**Day-1:**

1. **Installation of NoSQL Database - Redis on Windows.**
    a. Redis is an open source data structure store used for pops-up, caching, queueing, streaming engine, storing session variables, message broker, and many more operations.

    b. It provides data structures such as hashes, lists, sets, sorted sets, strings, bitmaps, and streams.

1. Download Redis for 32-bit and 64-bit Windows from the official Redis website www.redis.com, Find Redis-x64-5.0.14.1.msi file.
2. Go to the File location and Double Click on it to install.
3. Accept the "End-User License Agreement" and press the "Next" button:
4. Next, the Destination Folder Window opened.
5. Set the location for Redis installation,
6. Give the path C:\Program Files\Redis. Then, press the "Next" button.
7. Next Screen, It asks for the Port to run Redis. (6379)
8. Additionally, you can also set the "Max Memory limit" and press the "Next" button:
9. Finally, click on the "Install" button to begin the Redis installation:
10. After successfully installing Redis, press the "Finish" button and start using it:

**2. Installation of NoSQL Database - Redis on Linux?**

1.Open to terminal and type sudo apt-get update command
2. Next, Run the Redis server using the apt-get command
3. sudo apt-get install redis-server
4. You can find the location of the Redis installation folder using the redis-cli command.
5. To stop the redis server:
    sudo service redis stop
    sudo systemctl stop redis

To start the redis server:
    sudo service redis start
    sudo systemctl start redis

To restart the redis server:
    sudo service redis restart
    sudo systemctl restart redis

**3.Configuration of Redis Database. (CONFIG, GET).**

127.0.0.1:6379 > CONFIG GET configuration_name
Example: 127.0.0.1:6379> CONFIG GET loglevel

Output: 1) "loglevel"
2) "notice"
127.0.0.1:6379> CONFIG GET *
Output: 1) "dbfilename"
2) "dump.rdb"
3) "requirepass"
4) ""
5) "masterauth"
6) ""
7) "unixsocket"
8) ""
9) "logfile"
10) "Logs/redis_
……..
……..
128) ""
129) "bind"
130) ""

**4. Modifying or Editing the Configurations (CONFIG, SET)**
Syntax:
127.0.0.1:6379> CONFIG SET CONFIGURATION_NAME NEW_CONFIG_VALUE
127.0.0.1:6379> CONFIG SET requirepass secret_password
OK
This will set password temporarily (until redis or server restart)
**5. Redis-CLI Connection Commands (ECHO,PING,AUTH,SELECT,QUIT)**
1. AUTH : It is used to authenticate to the server with the given password.
Syntax: AUTH password

```
127.0.0.1:6379> AUTH secret_password
OK

redis 127.0.0.1:6379> AUTH PASSWORD
(error) ERR Client sent AUTH, but no password is set
redis 127.0.0.1:6379> CONFIG SET requirepass "mypass"
OK
redis 127.0.0.1:6379> AUTH mypass
Ok
```

```
redis 127.0.0.1:6379> GET nonexisting
(nil)
redis 127.0.0.1:6379> SET key "secret"
"OK"
redis 127.0.0.1:6379> GET mykey
"secret"
```

**2. ECHO : It is used to print the given string.**

Syntax: ECHO message

Ex: 127.0.0.1:6379> ECHO "Welcome to redis"

"Welcome to redis"

3. PING : It is used to check whether the server is running or not.

Syntax: PING

Ex: 127.0.0.1:6379> PING

PONG

4. QUIT : It is used to close the current connection.

Syntax: QUIT

5. SELECT : It is used to change the selected database for the current connection

Syntax: SELECT index

These are few Redis-CLI commands:

SET - sets the value of a key.

GET - Gets the value of the key. nil will be returned if the key does not exist.

```
127.0.0.1:6379> SET NAME SSDC
OK
127.0.0.1:6379> GET NAME
"SSDC"
```

**6. Practice Different Data types in Redis Database using Redis-CLI.**

<u>String:</u>

Example : 127.0.0.1:6379> SET name StringDT

OK

127.0.0.1:6379> GET name

"StringsDT"

Hashes:

127.0.0.1:6379> HMSET std:1 name sai percentage 78

OK

127.0.0.1:6379> HGETALL std:1

1) "name"

2) "sai"
3) "percentage"
4) "78"

List:

LPUSH command inserts a new element on the head (first).
RPUSH command inserts a new element on the tail (last).
LRANGE command in order to retrieve a few of recently inserted items.
LPOP command removes an element from the head.
RPOP command removes an element from the tail.

127.0.0.1:6379> LPUSH mylist a # now the list is "a"
127.0.0.1:6379> LPUSH mylist b # now the list is "b","a"
127.0.0.1:6379> RPUSH mylist c # now the list is "b","a","c"
LRANGE mylist
1) “b”
2) “a”
3) “c”

Set:

- SADD – Add one or more items to a set.
- SMEMBERS – Retrieves all items from a set.
- SREM – Removes an existing item from a set.
- SPOP - Removes an existing item randomly from a set.
- SCARD – Returns number of item in the set
- SDIFF - Deference between set1 and set2
- SDIFFSTORE – Finds The deference between set1 and set2 and Stores into new set.
- SUNION - Combines unique items of the two sets
- SUNIONSTORE - Combines unique items of the two sets and Stores into new set.
- SINTER
- SINTERSTORE

Example:
127.0.0.1:6379> sadd set1 10
(integer) 1
127.0.0.1:6379> sadd set1 20 25 30
(integer) 3
127.0.0.1:6379> smembers set1
1) “10”
2) “20”

3) “25”
4) “30”
127.0.0.1:6379> srem set1 25
(integer) 1
127.0.0.1:6379> smembers

**7. Strings with Example, Lists with Example**
**8.Sets, Hashes, Stored Sets with Example**
**<u>Day 2:</u>**

1. **Creating a Database using Redis**
   In Redis-cli we can use pre- defined databses. There are 16 databses in redis-cli. These are names as 1 to 15.
   127.0.0.11:6379> CONFIG GET databases
   Output: 1) "databases"
   2) "16"
   To use the database of redis use the command SELECT.
   Sntax: SELECT database_name
   Ex: 127.0.0.1:6379[1]> SELECT 13
   OK
   127.0.0.1:6379[13]>

2. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using** Redis-CLI for Student Database.
   Create:
   127.0.0.1:6379> HMSET std:1 Name Sai DOB 12-5-2000 Course 3-DS Address Hyd
   Mobile 9848012345
   OK
   127.0.0.1:6379> HMSET std:2 Name Raju DOB 12-6-2001 Course 3-DS Address
   Kompally Mobie 8813445905
   127.0.0.1:6379> HMSET std:3 Name Suresh DOB 12-6-2001 Course 3-BCOM
   Address Medchal Mobie 7395820923
   Read: HGETALL, HGET, HMGET
   a) Display all the fields of Student 1.
   127.0.0.1:6379> HGETALL std:1

```
1) "Name"
2) "Sai"
3) "DOB"
4) "12-5-2000"
5) "Course"
6) "3-DS"
7) "Address"
8) "Hyd"
9) "Mobile"
10) "9848012345"
```

b) Display name of the Student 3

```
127.0.0.1:6379> HGET std:3 Name
"Suresh"
```

c) Display Name, Course ,Address of student 2

```
127.0.0.1:6379> HMGET std:2 Name Course Address
1) "Raju"
2) "3-DS"
3) "Kompally"
```

Update: HSET

a) Change the address of Student 1 as Hyderabad

```
127.0.0.1:6379> HSET  std:1  Address  Hyderabad
(integer) 0
127.0.0.1:6379> HGET std:1 Address
"Hyderabad"
```

b) Update the mobile number of student 2 as 7493452234

```
127.0.0.1:6379> HSET  std:2  Mobile  7493452234
(integer) 0
127.0.0.1:6379> HGET std:1  Mobile
7493452234
```

Delete:

a) Delete the Address of student 3

```
127.0.0.1:6379> HDEL std:3 Address
(integer) 1
```

3. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using Redis-CLI for Employee Database.**

Write same as Above for Employee Details

Create:

Write same as Above for Product Details

(Name, Qualification, Experience, Department, Mobile, Salary)

Read:

a) Display details of employee 2
b) Display name, department, Salary of employee 1
c) Display name , experience of employee

Update:

a) update the salary of employee 2 by incrementing 5000.
b) Change the department of employee 3

Delete:

a) Delete the employee 2

4. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using Redis-CLI for Product Database.**

Create:

Write same as Above for Product Details

(Name, Company, Quantity, Price, Mfd-Date,Exp-Date)

Read:

d) Display Product Details of product 5
e) Disply name, price, exp-date of product 3

Update:

c) update the price of product 1
d) update the exp-date of product 3

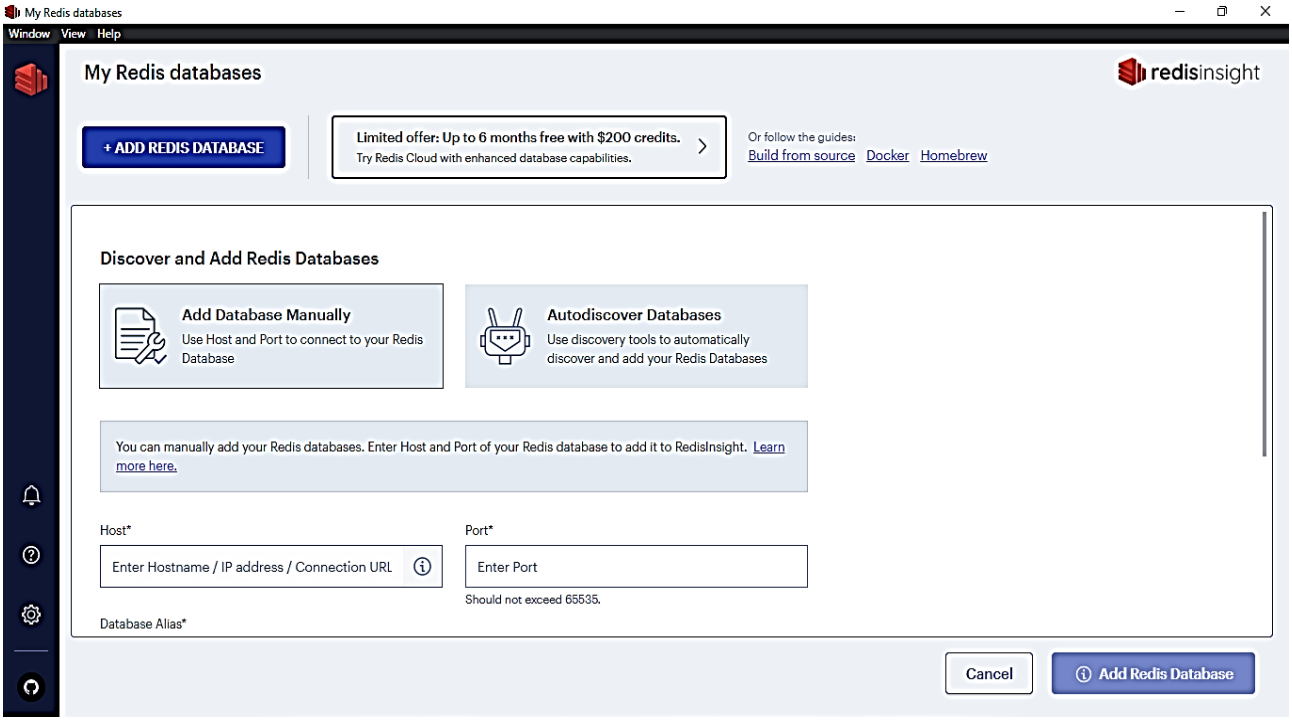
Delete:

