



NoSQL Technologies

Albar Khan

INDEX

| Sr. No. | Practical List | Pg. No. | Date | Sign |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------|------|
| 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 | Data Storage in Redis a) Implement caching functionality using Redis as a cache store. b) Store and retrieve data from Redis cache using appropriate commands | | | |

Practical – 1A: Lab Exercise: Setting up and Exploring MongoDB

What is MongoDB?

MongoDB is a popular open-source NoSQL database management system that uses a document-oriented data model. It is designed to be flexible and scalable, making it suitable for a wide range of applications. MongoDB stores data in flexible, JSON-like documents called BSON (Binary JSON), allowing developers to model data in a way that is natural for the application.

Key features of MongoDB include:

- **Document-Oriented:** MongoDB stores data in flexible, JSON-like BSON documents. Each document can have a different structure, allowing for easy schema evolution.
- **Scalability:** MongoDB can scale horizontally by sharding data across multiple servers. This allows it to handle large amounts of data and high traffic volumes.
- **Indexing:** MongoDB supports indexes, making it efficient for queries and searches. Indexes can be created on any field, including those within arrays and subdocuments.
- **Aggregation Framework:** MongoDB provides a powerful aggregation framework for data processing and transformation. It supports a variety of operations, such as filtering, grouping, sorting, and projecting.
- **Schema-less Design:** Unlike traditional relational databases, MongoDB does not require a predefined schema. This makes it easier to work with evolving data models.
- **Flexible Schema:** MongoDB allows for flexible schema design, meaning fields can be added to documents without affecting existing data. This flexibility is particularly useful in agile development environments.
- **Ad Hoc Queries:** MongoDB supports ad hoc queries, allowing developers to perform complex queries on the data.
- **Replication:** MongoDB supports replication for high availability. It can maintain multiple copies of data across different servers.
- **Aggregation and Map-Reduce:** MongoDB supports aggregation pipelines and Map-Reduce for complex data processing tasks.

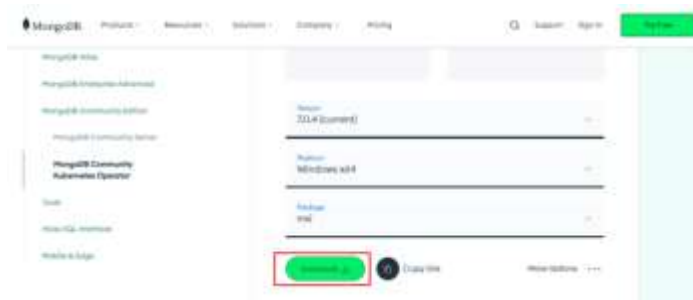
Install MongoDB on your local machine or lab server.

To interact with MongoDB, you can use the MongoDB Shell, which provides a JavaScript-based interface. Additionally, there are drivers and libraries available for various programming languages, making it easy to integrate MongoDB into your application.

MongoDB is commonly used in web development, content management systems, real-time analytics, and other scenarios where flexible and scalable data storage is required.

Steps to install MongoDB on Windows using MSI

Step 1: Go to the [MongoDB Download Center](#) to download the MongoDB Community Server



Here, You can select any version, Windows, and package according to your requirement. For Windows, we need to choose:

- Version: 7.0.4
- OS: Windows x64
- Package: msi

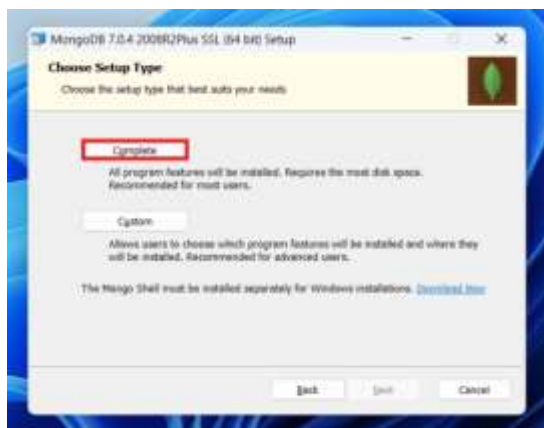
Step 2: When the download is complete open the msi file and click the *next button* in the startup screen:



Step 3: Now accept the End-User License Agreement and click the next button:



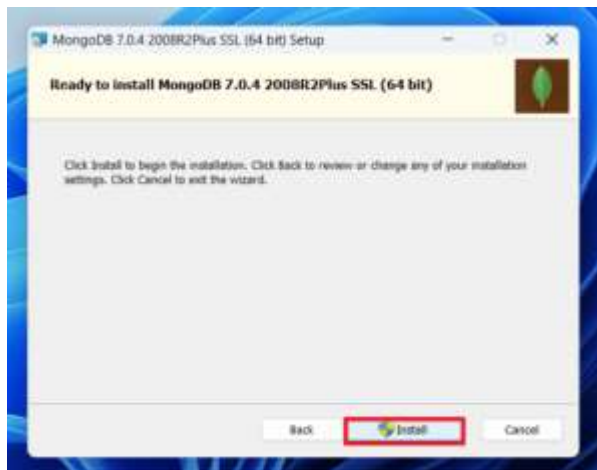
Step 4: Now select the *complete option* to install all the program features. Here, if you can want to install only selected program features and want to select the location of the installation, then use the *Custom option*:



Step 5: Select "Run service as Network Service user" and copy the path of the data directory. Click Next:



Step 6: Click the *Install button* to start the MongoDB installation process:

