Practical 8

AIM: NoSQL Database

- **Perform CRUD operation on Mongo DB**
- **Import CSV file in Mongo DB**
- **Perform CRUD operation on Couch DB**
- **Perform CRUD operation on Redis DB**

Performing CRUD in MongoDB

Step 1: To Create Database in MongoDB.
Type command **use RDNC**.

```
   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

test> use RDNC
switched to db RDNC
RDNC>
```

Step 2: To display the name of current database:

```
RDNC> db
RDNC
RDNC>
```

Step 3: To list down all the databases, use the command **show dbs**. This command lists down all the databases and their size on the disk.

```
RDNC> show dbs
admin   40.00 KiB
config  72.00 KiB
local   72.00 KiB
mera    40.00 KiB
mera2    8.00 KiB
```

Since our database has no collections inside it. Our database is not been listed.

Step 4: To create a collection in the database.
Type command **db.createCollection("Student")**

```
mera2    8.00 KiB
RDNC> db.createCollection("Student")
{ ok: 1 }
RDNC>
```

List all the databases again. Now, you will see that our database is also been shown here. Before it was not there.

```
RDNC> show dbs
RDNC      8.00 KiB
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
mera     40.00 KiB
mera2     8.00 KiB
RDNC>
```

Step 5: Insert data to the collection Student

```
RDNC> db.Student.insertOne({"rollno":22014,"name":"Sadiq","email":"sadiq@gmail.com"})
{
  acknowledged: true,
  insertedId: ObjectId("6371cc7b8c408e9788cbbbc8")
}
RDNC>
```

Step 6: Find the data in Student

```
RDNC> db.Student.find()
[
  {
    _id: ObjectId("6371cc7b8c408e9788cbbbc8"),
    rollno: 22014,
    name: 'Sadiq',
    email: 'sadiq@gmail.com'
  }
]
RDNC>
```

Another way of printing the data of collection.

```
RDNC> db.Student.find().pretty()
[
  {
    _id: ObjectId("6371cc7b8c408e9788cbbbc8"),
    rollno: 22014,
    name: 'Sadiq',
    email: 'sadiq@gmail.com'
  }
]
RDNC>
```

Step 7: Insert data in the collection with insertMany() method

```
RDNC> db.Student.insertMany([{"rollno":2,"name":"Pratik"},{"rollno":3,"name":"Kunal"},{"rollno":4,"name":"Hriday"},{"rollno":5,"name":"Avinash"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6371cdfc8c408e9788cbbbc9"),
    '1': ObjectId("6371cdfc8c408e9788cbbbca"),
    '2': ObjectId("6371cdfc8c408e9788cbbbcb"),
    '3': ObjectId("6371cdfc8c408e9788cbbbcc")
  }
}
RDNC>
```

Display the inserted data in the collection.

```
}
RDNC> db.Student.find().pretty()
[
  {
    _id: ObjectId("6371cc7b8c408e9788cbbbc8"),
    rollno: 22014,
    name: 'Sadiq',
    email: 'sadiq@gmail.com'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbc9"),
    rollno: 2,
    name: 'Pratik'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbca"),
    rollno: 3,
    name: 'Kunal'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcb"),
    rollno: 4,
    name: 'Hriday'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcc"),
    rollno: 5,
    name: 'Avinash'
  }
]
RDNC>
```

Step 8: Update the collection students.

```
RDNC> db.Student.updateOne({rollno:22014},{$set:{"name":"Sadiq Sonalkar"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
RDNC>
```

Print the data in the collection to check if the data has been updated or not.

```
RDNC> db.Student.find().pretty()
[
  {
    _id: ObjectId("6371cc7b8c408e9788cbbbc8"),
    rollno: 22014,
    name: 'Sadiq Sonalkar',
    email: 'sadiq@gmail.com'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbc9"),
    rollno: 2,
    name: 'Pratik'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbca"),
    rollno: 3,
    name: 'Kunal'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcb"),
    rollno: 4,
    name: 'Hriday'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcc"),
    rollno: 5,
    name: 'Avinash'
  }
]
RDNC>
```

Step 9: You can also add the extra data in the existing document with the same updateOne() method.

```
RDNC> db.Student.updateOne({rollno:2},{$set:{"mobile":"9988776655"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
RDNC>
```

Check if the data has been added or not.

```
RDNC> db.Student.find().pretty()
[
  {
    _id: ObjectId("6371cc7b8c408e9788cbbbc8"),
    rollno: 22014,
    name: 'Sadiq Sonalkar',
    email: 'sadiq@gmail.com'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbc9"),
    rollno: 2,
    name: 'Pratik',
    mobile: '9988776655'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbca"),
    rollno: 3,
    name: 'Kunal'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcb"),
    rollno: 4,
    name: 'Hriday'
  },
  {
    _id: ObjectId("6371cdfc8c408e9788cbbbcc"),
    rollno: 5,
    name: 'Avinash'
  }
]
RDNC>
```

Step 10: To drop the collection in the database. Then type show collections to confirm if our Student collection is dropped or not.

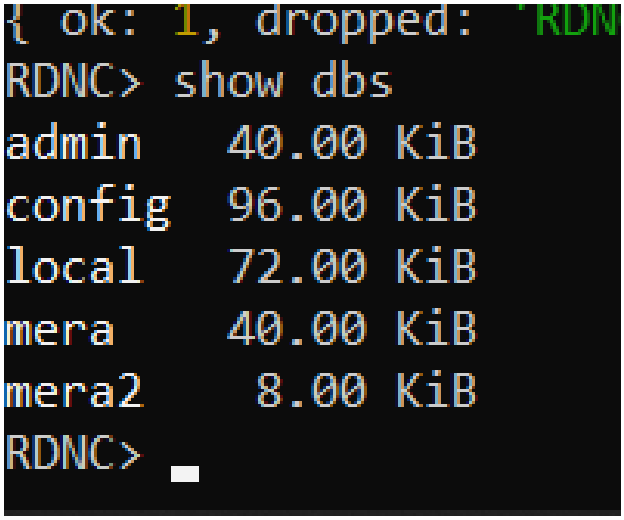
```
RDNC> db.Student.drop()
true
RDNC> show collections

RDNC>
```

So, it’s dropped.