a) Delete the product 3

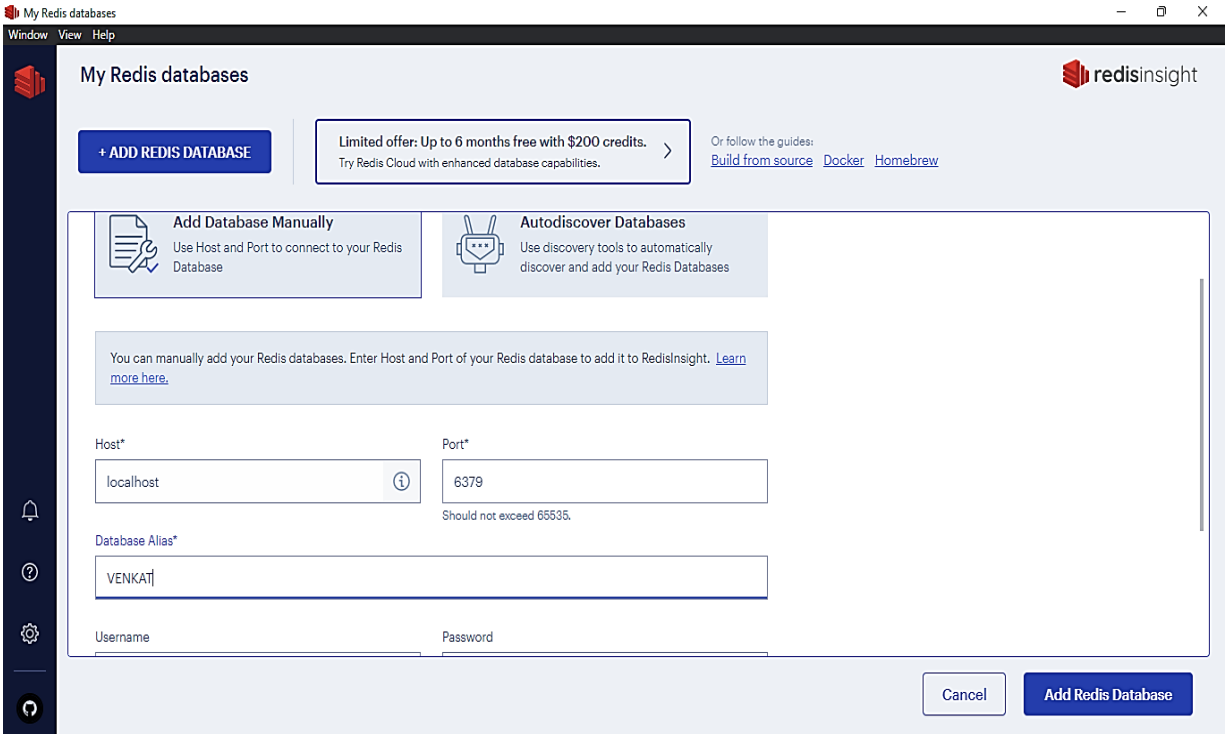
5. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using GUI Interface RedisInsight for Student Database.**

The RedisInsight graphic user interface helps you visually browse and interact with Redis data. Browse, filter, and visualize Redis keys, perform CRUD operations, or delete keys in bulk.

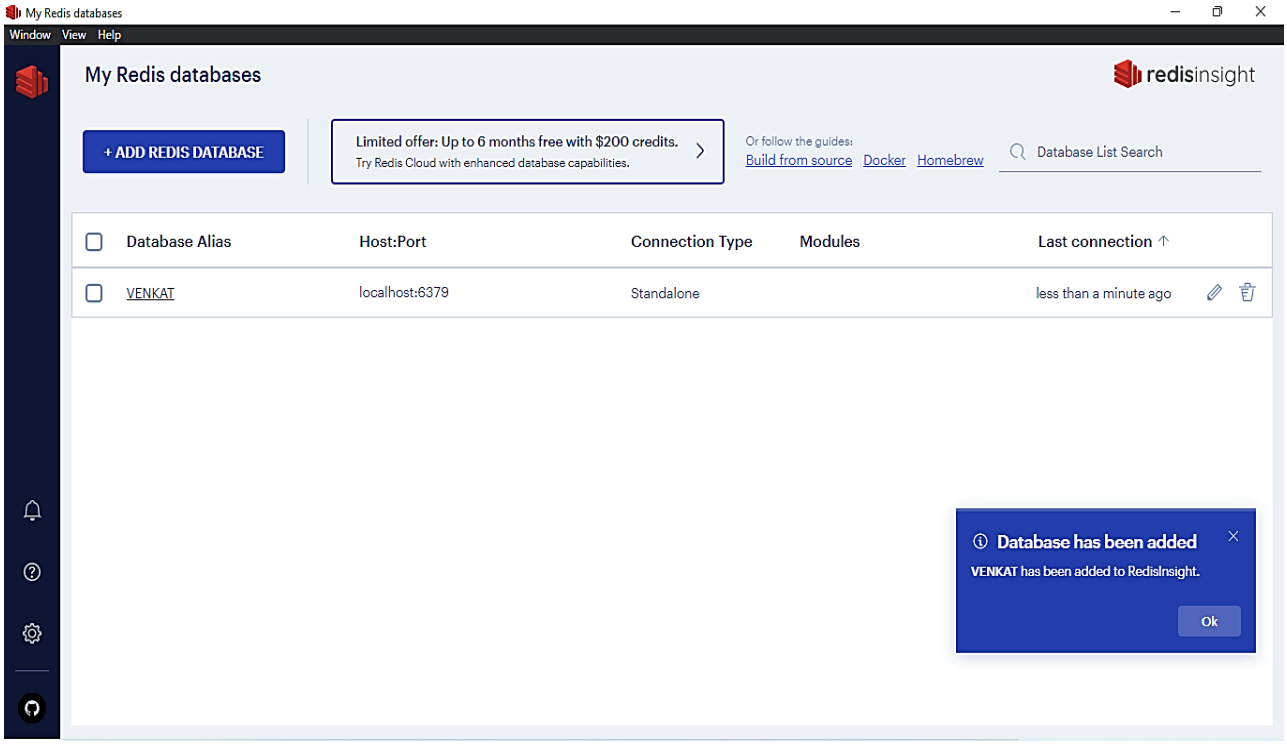
1. Redis Insight window. Click on ADD REDIS DATABASE Button.



