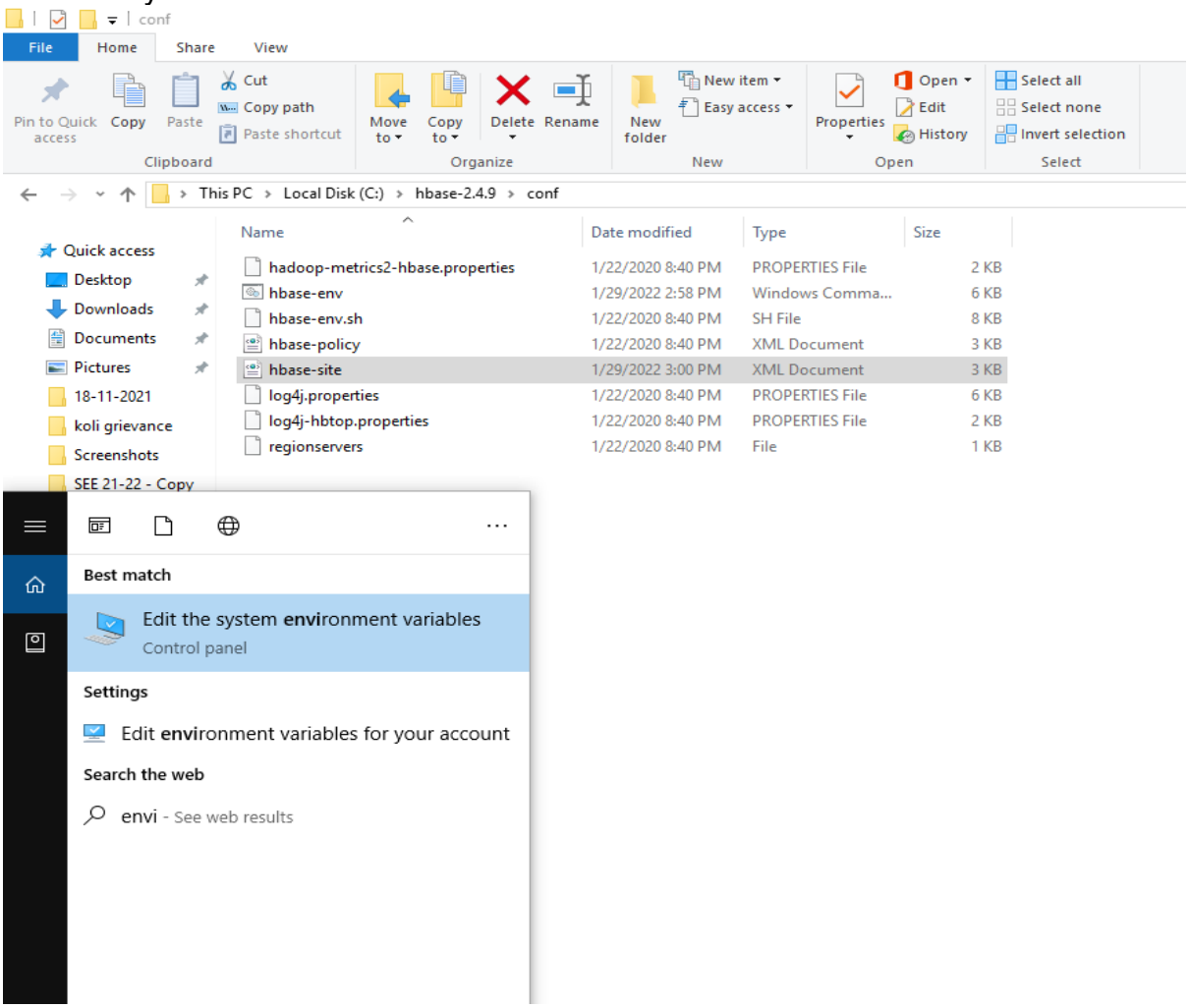
Open hbase-site.xml, which is in the conf folder in any text editor.
Add the lines inside the <configuration> tag.
A distributed HBase entirely relies on Zookeeper (for cluster configuration and management). ZooKeeper coordinates, communicates and distributes state between the Masters and RegionServers in Apache HBase. HBase's design strategy is to use ZooKeeper solely for transient data (that is, for coordination and state communication). Thus, removing HBase's ZooKeeper data affects only temporary operations – data can continue to be written and retrieved to/from HBase.

```
<property>
  <name>hbase.rootdir</name>
  <value>file:///C:/Documents/hbase-1.7.1/hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/C:/Documents/hbase-1.7.1/zookeeper</value>
</property>
<property>
   <name> hbase.zookeeper.quorum</name>
  <value>localhost</value>
</property>
```

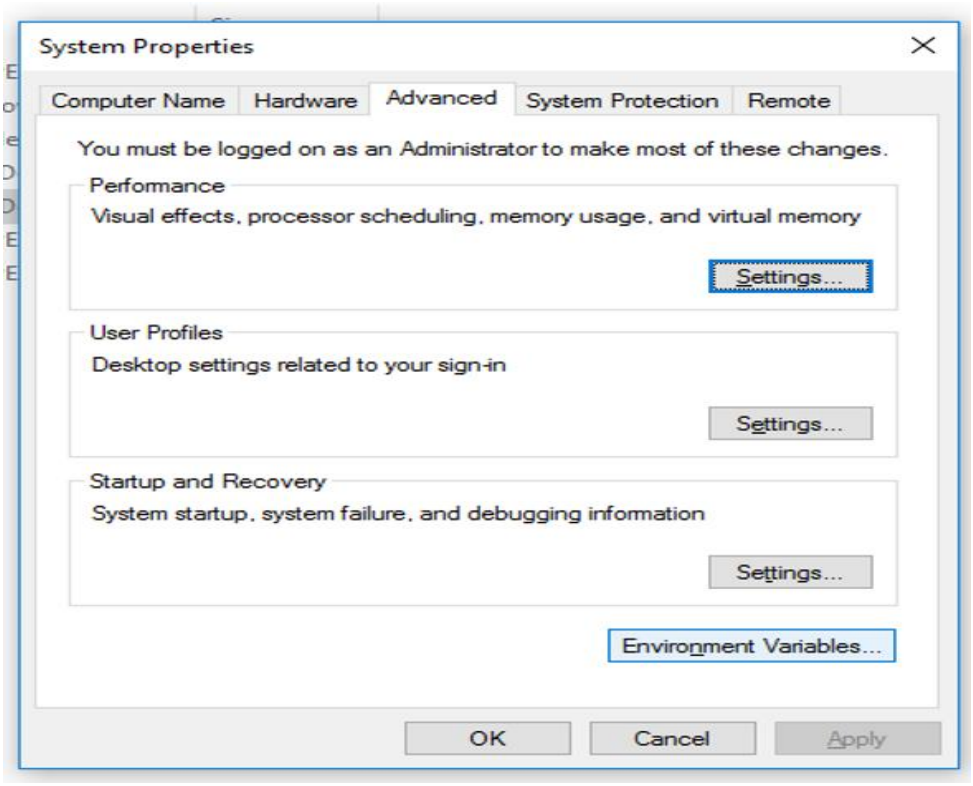
```
    <value>false</value>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>file:///C:/Documents/hbase-2.2.5/hbase</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/C:/Documents/hbase-2.2.5/zookeeper</value>
  </property>
  <property>
    <name> hbase.zookeeper.quorum</name>
    <value>localhost</value>
  </property>
</configuration>
```