Step 11: Drop the current database.

```
RDNC> db.dropDatabase()
{ ok: 1, dropped: 'RDNC' }
RDNC>
```

Check if database has been dropped successfully.

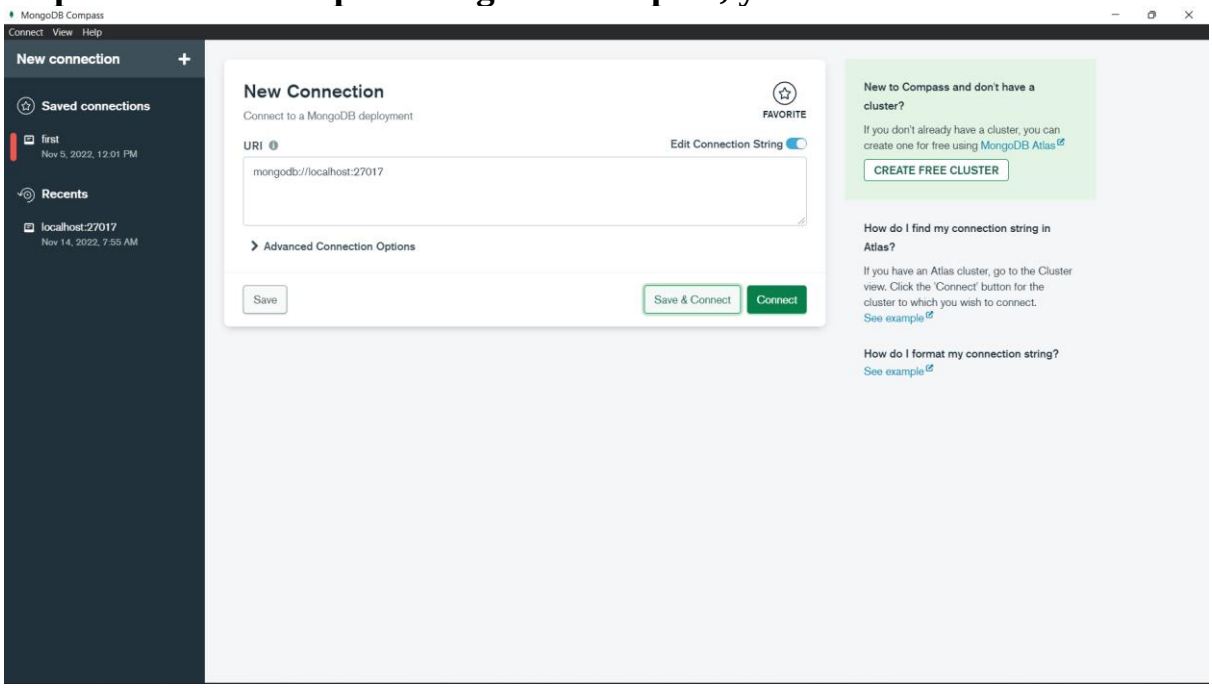


Our database is completely dropped.

Import CSV file in Mongo DB

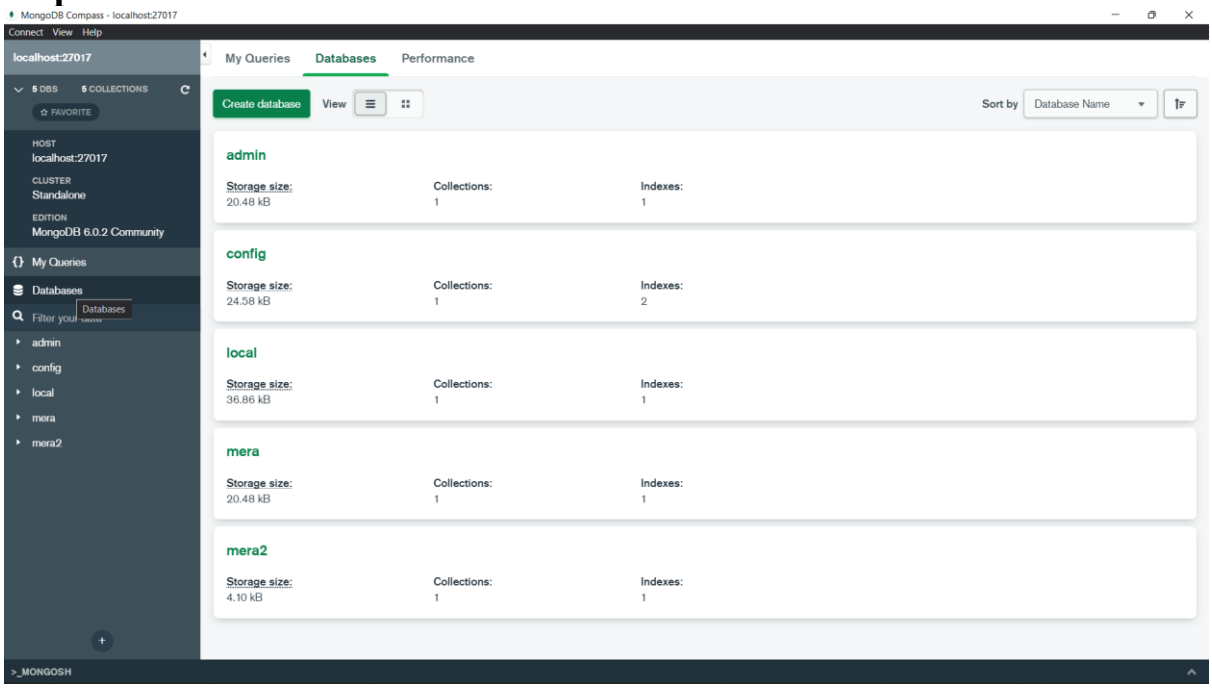
For importing csv file we should have MongoDB Compass.

Step 1: Install and Open MongoDB Compass, you will land on this screen.