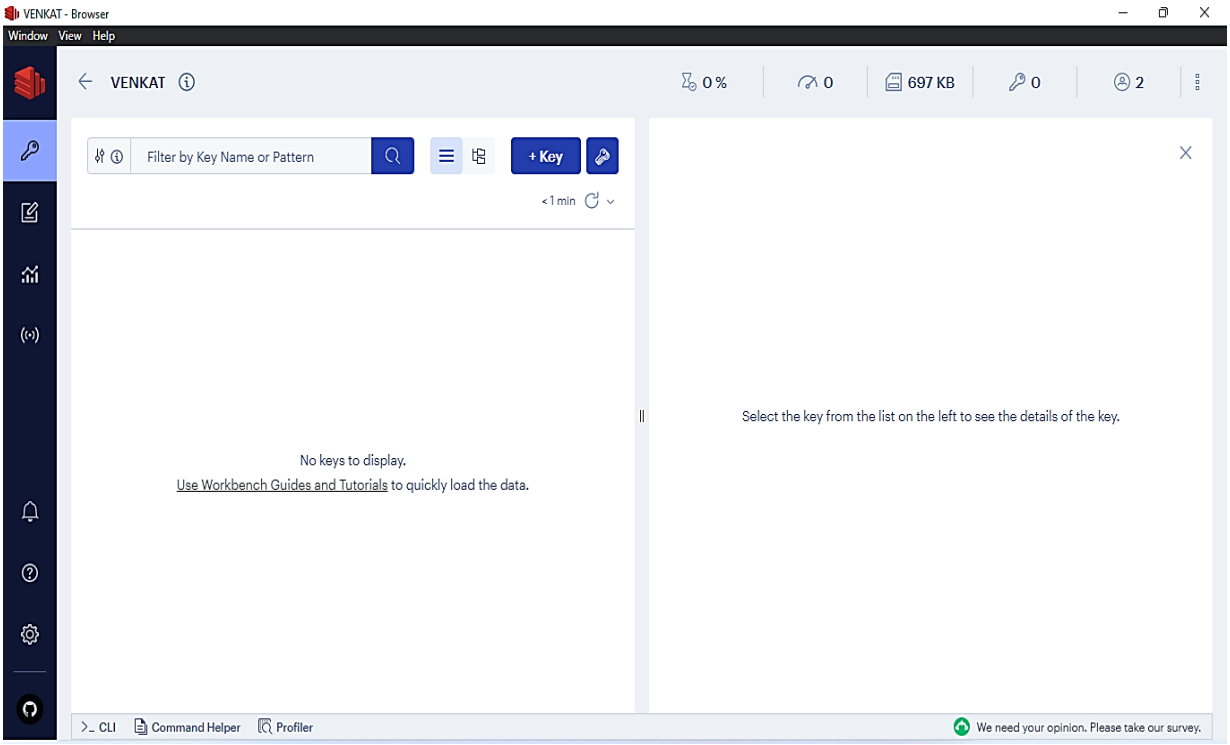

2. Select Add Database Manually option, then enter the details of host- localhost, port:6379 and Database Alias: New Database Name. Then click on Add Redis Database.