Step-6 (Setting Environment Variables)

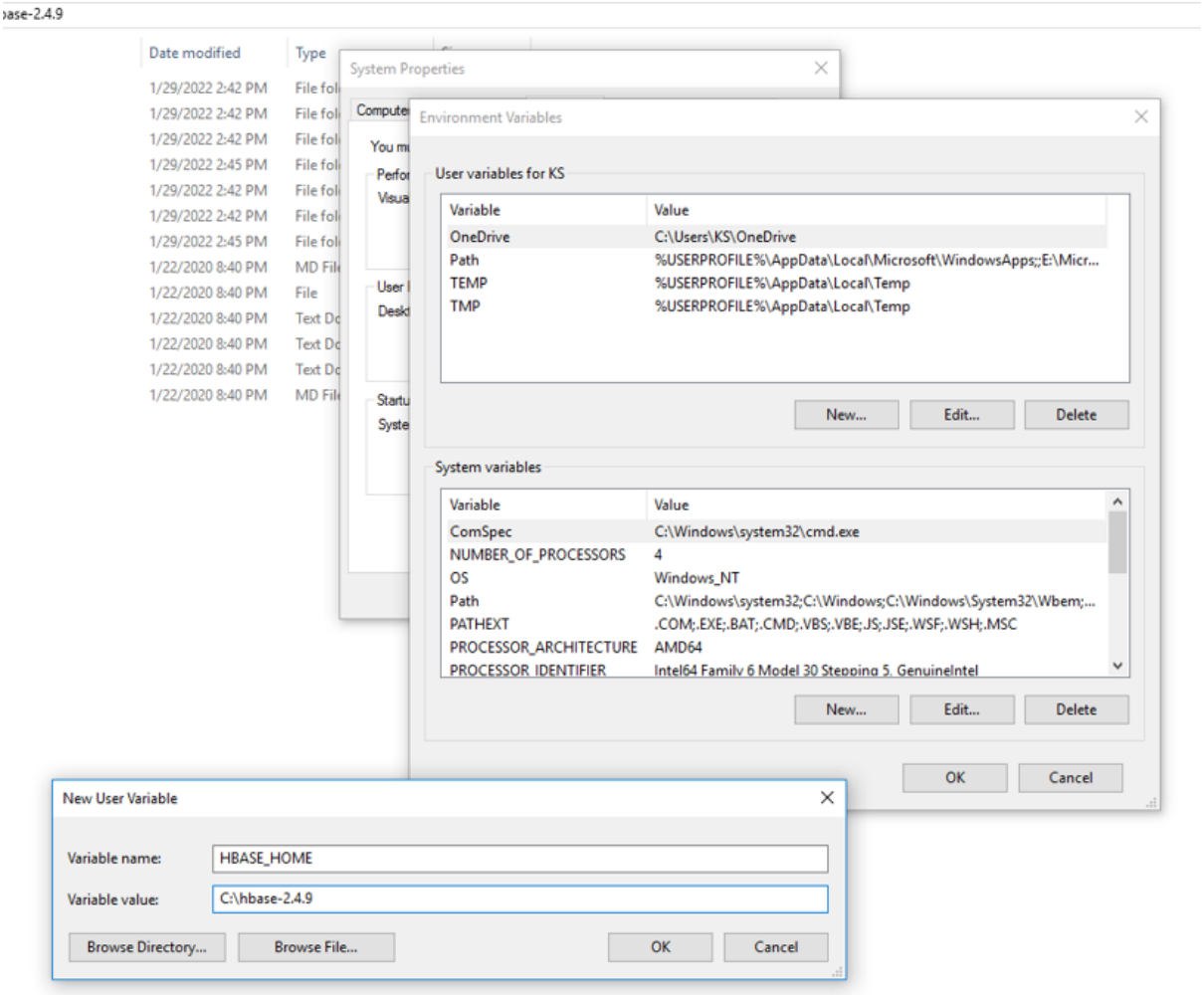
Now set up the environment variables.
Search "System environment variables."



Now click on " Environment Variables."



Then click on "New."



Variable name: HBASE_HOME

Variable Value: Put the path of the Hbase folder.

Step-7 Create a hadoop folder and copy the hadoopwinutil files in hadoop\bin folder- these utility files are needed to run hbase on windows environment

https://drive.google.com/file/d/1DDplK7jAZLRNIcGkTMCrSCUDAyBUwdln/view?usp=sharing

Done with installation

Note : Before running hbase check the JAVA_HOME,HBASE_HOME and java path is set correctly

path -C:\Program Files\Java\jdk1.8.0_111\bin

->Run **command prompt** as **Admin** and goto **hbase\bin** directory

->Command to start hbase

Start-hbase.cmd

CRUD Operation using HBase

hbase shell

HBase provides shell commands to directly interact with the Database and below are a few most used shell commands.

status: This command will display the cluster information and health of the cluster.

```
1    hbase(main):>status
2    hbase(main):>status "detailed"
```

version: This will provide information about the version of HBase.

```
1    hbase(main):> version
```

whoami : This will list the current user.

```
1    hbase(main):> whoami
```

table_help : This will give the reference shell command for HBase.

```
1    hbase(main):009:> table_help
```

Create an HBase table and insert data into the table.

Now that we know, while creating a table user needs to create required Column Families.

Here we have created two-column families for table ‘employee’. First Column Family is ‘Personal Info’ and Second Column Family is ‘Professional Info’.

```
1  create 'employee', 'Personal info', 'Professional Info'
2  0 row(s) in 1.4750 seconds
3
4  => Hbase::Table - employee
```

Upon successful creation of the table, the shell will return 0 rows.

Create a table with Namespace:
A namespace is nothing but a logical grouping of tables.’company_empinfo’ is the namespace id in the below command.

```
1  create 'company_empinfo:employee', 'Personal info', 'Professional Info'
```

Create a table with version:
By default, versioning is not enabled in HBase. So users need to specify while creating. Given below is the syntax for creating an HBase table with versioning enabled.

```
1  create 'tableName',{NAME=>"CF1",VERSIONS=>5},{NAME=>"CF2",VERSIONS=>5}
2  create 'bankdetails',{NAME=>"address",VERSIONS=>5}
```

Put:
Put command is used to insert records into HBase.

```
1  put 'employee', 1, 'Personal info:empId', 10
2  put 'employee', 1, 'Personal info:Name', 'Alex'
3  put 'employee', 1, 'Professional Info:Dept, 'IT'
```

Here in the above example all the rows having Row Key as 1 is considered to be one row in HBase.To add multiple rows

```
1  put 'employee', 2, 'Personal info:empId', 20
2  put 'employee', 2, 'Personal info:Name', 'Bob'
3  put 'employee', 2, 'Professional Info:Dept', 'Sales'
```

As discussed earlier, the user can add any number of columns as part of the row.