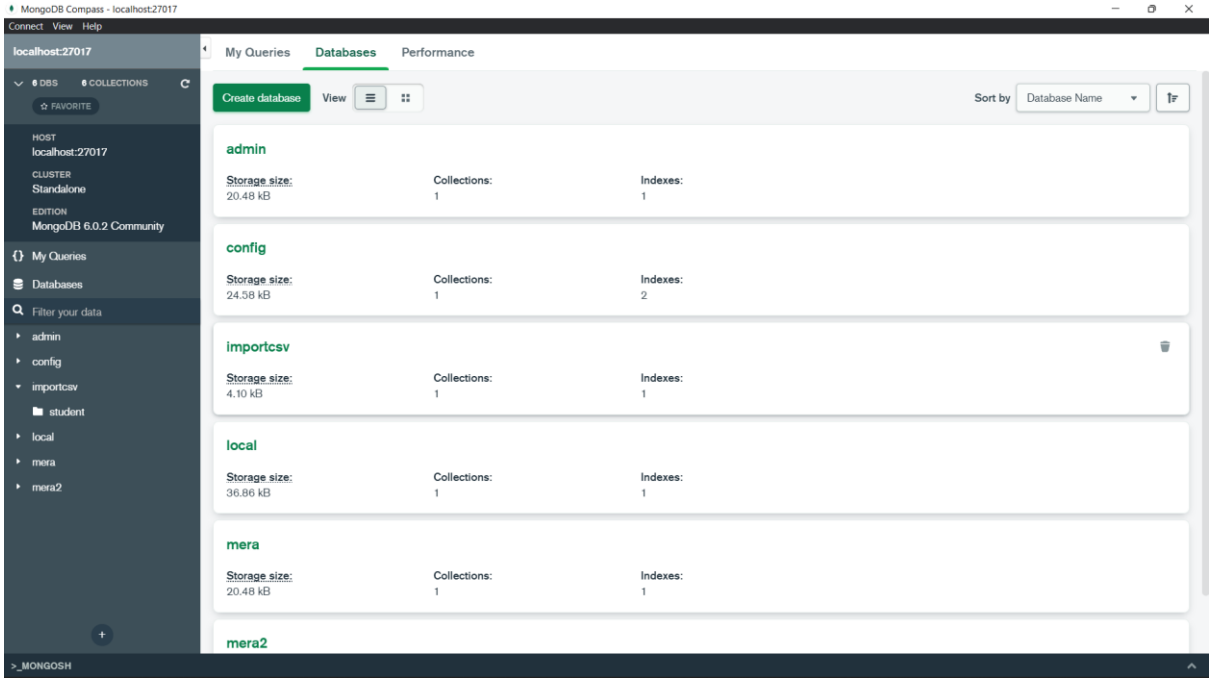
Localhost is 27017 so keep it as it is. Just click on connect button.

Step 2: After click connect. Click on database:

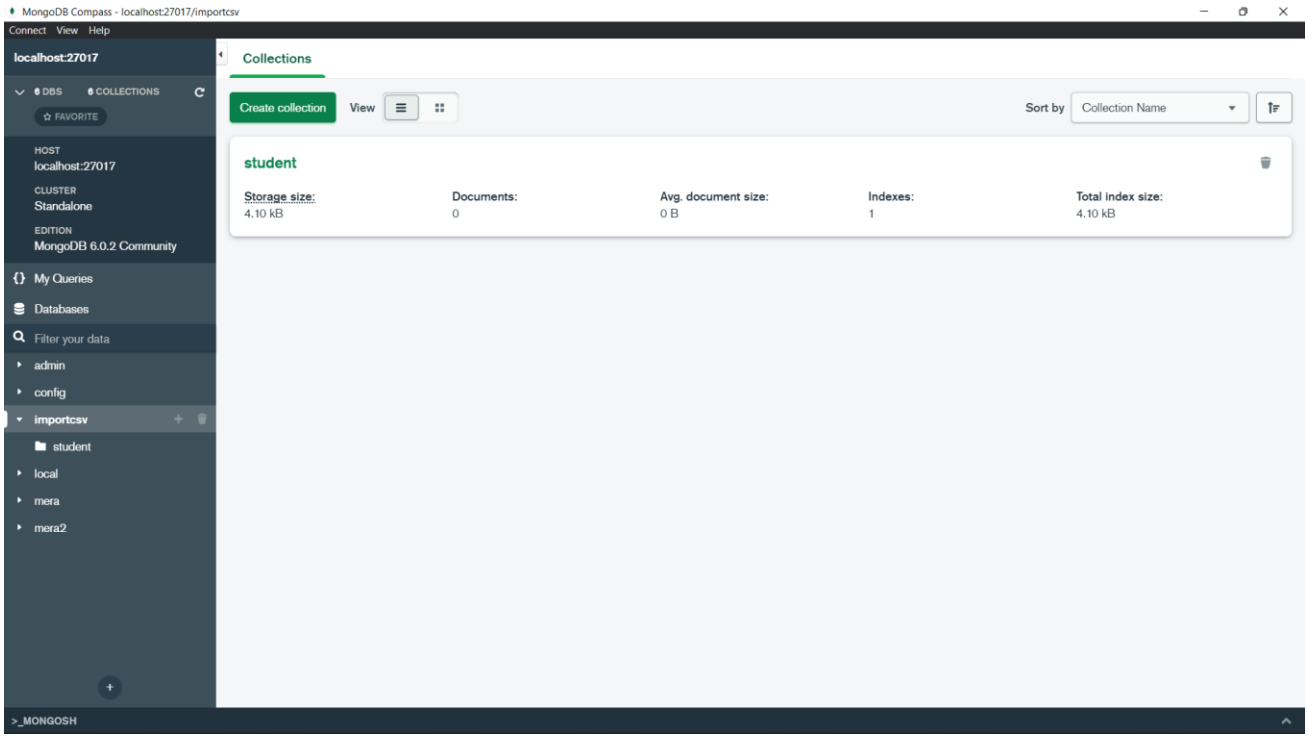


Step 3: Click on create database and create a database and collection of any name.