3. Redis Database created with specified Alias Name.




4. Click on Database Name to perform CRUD operations on RADIS database.

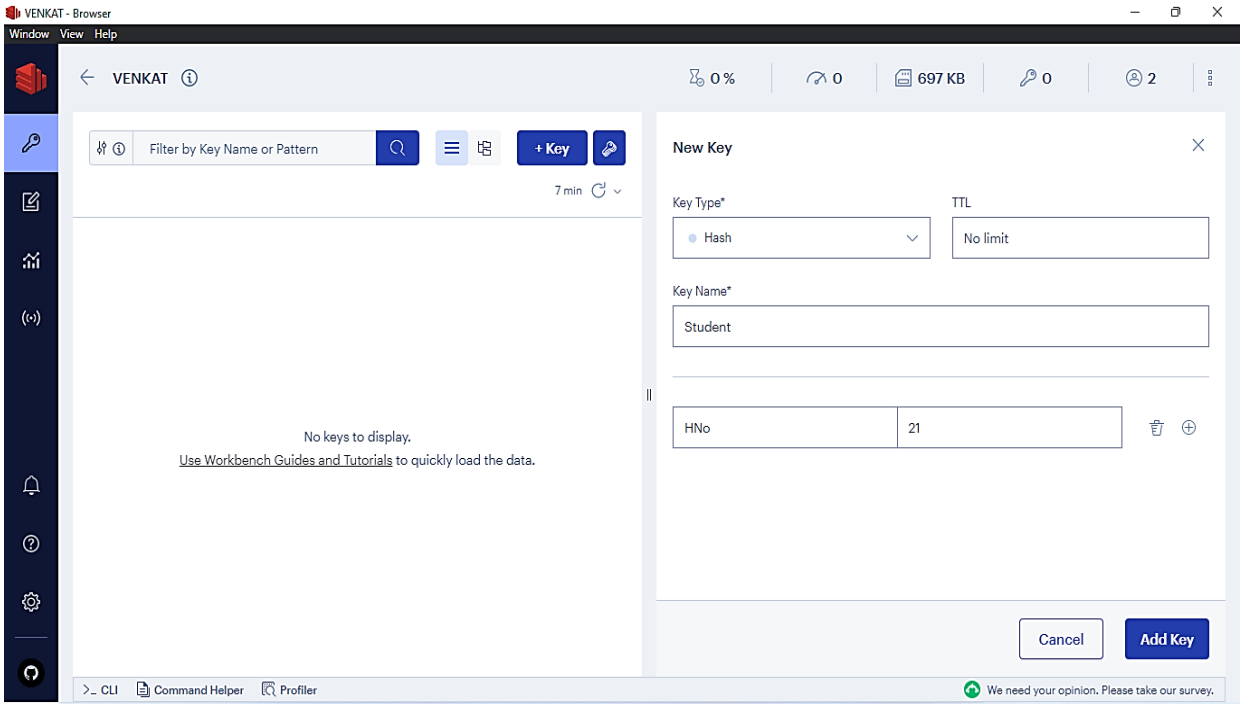


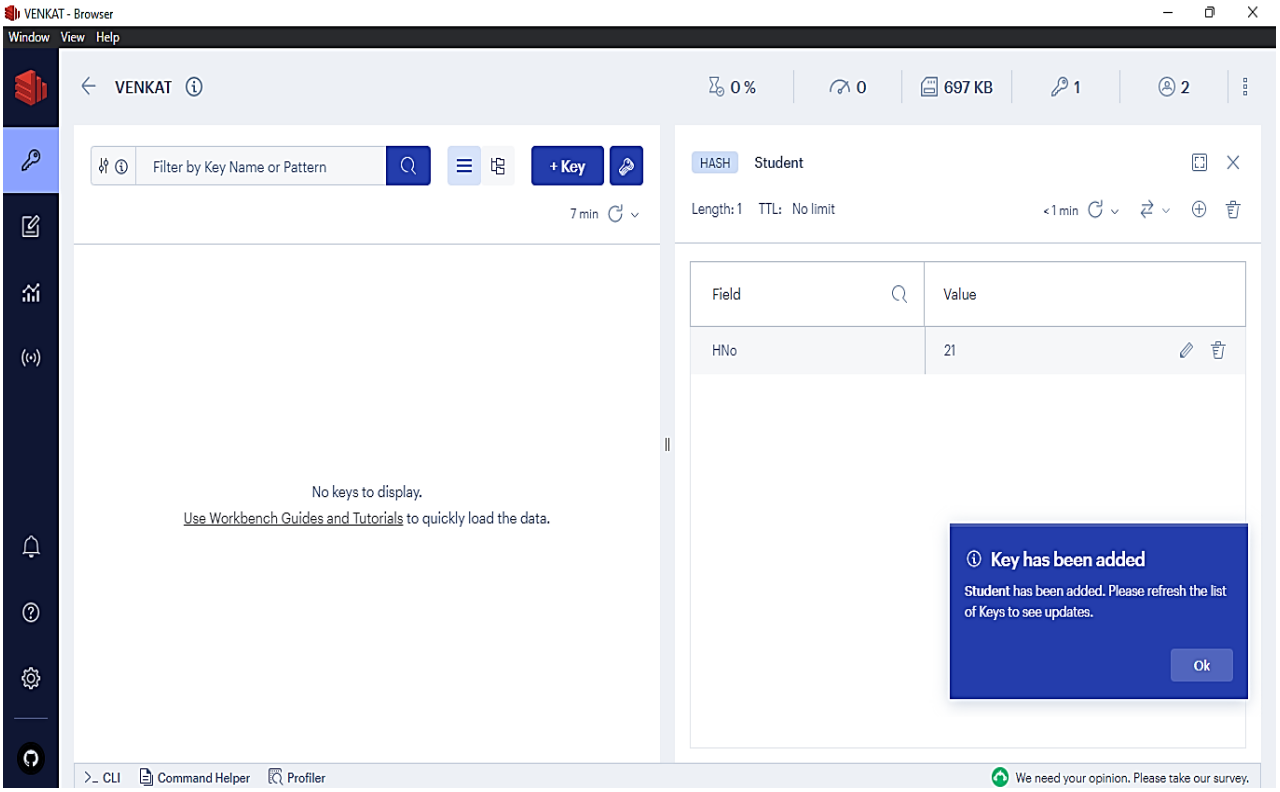
6. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using GUI Interface RedisInsight for Employee Database**.

   Create:

   1. Click on “ +Key ” Button. New key window opened in the same window.
   2. Select which type of key going to create (Set, String, Hash, List, Sortedlist) from the option “ Key Type ”
   3. Then Type “ Key Name ” and value for the Key.
   4. Then click on “ Add Key ”

      The key is successfully added to database.



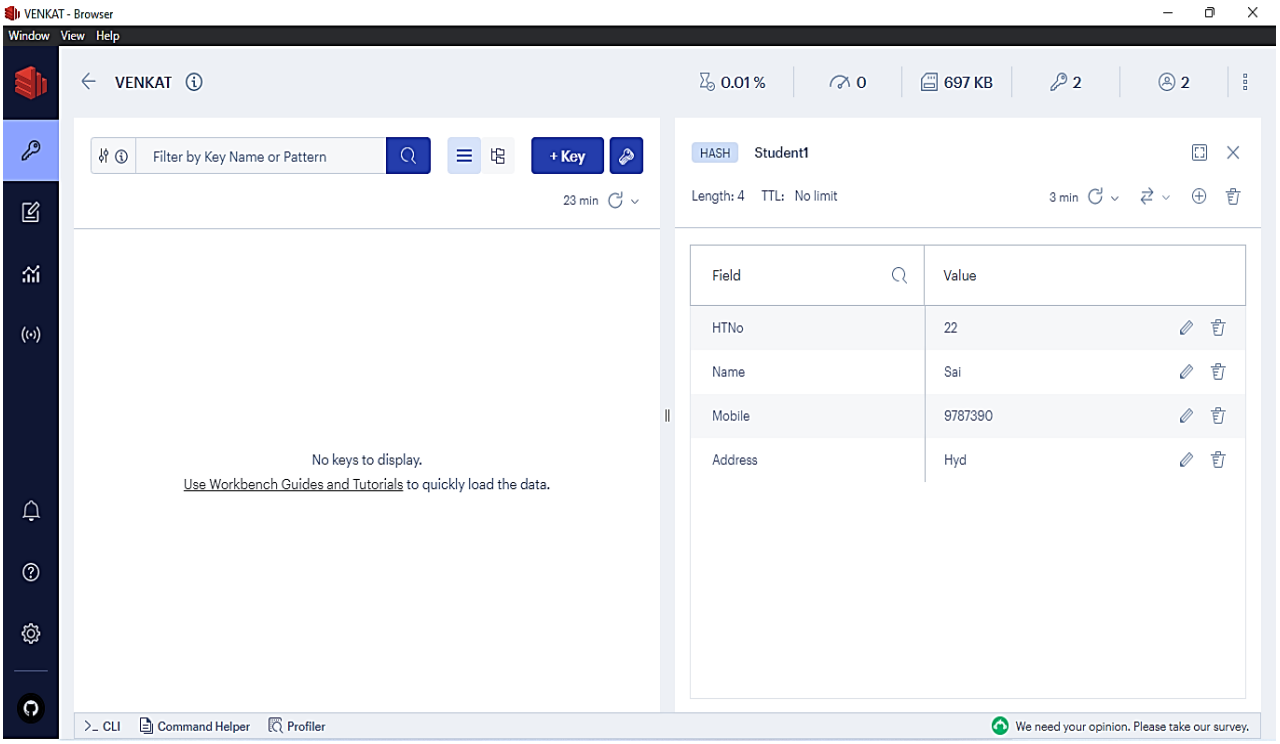



Read: To read the data click on “ Key Name ” then the values are displayed right side of the same window.

a) Read the details of Student 1

Step 1: Click on the Student1 Key which is listed in Keys.

Right side window the student 1 details will be displayed.