Read

'get' and 'scan' command is used to read data from HBase. Lets first discuss 'get' operation.

get: 'get' operation returns a single row from the HBase table. Given below is the syntax for the 'get' method.

```
1    get 'table Name', 'Row Key'
```

```
1    hbase(main):022:get 'employee', 1
```

```
COLUMN                          CELL
 Personal info:Name              timestamp=1504600767520, value=Alex
 Personal info:empId             timestamp=1504600767491, value=10
 Professional Info:Dept          timestamp=1504600767540, value=IT
3 row(s) in 0.0250 seconds
```

Update

To update any record HBase uses the 'put' command. To update any column value, users need to put new values and HBase will automatically update the new record with the latest timestamp.

```
1    put 'employee', 1, 'Personal info:empId', 30
```

The old value will not be deleted from the HBase table. Only the updated record with the latest timestamp will be shown as query output.

To check the old value of any row use below command.

```
1    get 'Table Name', 'Row Key', {COLUMN => 'Column Family', VERSIONS => 3}
```

Delete

'**delete**' command is used to delete individual cells of a record.
The below command is the syntax of delete command in the HBase Shell.

```
1    delete 'Table Name' ,'Row Key','Column Family:Column'
```

```
1    delete 'employee',1, 'Personal info:Name'
```

Practical 4

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.

Administrator: Command Prompt - cqlsh

```
C:\apache-cassandra-3.11.16\bin>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.16 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> create keyspace if not exists college with replication={'class':'SimpleStrategy','replication_factor':1};
cqlsh> use college;
cqlsh:college> create table student(sid int primary key,sname varchar,age int);
cqlsh:college> insert into student(sid,sname,age)values(1,'yash',22);
cqlsh:college> select * from student;

 sid | age | sname
-----+-----+-------
   1 |  22 |  yash

(1 rows)
cqlsh:college> insert into student(sid,sname,age)values(2,'teju',21);
cqlsh:college> insert into student(sid,name,age)values(3,'tanmay',24);
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name name"
cqlsh:college> insert into student(sid,sname,age)values(3,'tanmay',24);
cqlsh:college> insert into student(sid,sname,age)values(4,'popat',25);
cqlsh:college> select * from student;

 sid | age | sname
-----+-----+--------
   1 |  22 |   yash
   2 |  21 |   teju
   4 |  25 |  popat
   3 |  24 | tanmay
```

```
(4 rows)
cqlsh:college> update student set age=19 where sid=1;
cqlsh:college> select * from student;

 sid | age | sname
-----+-----+--------
   1 |  19 |   yash
   2 |  21 |   teju
   4 |  25 |  popat
   3 |  24 | tanmay

(4 rows)
cqlsh:college>
```

```
(4 rows)
cqlsh:college> delete from student where sid=4;
cqlsh:college> select * from student;

 sid | age | sname
-----+-----+--------
   1 |  19 |   yash
   2 |  21 |   teju
   3 |  24 | tanmay

(3 rows)
cqlsh:college>
```

Practical 5

Querying MongoDB and HBase

a) Write and execute MongoDB queries to retrieve specific data from a collection.

- List the products of sports or electronics category
  > db.Product.find({$or:[{category:"Sports"},{category:'Electronic'}]})

  { "_id" : ObjectId("61de99f80e531b700abf0fe2"), "pid" : 1, "descr" : "Mobile", "mfr" : "Samsung", "category" : "Electronic", "uprice" : 30000 }

  { "_id" : ObjectId("61de9a1d0e531b700abf0fe3"), "pid" : 2, "descr" : "TAB", "mfr" : "Samsung", "category" : "Electronic", "uprice" : 50000 }

  { "_id" : ObjectId("61de9a560e531b700abf0fe4"), "pid" : 3, "descr" : "Refrigerator", "mfr" : "LG", "category" : "Electronic", "uprice" : 45000 }

  { "_id" : ObjectId("61de9af90e531b700abf0fe5"), "pid" : 4, "descr" : "Cricket Kit", "mfr" : "Adidas", "category" : "Sports", "uprice" : 5000 }

  { "_id" : ObjectId("61de9af90e531b700abf0fe6"), "pid" : 5, "descr" : "Shoes", "mfr" : "Nike", "category" : "Sports", "uprice" : 450 }

- List the order details for product no 1.
  > db.order.find({pid:1})

  { "_id" : ObjectId("61de9b9f0e531b700abf0fe7"), "ono" : 101, "odate" : "1-jan-22", "cust" : "abc", "pid" : 1, "qty" : 1, "oamt" : 29000 }

  { "_id" : ObjectId("61de9bbb0e531b700abf0fe8"), "ono" : 102, "odate" : "1-jan-22", "cust" : "pqr", "pid" : 1, "qty" : 2, "oamt" : 55000 }

- List the order details like ono, odate & oamt of product no 1.
  > db.order.find({pid:1},{ono:1,odate:1,oamt:1}).pretty()

  {

  "_id" : ObjectId("61de9b9f0e531b700abf0fe7"),

  "ono" : 101,

  "odate" : "1-jan-22",

  "oamt" : 29000

  }

  {

  "_id" : ObjectId("61de9bbb0e531b700abf0fe8"),

  "ono" : 102,

  "odate" : "1-jan-22",

"oamt" : 55000

}

> db.order.find({cust:'xyz'},{ono:1,odate:1,oamt:1}).pretty()

{

"_id" : ObjectId("61de9bf40e531b700abf0fea"),

"ono" : 104,

"odate" : "11-jan-22",

"oamt" : 1750

}

{

"_id" : ObjectId("61de9c180e531b700abf0feb"),

"ono" : 105,

"odate" : "5-jan-22",

"oamt" : 40050

}

- List the product description, manufacturer & unit price in descending order of price.