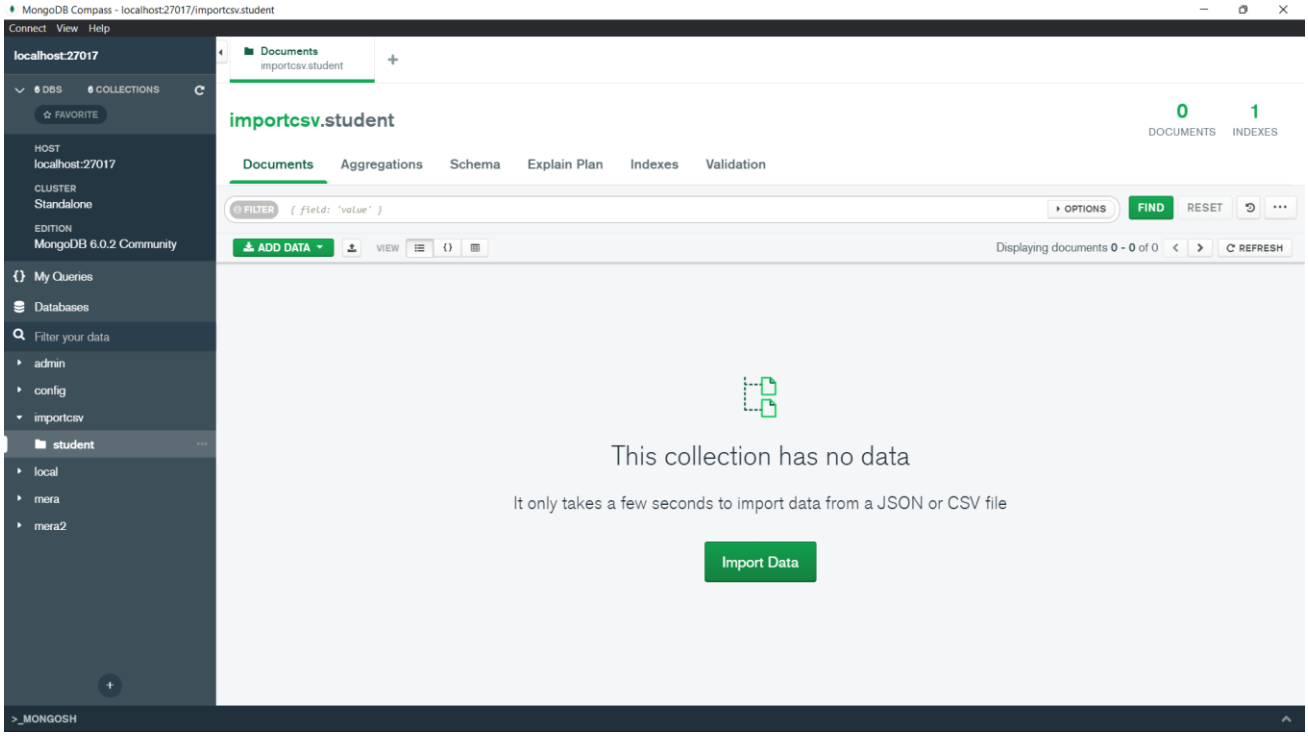
I have create a database name ‘importcsv’



To create a database there should be one collection. I have create a collection name ‘student’.



Step 4: After creating collection and database you will see this screen.

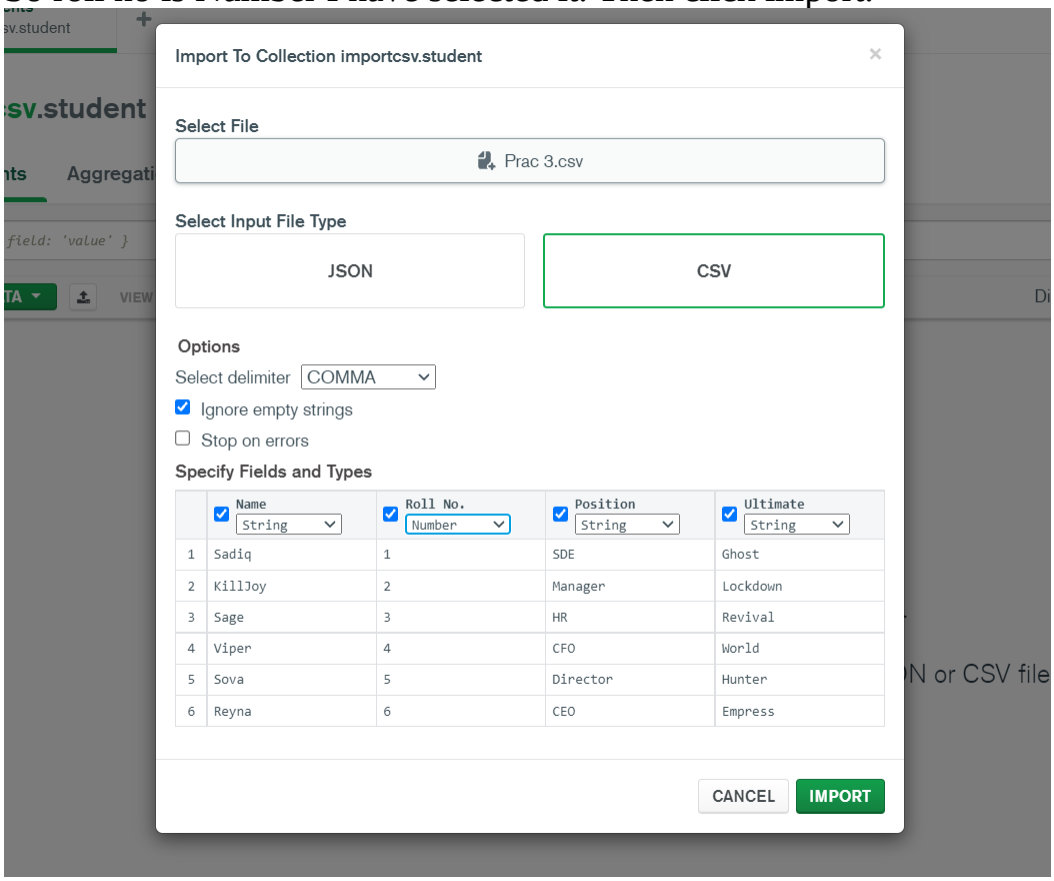


Step 5: Click on import Data:

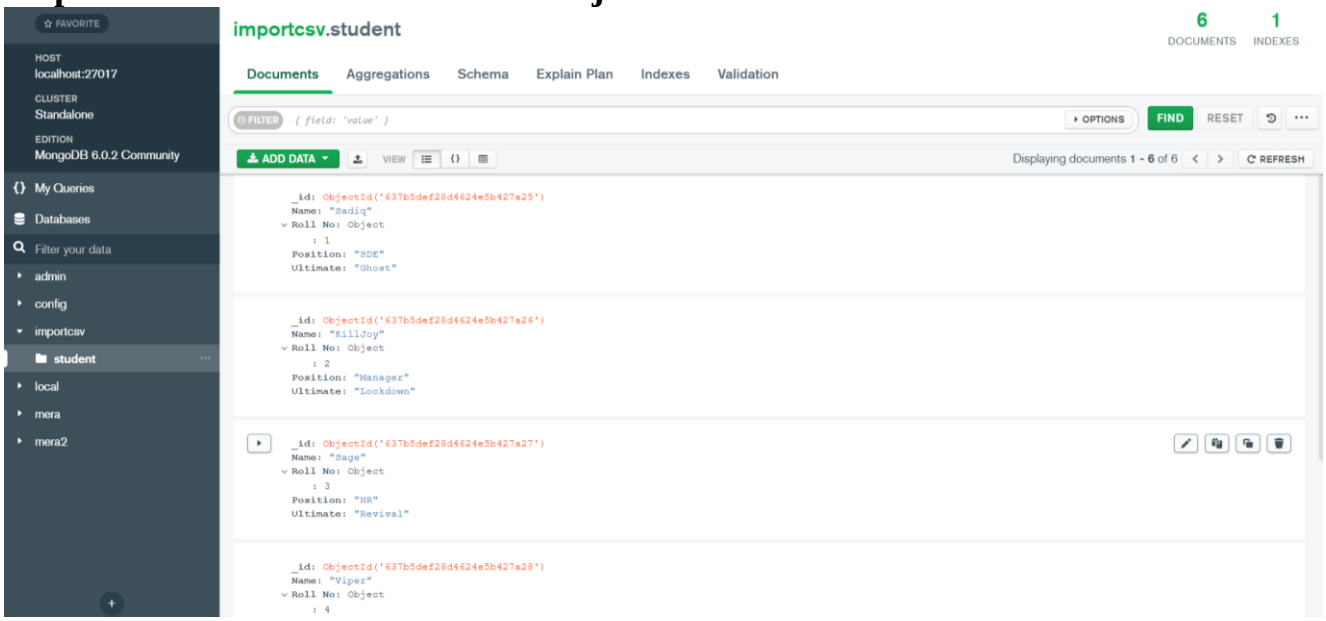
And select file type as csv file. To create a csv file just create excel file and save as csv.

We can select the datatype.

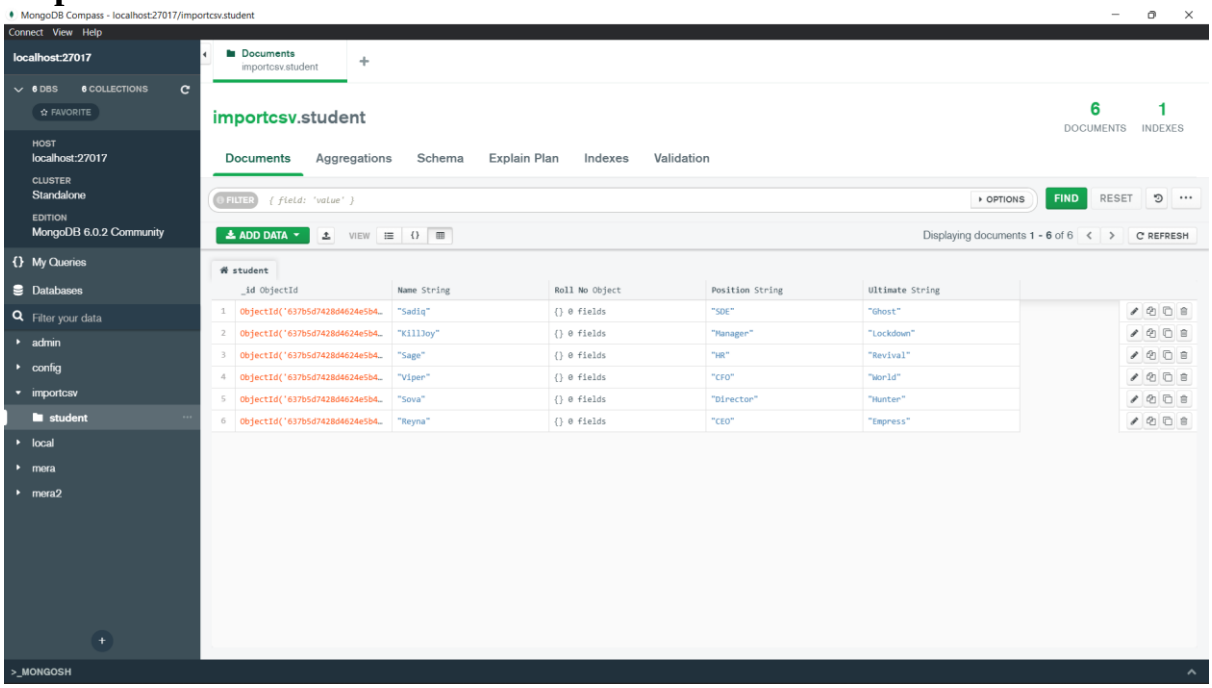
So roll no is Number I have selected it. Then click import.



Step 6: There we will see our data in json format.



Step 7: Click on Table view to view it in table format.



So this is how you can import your csv file in mongodb.