Update:

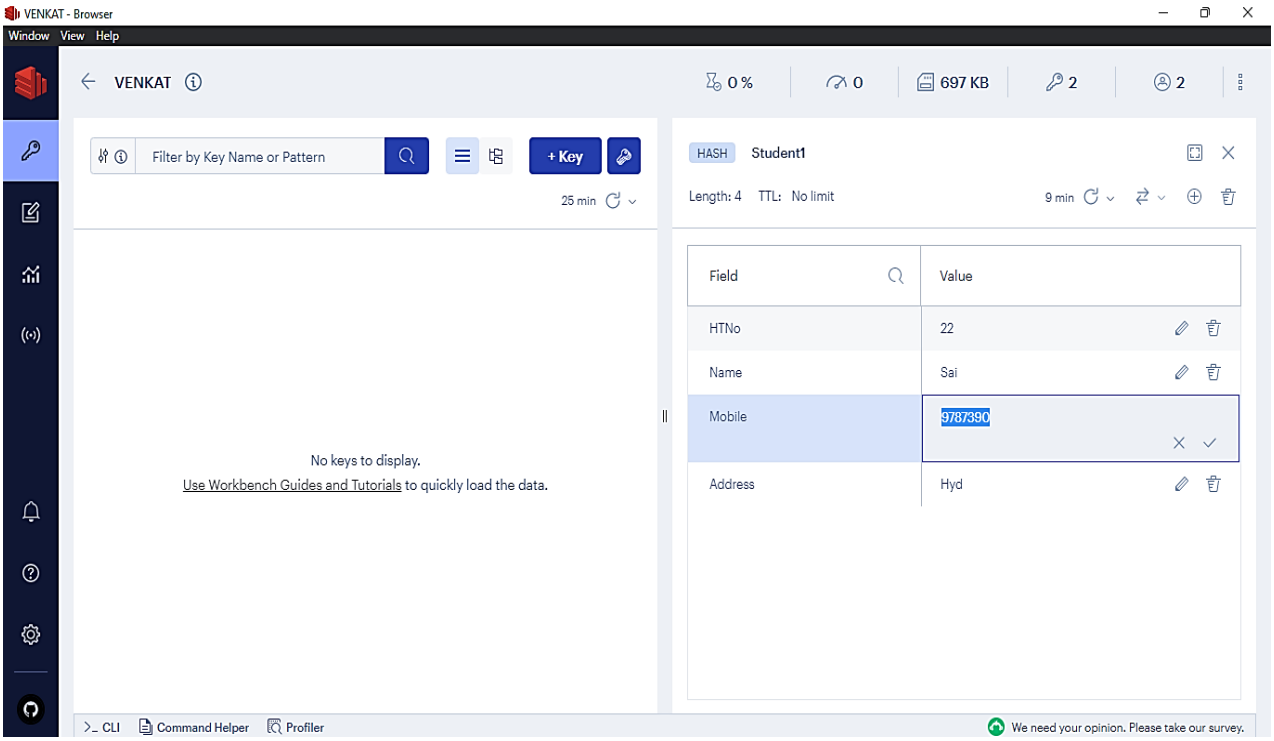
1. click on " Key Name " then the values are displayed right side of the same window.

2. Which field / key value want to change simply click  edit icon of that field value and edit.

a) Change the student1 mobile number.

Step 1: Select the Student1 Key, which is to be update. The student details are displayed right of the window.

Step 2: Click on  edit icon of the value field of mobile field. Delete and Retype new mobile number.




Delete: We can delete the total Key or some fields of Key using delete icon.

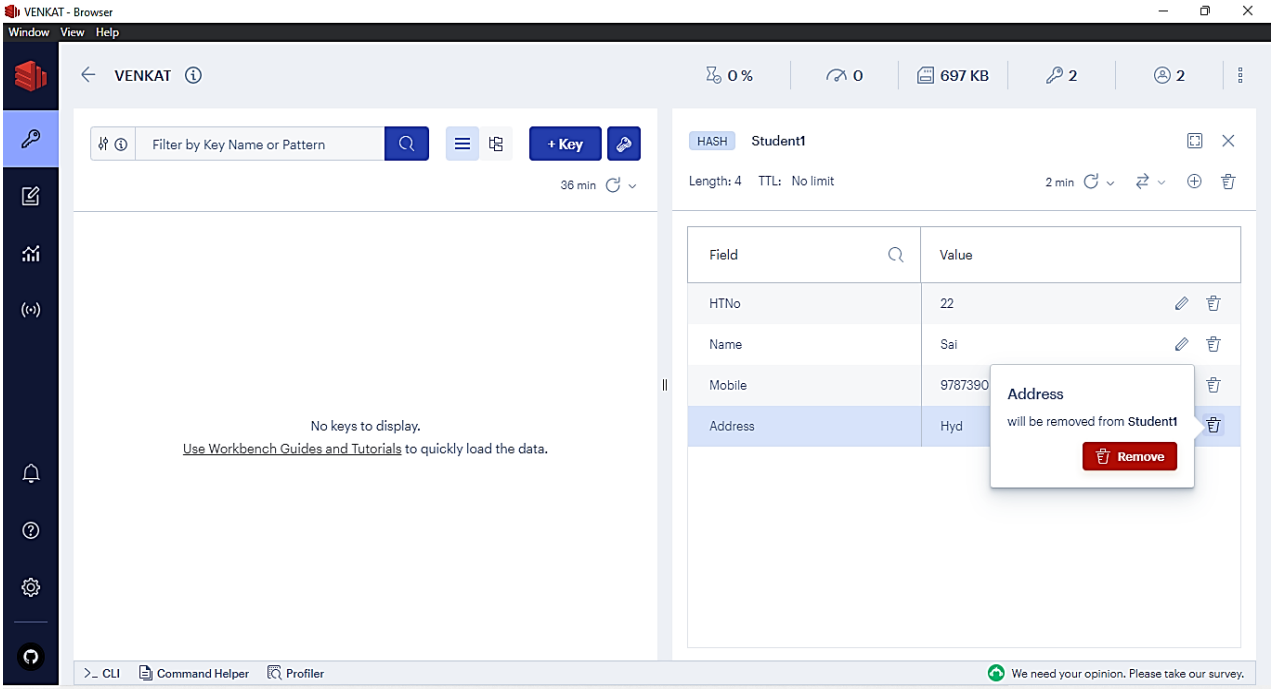
Q: Delete the Address field of student 1.