> db.Product.find({},{descr:1,mfr:1,uprice:1}).sort({uprice:-1})

{ "_id" : ObjectId("61de9a1d0e531b700abf0fe3"), "descr" : "TAB", "mfr" : "Samsung", "uprice" : 50000 }

{ "_id" : ObjectId("61de9a560e531b700abf0fe4"), "descr" : "Refrigerator", "mfr" : "LG", "uprice" : 45000 }

{ "_id" : ObjectId("61de99f80e531b700abf0fe2"), "descr" : "Mobile", "mfr" : "Samsung", "uprice" : 30000 }

{ "_id" : ObjectId("61de9af90e531b700abf0fe5"), "descr" : "Cricket Kit", "mfr" : "Adidas", "uprice" : 5000 }

{ "_id" : ObjectId("61de9af90e531b700abf0fe6"), "descr" : "Shoes", "mfr" : "Nike", "uprice" : 450 }

- List the order no and customer name which amount is less than 10,000.
> db.order.find({oamt:{$lt:10000}},{ono:1,cust:1})

{ "_id" : ObjectId("61de9bd70e531b700abf0fe9"), "ono" : 103, "cust" : "pqr" }

{ "_id" : ObjectId("61de9bf40e531b700abf0fea"), "ono" : 104, "cust" : "xyz" }

- Change the price of product no 4 to 10,000.
> db.Product.update({pid:4},{pid:4,descr:'cricket kit',category:'Sports',uprice:10000,mfr:'adidas'})
WriteResult({ "nMatched" : 1, "nUpserted" : , "nModified" :10 })

- Remove the products document which price is less than 1000.
  > db.Product.remove({uprice:{$lt:1000}})

  WriteResult({ "nRemoved" : 1 })

- List the product's name whose price is highest.
  > db.Product.find({},{pid:1,uprice:1}).sort({uprice:-1}).limit(1)

  { "_id" : ObjectId("61de9a1d0e531b700abf0fe3"), "pid" : 2, "uprice" : 50000 }

- List the second largest orders detail.
  > db.order.find().sort({oamt:-1}).skip(1).limit(1)

  { "_id" : ObjectId("61de9c180e531b700abf0feb"), "ono" : 105, "odate" : "5-jan-22", "cust" : "xyz", "pid" : 3, "qty" : 1, "oamt" : 40050 }

- Remove the only one order placed for product no 10.
  > db.order.remove({pid:1},{justOne:true})

  WriteResult({ "nRemoved" : 1 })

  > db.order.find()

  { "_id" : ObjectId("61de9bd70e531b700abf0fe9"), "ono" : 103, "odate" : "1-jan-22", "cust" : "pqr", "pid" : 5, "qty" : 2, "oamt" : 750 }

  { "_id" : ObjectId("61de9bf40e531b700abf0fea"), "ono" : 104, "odate" : "11-jan-22", "cust" : "xyz", "pid" : 4, "qty" : 1, "oamt" : 1750 }

  { "_id" : ObjectId("61de9c180e531b700abf0feb"), "ono" : 105, "odate" : "5-jan-22", "cust" : "xyz", "pid" : 3, "qty" : 1, "oamt" : 40050 }

  { "_id" : ObjectId("61de9b9f0e531b700abf0fe7"), "ono" : 101, "odate" : "1-jan-22", "cust" : "abc", "pid" : 1, "qty" : 1, "oamt" : 29000 }

- update product id 4 price as 10000

>db.product.update({pid:4},{$set:{uprice:10000}})

b) Perform queries on HBase tables using HBase shell commands.

To retrieve a specific column of row:
Follow the command to read a specific column of a row.

```
1    get 'table Name', 'Row Key',{COLUMN => 'column family:column'}

2    get 'table Name', 'Row Key' {COLUMN => ['c1', 'c2', 'c3']

1    get 'employee', 1 , 'Personal info:empId'
```

```
COLUMN                          CELL
 Personal info:Name              timestamp=1504600767520, value=Alex
 Personal info:empId             timestamp=1504600767491, value=10
 Professional Info:Dept          timestamp=1504600767540, value=IT
3 row(s) in 0.0250 seconds
```

Note: Notice that there is a timestamp attached to each cell. These timestamps will update for the cell whenever the cell value is updated. All the old values will be there but timestamp having the latest value will be displayed as output.

Get all version of a column

Below given command is used to find different versions. Here 'VERSIONS => 3' defines number of version to be retrieved.

```
1    get 'Table Name', 'Row Key', {COLUMN => 'Column Family', VERSIONS => 3}
```

scan:
'scan' command is used to retrieve multiple rows.
Select all:
The below command is an example of a basic search on the entire table.

```
1    scan 'Table Name'

1    hbase(main):074:> scan 'employee'
```

```
ROW                     COLUMN+CELL
 1                      column=Personal info:Name, timestamp=1504600767520, value=Alex
 1                      column=Personal info:empId, timestamp=1504606480934, value=15
 1                      column=Professional Info:Dept, timestamp=1504600767540, value=IT
 2                      column=Personal info:Name, timestamp=1504600767588, value=Bob
 2                      column=Personal info:empId, timestamp=1504600767568, value=20
 2                      column=Professional Info:Dept, timestamp=1504600768266, value=Sales
2 row(s) in 0.0500 seconds
```

Note: All the Rows are arranged by Row Keys along with columns in each row.
Column Selection:
The below command is used to Scan any particular column.

```
1    hbase(main):001:>scan 'employee' ,{COLUMNS => 'Personal info:Name' }
```

```
ROW                                   COLUMN+CELL
 1                                    column=Personal info:Name, timestamp=1504600767520, value=Alex
 2                                    column=Personal info:Name, timestamp=1504600767588, value=Bob
2 row(s) in 0.3660 seconds
```

Limit Query:
The below command is used to Scan any particular column.

```
1    hbase(main):002:>scan 'employee' ,{COLUMNS => 'Personal info:Name',LIMIT =>1 }
```

```
ROW                                   COLUMN+CELL
 1                                    column=Personal info:Name, timestamp=1504600767520, value=Alex
1 row(s) in 0.0270 seconds
```

Update

To update any record HBase uses the 'put' command. To update any column value, users need to put new values and HBase will automatically update the new record with the latest timestamp.

```
1    put 'employee', 1, 'Personal info:empId', 30
```

The old value will not be deleted from the HBase table. Only the updated record with the latest timestamp will be shown as query output.

To check the old value of any row use below command.

```
1    get 'Table Name', 'Row Key', {COLUMN => 'Column Family', VERSIONS => 3}
```

Delete

'**delete**' command is used to delete individual cells of a record.
The below command is the syntax of delete command in the HBase Shell.

```
1    delete 'Table Name' ,'Row Key','Column Family:Column'
```

```
1    delete 'employee',1, 'Personal info:Name'
```

Drop Table:
To drop any table in HBase, first, it is required to disable the table. The query will return an error if the user is trying to delete the table without disabling the table. Disable removes the indexes from memory. The below command is used to disable and drop the table.

```
1    disable 'employee'
```

Once the table is disabled, the user can drop using below syntax.

```
1    drop 'employee'
```

You can verify the table in using ‘exist’ command and enable table which is already disabled, just use ‘enable’ command.