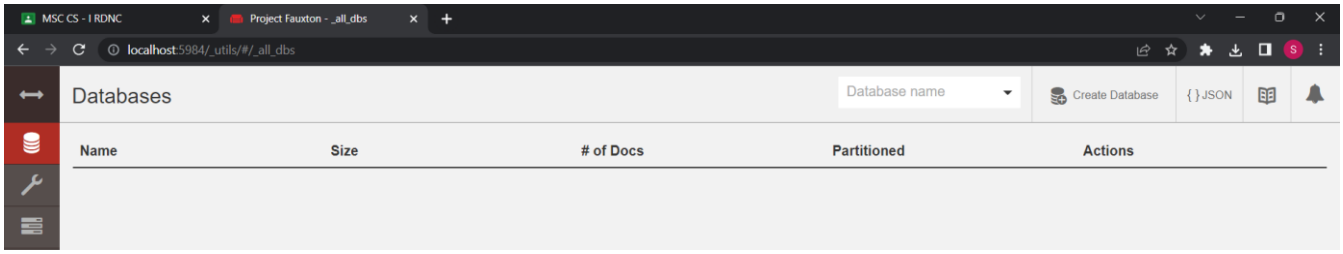
Performing CRUD in Couch DB

Step 1: Install the CouchDB.

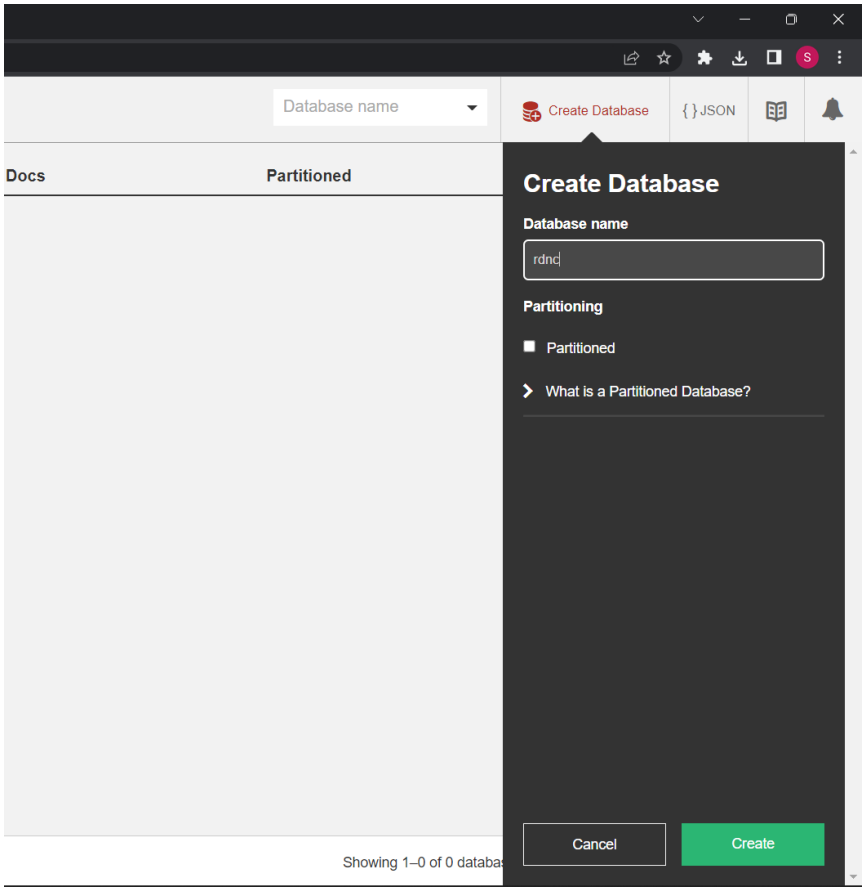
Step 2: Go to the browser and type: http://localhost:5984/

If you get to this screen this means that CouchDB is successfully install and running.

Step 3: Now type http://localhost:5984/_utils and enter your username and password.
After entering the credential, you will see this screen.

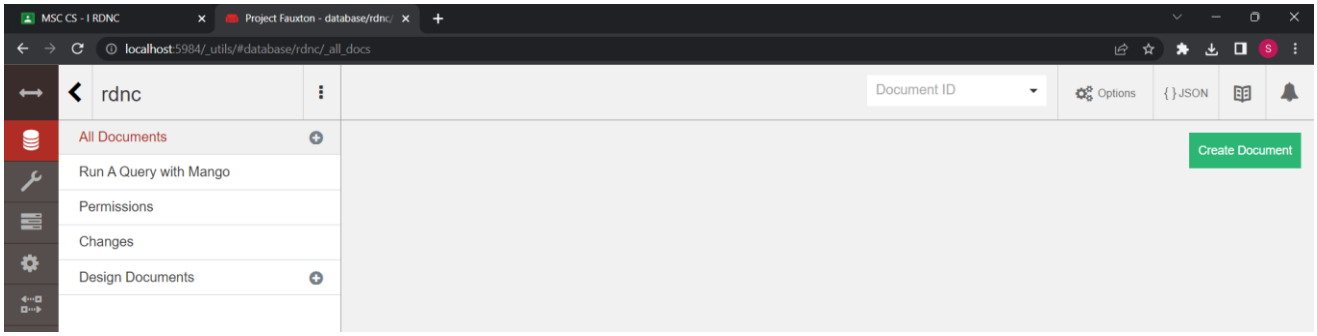


Step 4: At the top right corner there is an option of create database.
Click on the button and it will show the database creation option as given below.
Type the name of database you want and then click create button.