Ans: 1. Select the Student1 key from Keys list

2. Click on delete icon of the Address field. Popup window opened.

3. Then click on Remove button in popup window.

4. The address field will be deleted.



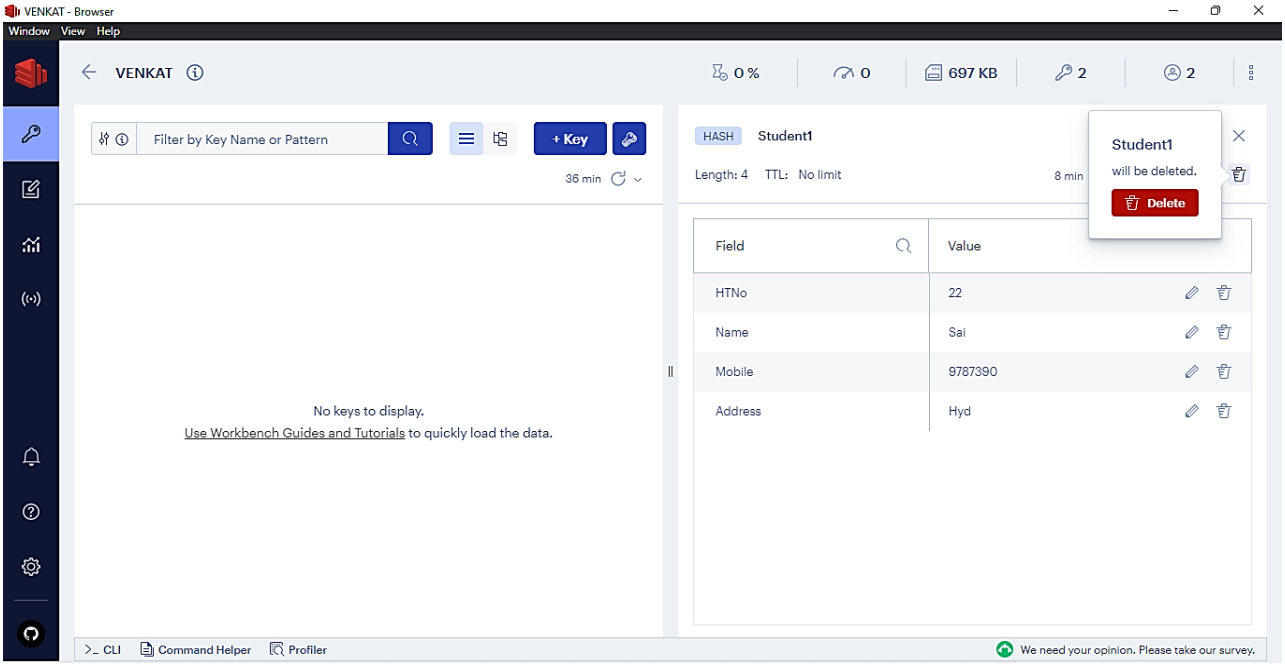
Q: Delete the record of student1

Ans: 1. Select the Student1 key from Keys list

2. Click on delete icon available in Key properties. Popup window opened.

3. Then click on Delete button in popup window.

4. The student1 will be deleted.




7. **CRUD (Create, Read, Update, and Delete) operations on the Redis Database using GUI Interface RedisInsight for Product Database.**

Ans: Write the above steps for Product Table

**<u>Day 3:</u>**

8. **Connecting Python Application program with Redis Database.**

Step 1: Run a Redis server

Step 2: Install the Redis client library using pip command

```
pip install redis
```

Step 3: import redis module

```
import redis
```

Step 4: Establish connection to localhost

```
Redis_obj = redis.Redis( host= 'localhost', port= '6379')
```

Step 5: Write the code using differ data type keys.

Ex:

```
redis_obj.set ('mykey', 'Hello from Python!')
value = redis.get('mykey')
```

```
print(value)
```

**9. Practice Different Data types in Redis Database using python programming.**

Set: Ex1:
```
import redis
R = redis.Redis( host= 'localhost', port= '6379')
R .set('Sno', 21)
R.get('Sno')
```

Ex 2:
```
import redis
R = redis.Redis( host= 'localhost', port= '6379')
fr = ["avocado", "strawberry", "strawberry", "mango", "orange"]
R.sadd('fruits', *fr)
R.smembers('fruits')
```
Output: {'avocado', 'mango', 'orange', 'strawberry'}

List:
```
import redis
R = redis.Redis( host= 'localhost', port= '6379')
plang = ['python', 'C#', 'C++', 'C++', 'javascript']
R.lpush ('languages', *plang)
r.lrange('languages', 0, -1)
```

Q: Perform CRUD (Create, Read, Update, and Delete) operations on the Redis Database using Python for Student Database.

```
import radis
import json
R = redis.Redis( host= 'localhost', port= '6379')
Student1 = { 'sno':01, 'name': 'Rahul', 'Marks': [15,18,14,19,16] }
Student2 = { 'sno':02, 'name': 'Suresh', 'Marks': [16,12,15,79,14] }
R.set('Student1', json.dumps(Student1))
R.set('Student2', json.dumps(Student2))
res = json.loads(R.get('Student1'))
print(res)
res = json.loads(R.get('Student2'))
print(res)
R.delete('Student2')
print(R.get('Student2))
```

Q: Perform CRUD (Create, Read, Update, and Delete) operations on the Redis Database using Python for Employee Database.

Same as above Student
Q: Perform CRUD (Create, Read, Update, and Delete) operations on the Redis Database using Python for Product Database.
Same as above Student

**Day 4:**

**Download the MongoDB MSI Installer Package**

To download the latest version of MongoDB, visit the mongoDB official website www.mongodb.com.
In the Products Menu, under the Community Edition, click on the Community server.
The latest Version , Platform, package information displayed.
Click on Download button. The mongoDB.msi file will be downloaded in to system
Install the mongoDB software using the downloaded file in the downloads directory. Aollow the steps mentioned there and install the software.

1. Click Next to start installation.
2. Accept the licence agreement then click Next.
3. Select the Complete setup.
4. Select "Run service as Network Service user" and make a note of the data directory.
5. Click Install to begin installation.
6. Hit Finish to complete installation.