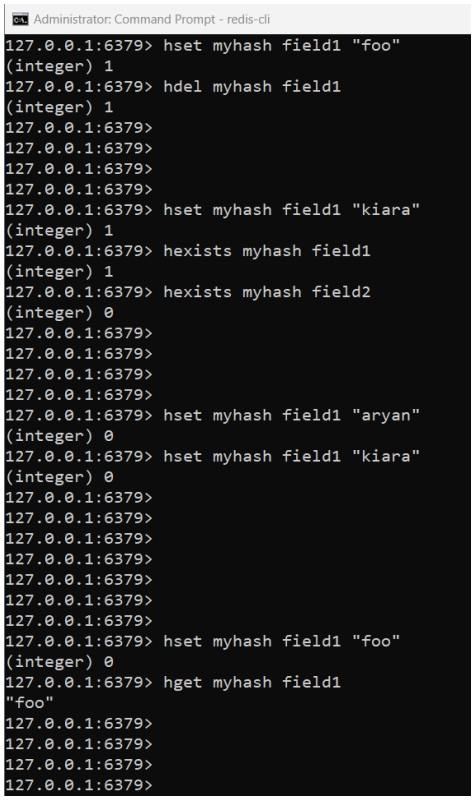
Practical 6

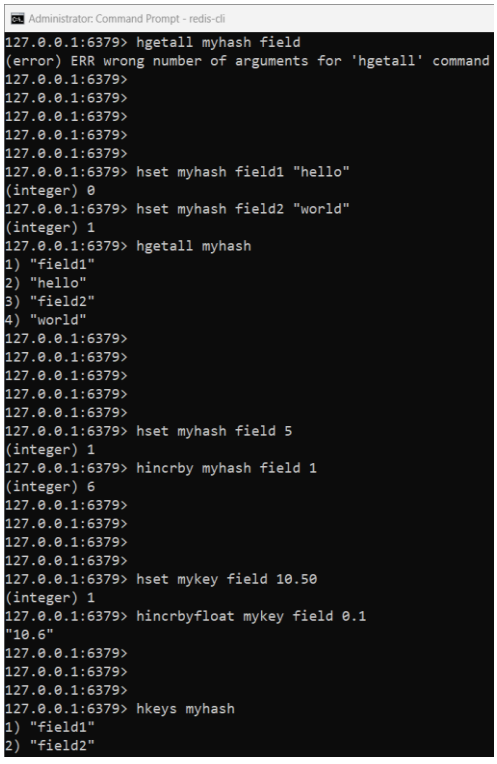
Redis Data Manipulation

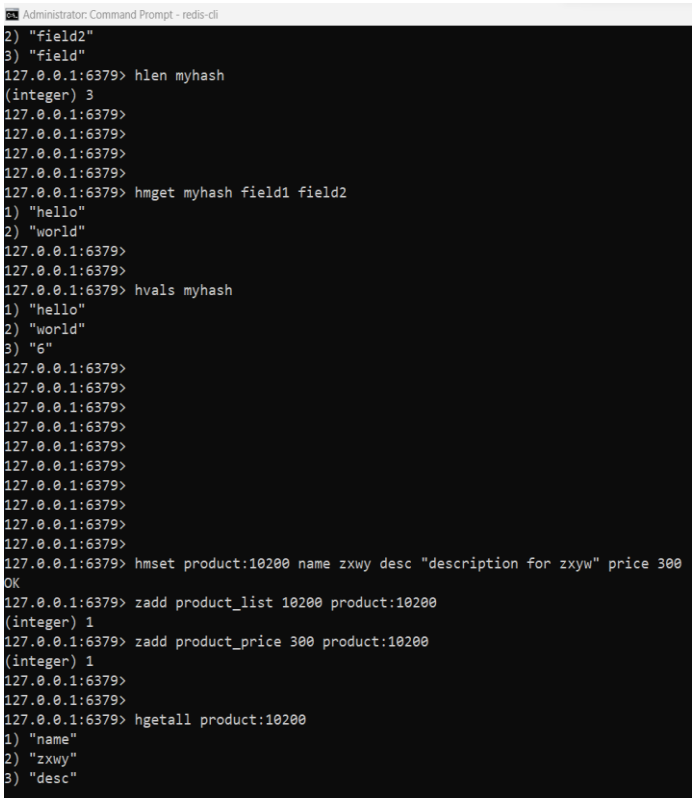
a) Use Redis commands to manipulate and modify data stored in different data structures.

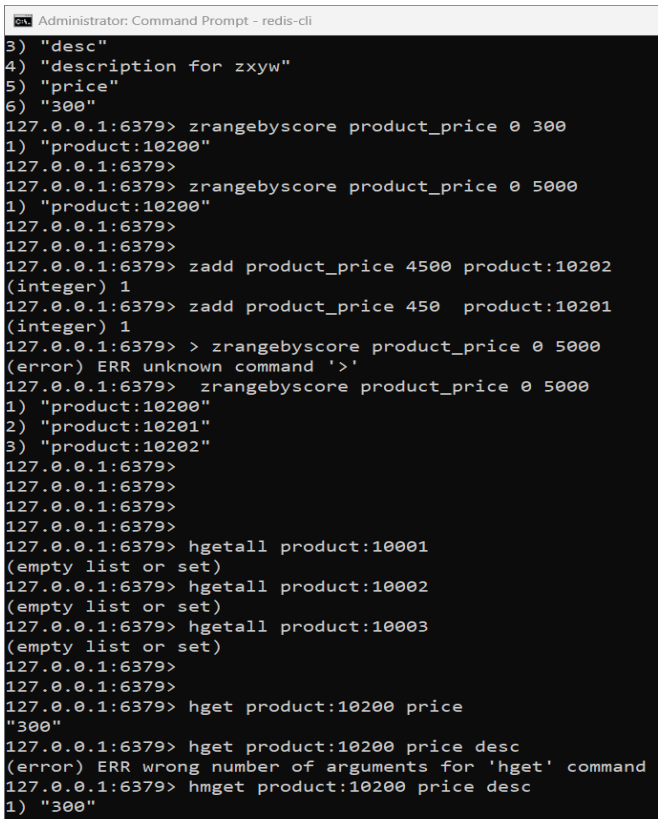
b) Retrieve specific data using Redis query operations.