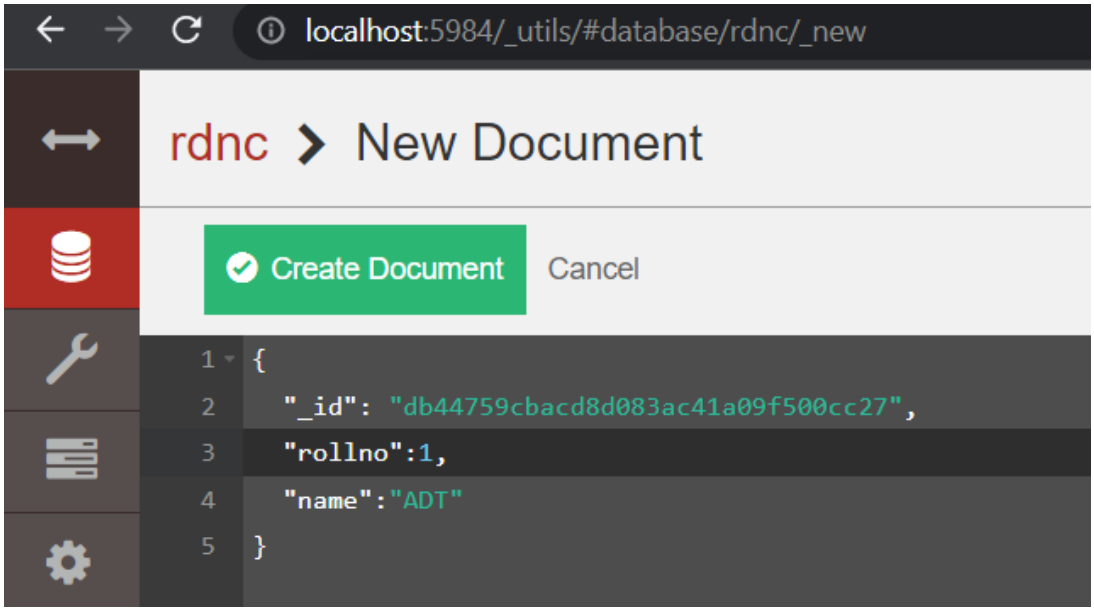


Database name should be in smallcase.

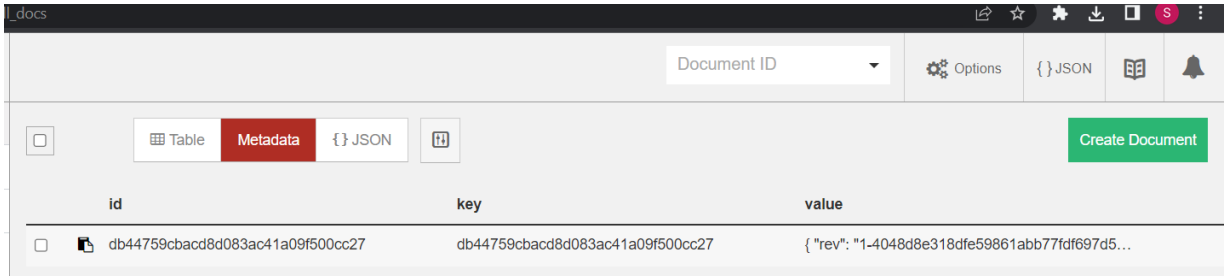
Step 5: After the creation of the database the screen Will look like this. At the top right corner there is an option of create document.



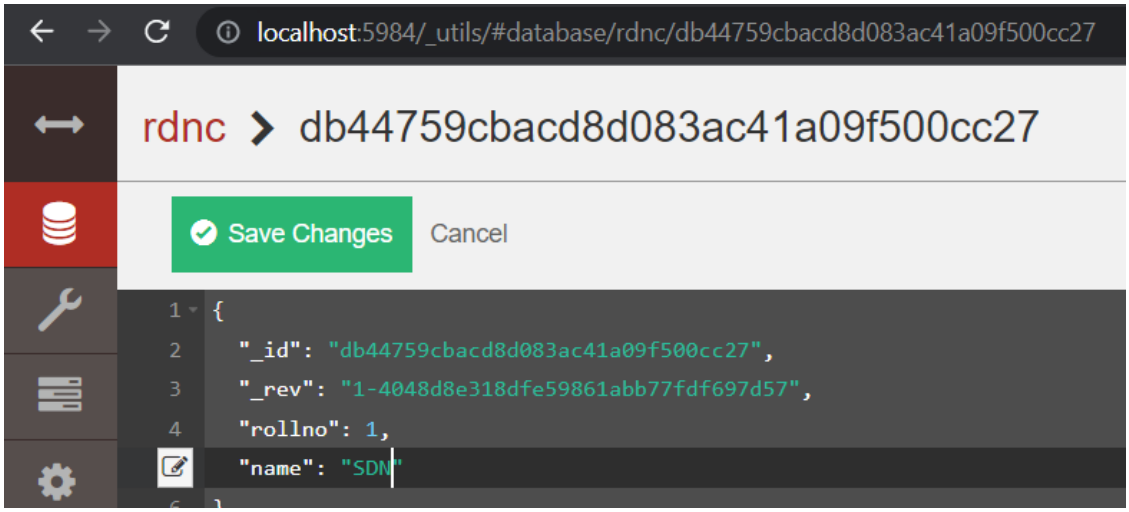
Step 6: Click on the create document button. And, the screen will appear as shown below.
Enter the data of your choice and click on create document button again.



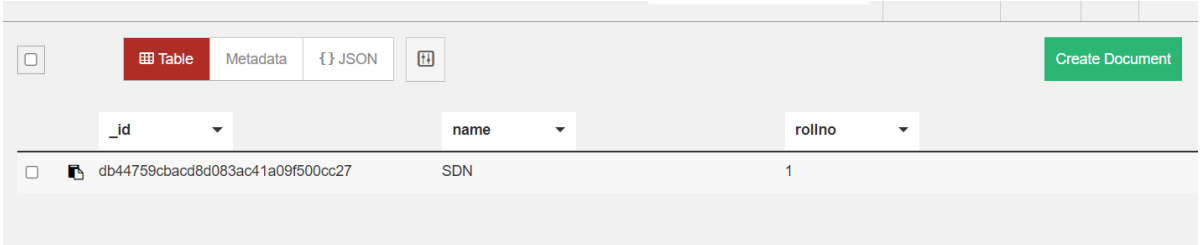
Once the data is been created, you will see your inserted data as shown below.



Step 7: You can update the data by clicking on the data and click on the save changes button on the top left corner and your data will be updated.



After updating you can view your data properly by click on the table and JSON view option given on top middle part.