**Ubuntu Mongodb installation:**

Ctrl+T
sudo systemctl start mongod
sudo systemctl daemon-reload
sudo systemctl status mongod
sudo systemctl stop mongod
sudo systemctl restart mongod
mongosh

**3.connect MongoDB**

Use Database name(Mongoshell)
Ex:use ssdc

**4. Creating a Database or Table (Collection) using MONGODB**

db.createCollection("student")
db.student.insertOne({ sno: 21, name: "Abhi", address: "Hyd", marks: [ 15,18,16,20 ] })

db.student.insertMany([{sno: 30, name: "Rohan", address: “Kompally”, marks: [ 17,14,13,18 ] },{sno: 23, name: "Sai", address: “Hyd”, marks: [ 19,20,15,12 ] }])

**5.Create a Database using MONGODB. Common Commands in MongoDB** (USE,SHOW)

Use ssdc;

Show dbs;

**Drop a Database using MONGODB**

Drop Database ssdc;

7.**CRUD (Create, Read, Update, and Delete) operations on the MongoDB Database**

db.createCollection(“employees”)

db.employees.insert( {empId: 3, name: 'Ava', dept: 'Sales' });

db.collection.insertOne(): Inserts one document

db.employees.insertOne( {empId: 4, name: 'Nick', dept: 'Accounting' });

db.collection.insertMany: Inserts multiple documents

db.employees.insertMany([
{empId: 1, name: 'Clark', dept: 'Sales' },
{empId: 2, name: 'Dave', dept: 'Accounting' } ]);

Retrive:

db. employees.find()
db.employees.updateOne(
{empId: 2 },
{ $set: { region: "Asia" } }
);
db.collection.updateMany() : Updates multiple documents in collection based on the condition.
db.employees.updateMany(
{ dept: 'Sales'},

{ $set: { region: "US" } }
);

Deleting documents

db.collection.deleteOne(<filter>, <options>): Deletes a Single document from collection
db.employees.deleteOne({ empId: 1})
db.collection.deleteMany(<filter>, <options>): Deletes all documents with matching filter
db.employees.deleteMany({ dept: 'Sales'})
db.employees.remove({empId : 2})

**Day 5:**

1.Create a table or collection Student. And insert 10 rows (Documents)
db.createCollection("student")

db.student.insertOne({ sno: 21, name: "Abhi", address: "Hyd", marks: [ 15,18,16,20 ] })
db.student.insertMany([{sno: 30, name: "Rohan", address: "Kompally", marks: [ 17,14,13,18 ] },{sno: 23, name: "Sai", address: "Hyd", marks: [ 19,20,15,12 ] }])

lly(10 Records)

2. **List all documents in the Student**

List all documents with formatted output

db.Student.find()
db.student.find({name: "Abhi"})

**3. Create a table or collection Employee. And insert 10 rows**

db.createCollection("employees")

db.employees.insert( {empId: 3, name: 'Ava', dept: 'Sales' ,age:35});

db.collection.insertOne(): Inserts one document

db.employees.insertOne( {empId: 4, name: 'Nick', dept: 'Accounting',age:45});

db.collection.insertMany: Inserts multiple documents

```
db.employees.insertMany([
 {empId: 1, name: 'Clark', dept: 'Sales',age:70},
 {empId: 2, name: 'Dave', dept: 'Accounting',age:50} ]);
```

Lly (10 records)

4.i **List all documents in the Student.**

db.Student.find()

ii. List all documents with formatted output.

db.Student.find().pretty()

iii. Delete a record

db.student.remove({sno:21})

5. **Practice Different Data types in MongoDB Database**

- String − This is the most commonly used datatype to store the data. String in
- MongoDB must be UTF-8 valid.
- Integer − This type is used to store a numerical value. Integer can be 32 bit or 64 bit
- depending upon your server.
- Boolean − This type is used to store a boolean (true/ false) value.
- Double − This type is used to store floating point values.
- Min/ Max keys − This type is used to compare a value against the lowest and highest
- BSON elements.
- Arrays − This type is used to store arrays or list or multiple values into one key.
- Timestamp − ctimestamp. This can be handy for recording when a document has been
- modified or added.
- Object − This datatype is used for embedded documents.
- Null − This type is used to store a Null value.
- Date − This datatype is used to store the current date or time in UNIX time format. You
- can specify your own date time by creating object of Date and passing day, month,

- Object ID − This datatype is used to store the document's ID.
- Binary data − This datatype is used to store binary data.
- Code − This datatype is used to store JavaScript code into the document.
- Regular expression − This datatype is used to store regular expression.

db.employee.insertOne({

"intern_name": "Rajesh",

"intern_skills": "Software Development",

"intern_salary": 7500,

"intern_score": 87.75,

"intern_status": true,

"skills": ["Software Development", "C++", "Java"],

"employee_dob": ISODate("2004-04-10T12:45:42.389Z"),

})

6. Usage of Where Clause equivalent in MongoDB

Greater Than Equals {<key>:{$gte:<value>}} where field >= value

db.student.find({name: "Abhi"})

db.student.find({'Address":'hyd'})

db.student.find({"per":{$gt:70}})

7. AND and OR operations in MongoDB

db.employees.find({ $and: [{"dept": "sales"}, {"emp_age": {$gte:20,$lte: 30}}]}).

db.employees.find({ $or: [{"dept": "Sales"},{"dept": "Accounting"}]})

**Day-6:**

1.Limit , Sort the records in MongoDB?

```
db.studdent.find().limit(5)
db.student.find().limit(5).skip(5)
db.student.find().sort({sno: -1})
db.student.find().sort({
 sno: -1,
name: -1
})
```

2. Indexing and Advanced Indexing

```
db.student.getIndexes()
[
{
“v” : 1,
“key” : {
“_id” : 1
},
“ns” : “mkyong.users”,
“name” : “_id_”
}
]
```

3.Create a table or collection Student. And insert 10 rows

```
db.createCollection(“student”)

db.student.insertOne({ sno: 21, name: "Abhi", address: “Hyd”, marks: [ 15,18,16,20 ] })
db.student.insertMany([{sno: 30, name: "Rohan", address: “Kompally”, marks: [ 17,14,13,18 ] },{sno: 23, name: "Sai", address: “Hyd”, marks: [ 19,20,15,12 ] }])
```

4.i.List all documents in the Student.

```
db.student.find({})
```

List all documents with formatted output.

```
db.student.find().pretty()
```

5.List the document of a student whose name is “Kiran”

db.student.find({name: " Kiran "})

List the students whose percentage is greater than 70

db.student.find({“per”:{$gt:70}})

5. List the students whose course is ‘data science’ and year is ‘3 year’

db.student.find([{“course”:”datascience”},{“year:3}])

Arrange the records by name in ascending order

db.student.find.sort({name:1})

6. Arrange the records by percentage in descending order.

db.student.find({“per”:-1})

List first 4 documents in the Student

db. student.find().limit(4);

7. List 3rd and 4th documents in the Student.

db. student.find().skip(2).limit(2);

Delete a document(s) or student(s), whose percentage < 40

db. student.deleteOne( {per: {$lt:40}} )