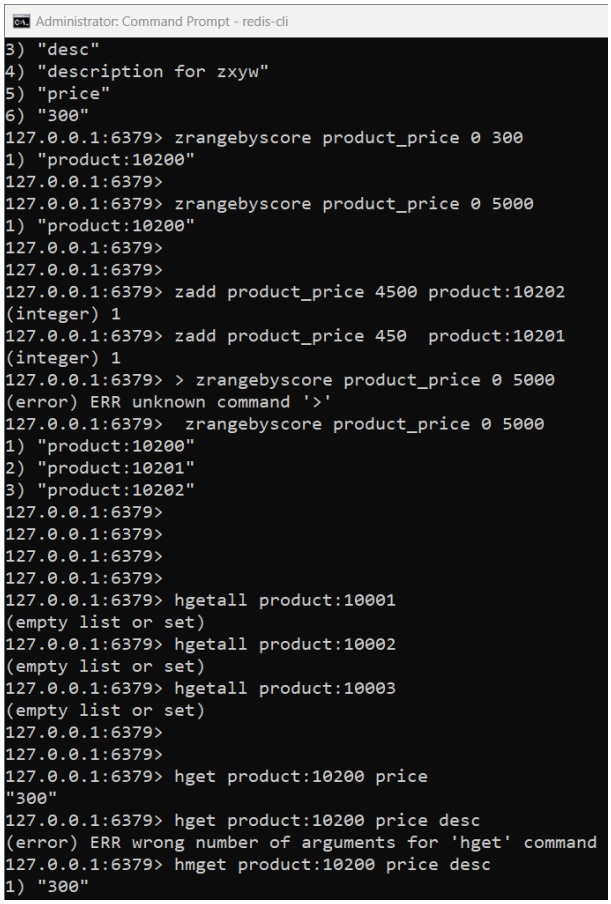
```
C:\Program Files\Redis>redis-cli
127.0.0.1:6379> keys *
 1) "rollno"
 2) "key3"
 3) "myset"
 4) "myset2"
 5) "keyvalue"
 6) "key51"
 7) "ulist"
 8) "list3"
 9) "key2"
10) "qlist"
11) "key21"
12) "key4"
13) "key50"
14) "mylist"
15) "key"
16) "key1"
17) "yash"
18) "ylist"
19) "xlist"
20) "list1"
21) "vinom"
```

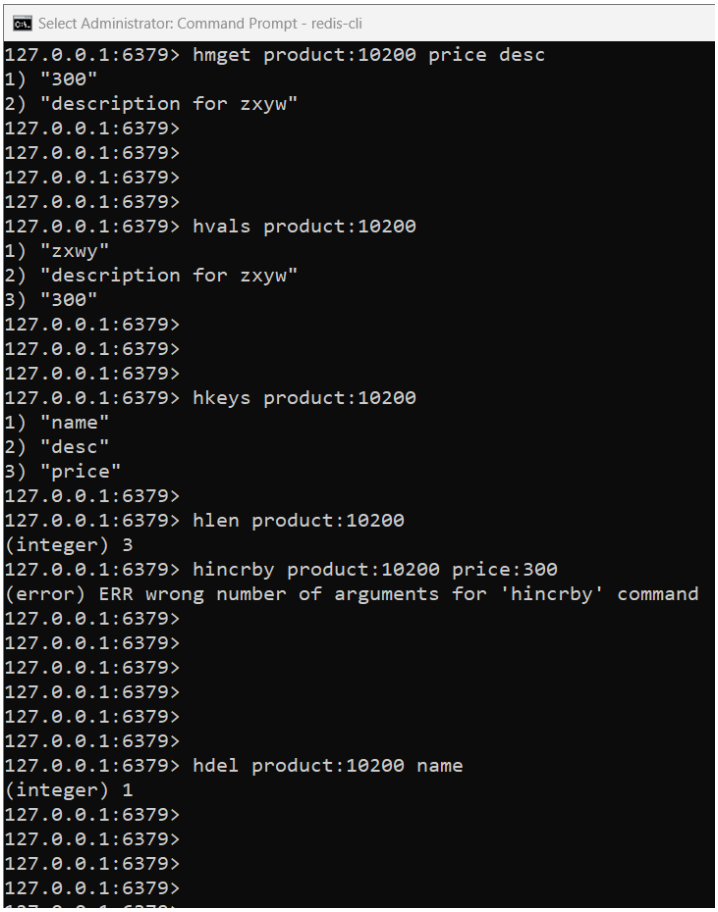

Administrator: Command Prompt - redis-cli
127.0.0.1:6379> hset myhash field1 "foo"
(integer) 1
127.0.0.1:6379> hdel myhash field1
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset myhash field1 "kiara"
(integer) 1
127.0.0.1:6379> hexists myhash field1
(integer) 1
127.0.0.1:6379> hexists myhash field2
(integer) 0
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset myhash field1 "aryan"
(integer) 0
127.0.0.1:6379> hset myhash field1 "kiara"
(integer) 0
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset myhash field1 "foo"
(integer) 0
127.0.0.1:6379> hget myhash field1
"foo"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>


Administrator: Command Prompt - redis-cli
127.0.0.1:6379> hgetall myhash field
(error) ERR wrong number of arguments for 'hgetall' command
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset myhash field1 "hello"
(integer) 0
127.0.0.1:6379> hset myhash field2 "world"
(integer) 1
127.0.0.1:6379> hgetall myhash
1) "field1"
2) "hello"
3) "field2"
4) "world"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset myhash field 5
(integer) 1
127.0.0.1:6379> hincrby myhash field 1
(integer) 6
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hset mykey field 10.50
(integer) 1
127.0.0.1:6379> hincrbyfloat mykey field 0.1
"10.6"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hkeys myhash
1) "field1"
2) "field2"


Administrator: Command Prompt - redis-cli
2) "field2"
3) "field"
127.0.0.1:6379> hlen myhash
(integer) 3
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hmget myhash field1 field2
1) "hello"
2) "world"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hvals myhash
1) "hello"
2) "world"
3) "6"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hmset product:10200 name zxwy desc "description for zxyw" price 300
OK
127.0.0.1:6379> zadd product_list 10200 product:10200
(integer) 1
127.0.0.1:6379> zadd product_price 300 product:10200
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hgetall product:10200
1) "name"
2) "zxwy"
3) "desc"


Administrator: Command Prompt - redis-cli
3) "desc"
4) "description for zxyw"
5) "price"
6) "300"
127.0.0.1:6379> zrangebyscore product_price 0 300
1) "product:10200"
127.0.0.1:6379>
127.0.0.1:6379> zrangebyscore product_price 0 5000
1) "product:10200"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> zadd product_price 4500 product:10202
(integer) 1
127.0.0.1:6379> zadd product_price 450 product:10201
(integer) 1
127.0.0.1:6379> > zrangebyscore product_price 0 5000
(error) ERR unknown command '>'
127.0.0.1:6379> zrangebyscore product_price 0 5000
1) "product:10200"
2) "product:10201"
3) "product:10202"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hgetall product:10001
(empty list or set)
127.0.0.1:6379> hgetall product:10002
(empty list or set)
127.0.0.1:6379> hgetall product:10003
(empty list or set)
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hget product:10200 price
"300"
127.0.0.1:6379> hget product:10200 price desc
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> hmget product:10200 price desc
1) "300"