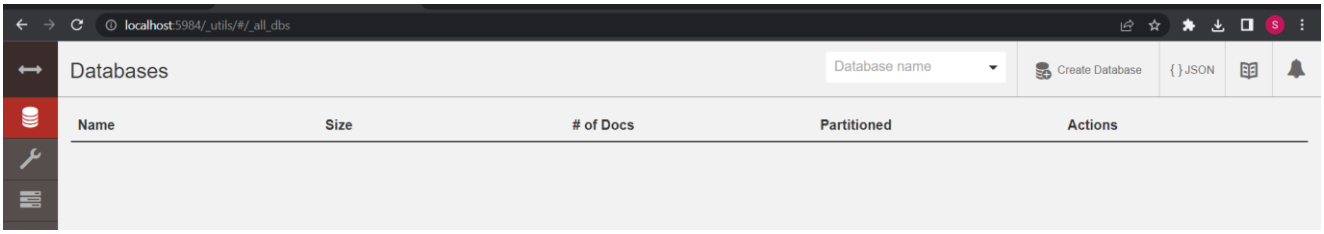


Step 8: After checking the check box of the row you can delete the data by clicking on the delete icon. When you click on delete icon. It will show you a prompt asking for confirmation. Once you confirm it your data will be deleted.

Step 9: Now this below picture shows the name of your database. You can delete your database by clicking on the delete icon.

Step 10: After clicking on the delete icon, you will see the prompt asking for the name of your database, once you type your database name and click on delete button, your database will be deleted.

Below picture shows that there is no database in your databases list.



Sadiq S.

Perform CRUD operation on Redis DB

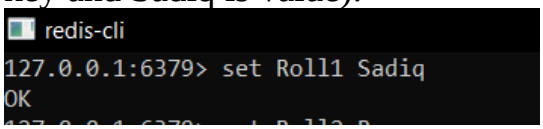
Step 1: Install Redis DB on Windows using **https://github.com/microsoftarchive/redis/releases** .

Step 2: After Installation open Redis Cli. There you will see localhost IP address and the port number.

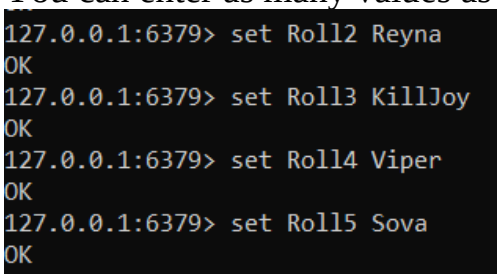
Step 3: Now Server is started we can enter the commands now.
Here we won't be having any table, document or collection. We will be working on set of data in memory.

Step 4: Inserting values in Database.

- To insert values in database use command **set Roll1 Sadiq** (Here Roll1 is key and Sadiq is value).

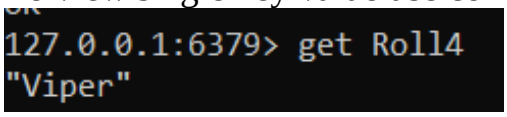


- You can enter as many values as you want.

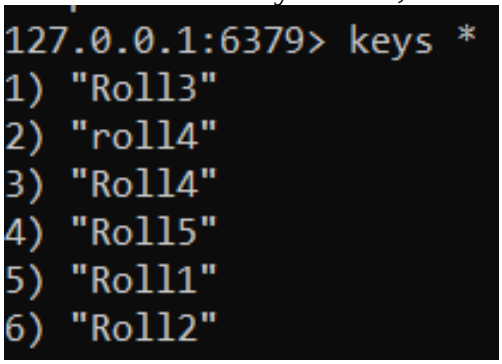


Step 5: View inserted values.

- To view single key value use command, **get Roll4.**



- To view all the key values, use command **keys ***



- To see existing key value use command **exists Roll2**

```
127.0.0.1:6379> exists Roll2
(integer) 1
127.0.0.1:6379> exists Roll3
(integer) 1
127.0.0.1:6379> exists Roll7
(integer) 0
```

As we don't have Roll7 its returning integer as 0 i.e., Roll7 doesn't exist.

Step 6: Update values in database.

- To update values in database use command **set roll3 Sage** (replace your value with some new value)

```
127.0.0.1:6379> set Roll3 Sage
OK
127.0.0.1:6379> get Roll3
"Sage"
127.0.0.1:6379> set Roll5 Cypher
OK
127.0.0.1:6379> get Roll5
"Cypher"
```

Values has been updated.

Step 7: Delete values in Database.

- To delete a single key value use command **del Roll2**

```
5) "Roll2"
127.0.0.1:6379> del Roll2
(integer) 1
127.0.0.1:6379> keys *
1) "Roll3"
2) "roll4"
3) "Roll5"
4) "Roll1"
```

Value has been deleted.

- To delete all key values from the database use command **flushall**

```
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>
```

All the values has been deleted.

Step 8 : To see server information use command info

```
127.0.0.1:6379> info
# Server
redis_version:3.0.504
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a4f7a6e86f2d60b3
redis_mode:standalone
os:Windows
arch_bits:64
multiplexing_api:WinSock_IOCP
process_id:5148
run_id:c06a9bec4ad018289e3eed060cb47bff93c6fc1c
tcp_port:6379
uptime_in_seconds:74285
uptime_in_days:0
hz:10
lru_clock:7580264
config_file:C:\Program Files\Redis\redis.windows-service.conf

# Clients
connected_clients:1
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0

# Memory
used_memory:693424
used_memory_human:677.17K
used_memory_rss:655656
used_memory_peak:693424
used_memory_peak_human:677.17K
used_memory_lua:36864
mem_fragmentation_ratio:0.95
mem_allocator:jemalloc-3.6.0

# Persistence
loading:0
rdb_changes_since_last_save:6
rdb_bgsave_in_progress:0
rdb_last_save_time:1668523987
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
```

```
# Stats
total_connections_received:2
total_commands_processed:20
instantaneous_ops_per_sec:0
total_net_input_bytes:595
total_net_output_bytes:307
instantaneous_input_kbps:0.00
instantaneous_output_kbps:0.00
rejected_connections:0
sync_full:0
sync_partial_ok:0
sync_partial_err:0
expired_keys:0
evicted_keys:0
keyspace_hits:3
keyspace_misses:0
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:10231
migrate_cached_sockets:0

# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0

# CPU
used_cpu_sys:0.22
used_cpu_user:0.36
used_cpu_sys_children:0.00
used_cpu_user_children:0.00

# Cluster
cluster_enabled:0

# Keyspace
127.0.0.1:6379>
```