Administrator: Command Prompt - redis-cli
3) "desc"
4) "description for zxyw"
5) "price"
6) "300"
127.0.0.1:6379> zrangebyscore product_price 0 300
1) "product:10200"
127.0.0.1:6379>
127.0.0.1:6379> zrangebyscore product_price 0 5000
1) "product:10200"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> zadd product_price 4500 product:10202
(integer) 1
127.0.0.1:6379> zadd product_price 450 product:10201
(integer) 1
127.0.0.1:6379> > zrangebyscore product_price 0 5000
(error) ERR unknown command '>'
127.0.0.1:6379> zrangebyscore product_price 0 5000
1) "product:10200"
2) "product:10201"
3) "product:10202"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hgetall product:10001
(empty list or set)
127.0.0.1:6379> hgetall product:10002
(empty list or set)
127.0.0.1:6379> hgetall product:10003
(empty list or set)
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hget product:10200 price
"300"
127.0.0.1:6379> hget product:10200 price desc
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> hmget product:10200 price desc
1) "300"


Select Administrator: Command Prompt - redis-cli
127.0.0.1:6379> hmget product:10200 price desc
1) "300"
2) "description for zxyw"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hvals product:10200
1) "zxwy"
2) "description for zxyw"
3) "300"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hkeys product:10200
1) "name"
2) "desc"
3) "price"
127.0.0.1:6379>
127.0.0.1:6379> hlen product:10200
(integer) 3
127.0.0.1:6379> hincrby product:10200 price:300
(error) ERR wrong number of arguments for 'hincrby' command
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> hdel product:10200 name
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>

Practical 7

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.

- Create an index on specified field in MongoDB collection

```
>
> db.collection.createIndex({orderDate:1})
{
        "createdCollectionAutomatically" : true,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

- Create a index on multiple fields

```
>
> db.student.createIndex({name:1},{unique:true})
{
        "createdCollectionAutomatically" : true,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

- Measure the impact of indexing on query performance

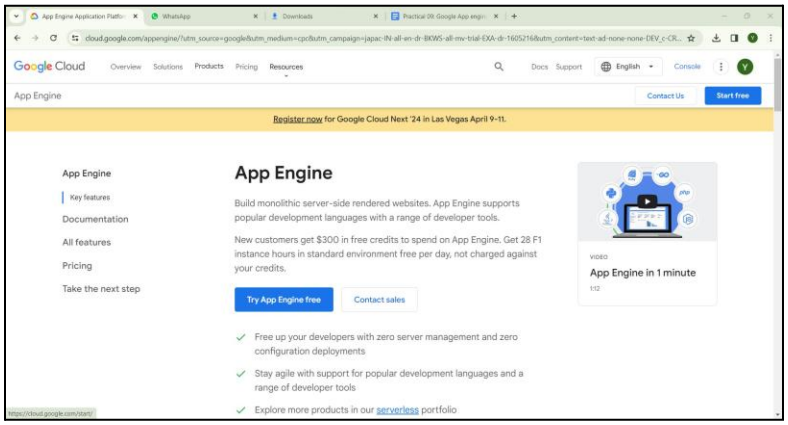
```
> db.product.find().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "alia.product",
                "indexFilterSet" : false,
                "parsedQuery" : {

                },
                "winningPlan" : {
                        "stage" : "EOF"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 0,
                "executionTimeMillis" : 3,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 0,
                "executionStages" : {
                        "stage" : "EOF",
                        "nReturned" : 0,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 1,
                        "advanced" : 0,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "invalidates" : 0
                }
        },
        "serverInfo" : {
                "host" : "Yash_Kamble",
                "port" : 27017,
                "version" : "4.0.28",
                "gitVersion" : "af1a9dc12adcfa83cc19571cb3faba26eeddac92"
        },
        "ok" : 1
}
```

```
> db.product.getIndexes()
[ ]
> db.Product.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "alia.Product"
        }
]
```

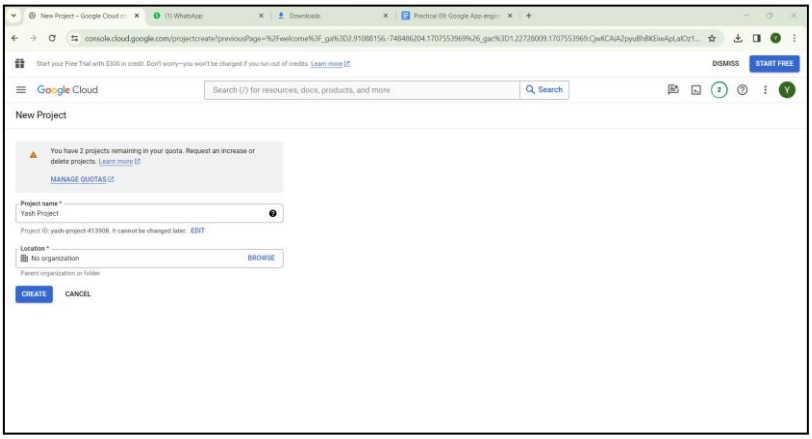
```
> db.product.find({pid:1}).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "alia.product",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "pid" : {
                                "$eq" : 1
                        }
                },
                "winningPlan" : {
                        "stage" : "EOF"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 0,
                "executionTimeMillis" : 4,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 0,
                "executionStages" : {
                        "stage" : "EOF",
                        "nReturned" : 0,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 1,
                        "advanced" : 0,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "invalidates" : 0
                }
        },
        "serverInfo" : {
                "host" : "Yash_Kamble",
                "port" : 27017,
                "version" : "4.0.28",
                "gitVersion" : "af1a9dc12adcfa83cc19571cb3faba26eeddac92"
        },
        "ok" : 1
}
```

Practical 8

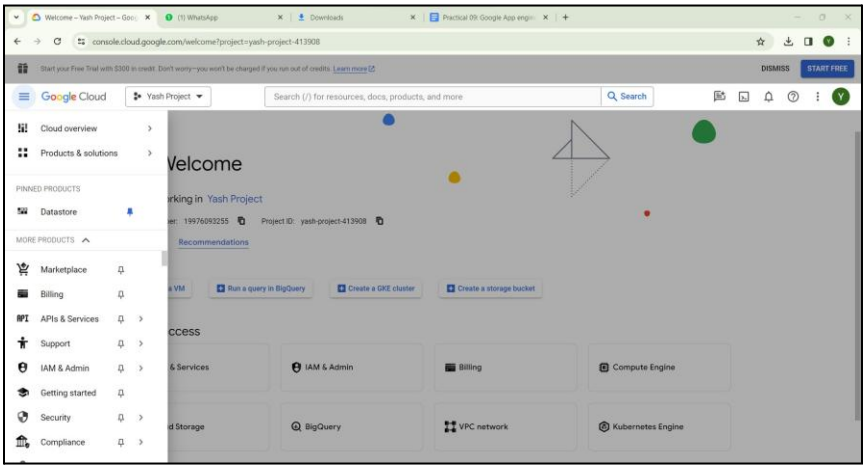
Step 1: Sign in to your google account to use the google app engine.



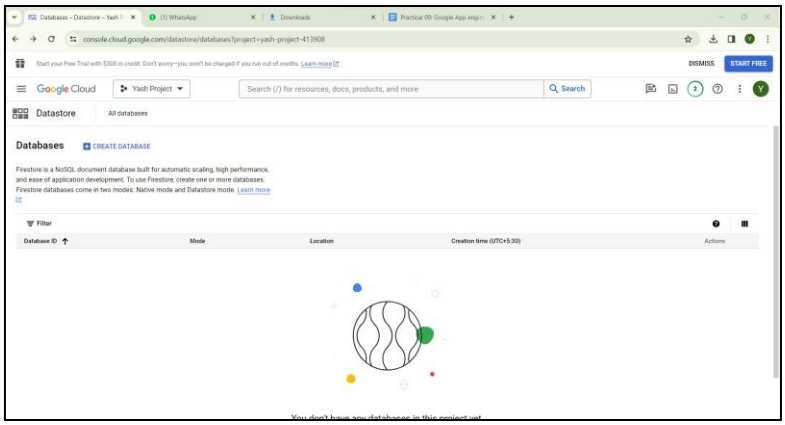
Step 2:Go to the console and click on a new project.



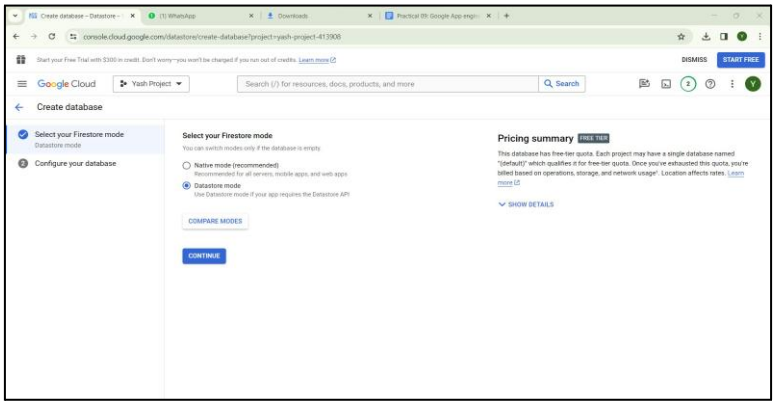
Step 3 : Go to the left option and click on Datastore.

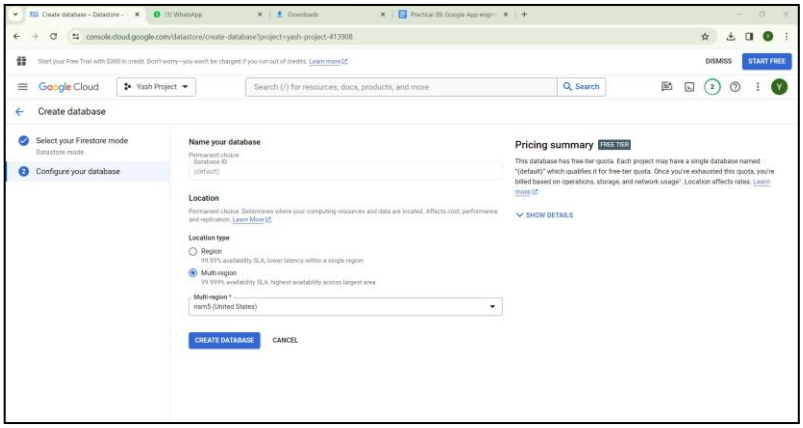


Step 4 : Click on create database.

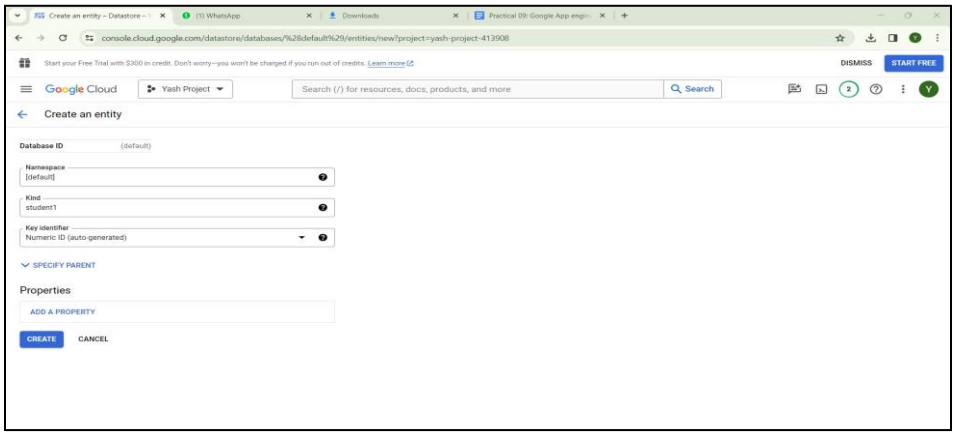


Step 5 : Click on Datastore mode, continue with it and select the multi region option to proceed with create database.

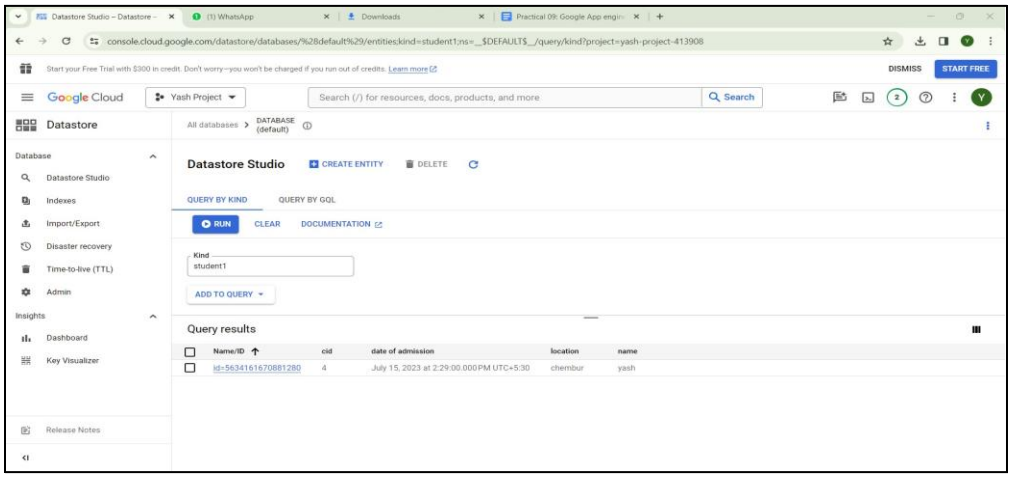




Step 6: Create an entity to make a database.



Step 7 : Add a property i.e datatype,name and value



Step 8: Click on run a query to filter the dbase by where and limit clause.