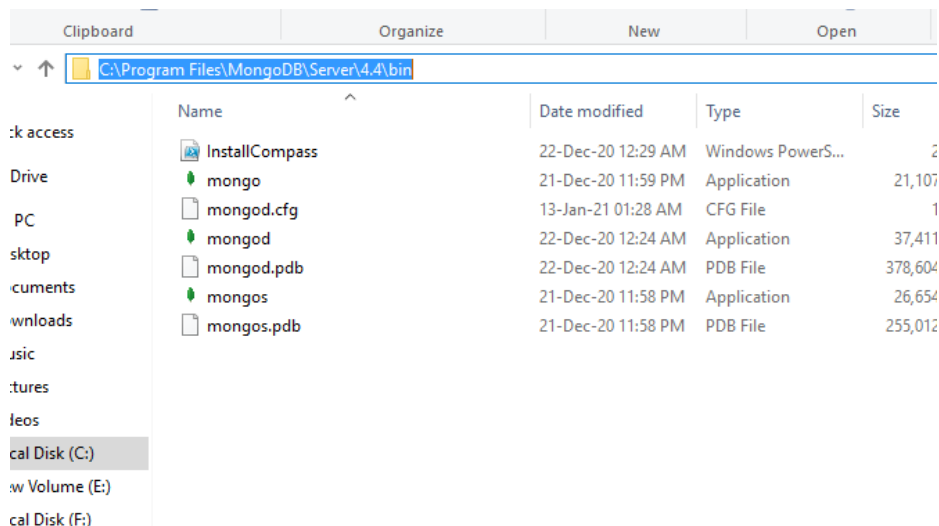


Step 7: After clicking on the install button installation of MongoDB begins:

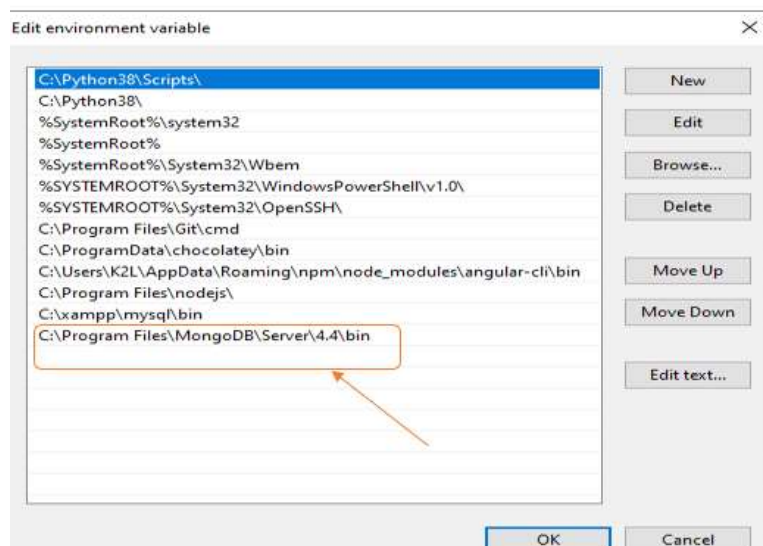


Step 8: Now click the ***Finish button*** to complete the MongoDB installation process:

Step 9: Now we go to the location where MongoDB installed in step 5 in your system and copy the bin path:



Step 10: Now, to create an environment variable open system properties << Environment Variable << System variable << path << Edit Environment variable and paste the copied link to your environment system and click Ok:



Step 11: After setting the environment variable, we will run the MongoDB server, i.e. `mongod`. So, open the command prompt and run the following command:

When you run this command you will get an error i.e. `C:/data/db/ not found`.

Step 12: Now, Open C drive and create a folder named "data" inside this folder create another folder named "db". After creating these folders. Again open the command prompt and run the following command:


```

C:\Users\Nikhil Chhipa>mongod
{"t":{"$date":"2021-01-31T00:56:54.081+05:30"},"s":"I", "c":"CONTROL", "id":23285, "ctx"
ify --sslDisabledProtocols 'none'}}
{"t":{"$date":"2021-01-31T00:56:54.087+05:30"},"s":"I", "c":"ASIO", "id":22601, "ctx"
}
{"t":{"$date":"2021-01-31T00:56:54.088+05:30"},"s":"I", "c":"NETWORK", "id":4648602, "ctx"
{"t":{"$date":"2021-01-31T00:56:54.090+05:30"},"s":"I", "c":"STORAGE", "id":4615611, "ctx"
bPath":"C:/data/db/", "architecture":"64-bit", "host":"DESKTOP-L9MUQ7N"}}
{"t":{"$date":"2021-01-31T00:56:54.090+05:30"},"s":"I", "c":"CONTROL", "id":23398, "ctx"
rgetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2021-01-31T00:56:54.090+05:30"},"s":"I", "c":"CONTROL", "id":23403, "ctx"
gitVersion":"913d6b62acfb344dde1b116f4161360acd8fd13", "modules":[], "allocator":"tcmalloc",
}}
{"t":{"$date":"2021-01-31T00:56:54.090+05:30"},"s":"I", "c":"CONTROL", "id":51765, "ctx"
ndows 10", "version":"10.0 (build 14393)"}
{"t":{"$date":"2021-01-31T00:56:54.090+05:30"},"s":"I", "c":"CONTROL", "id":21951, "ctx"
{"t":{"$date":"2021-01-31T00:56:54.157+05:30"},"s":"I", "c":"STORAGE", "id":22270, "ctx"
:{"dbpath":"C:/data/db/", "storageEngine":"wiredTiger"}}
{"t":{"$date":"2021-01-31T00:56:54.158+05:30"},"s":"I", "c":"STORAGE", "id":22315, "ctx"
ize=1491M, session_max=33000, eviction=(threads_min=4, threads_max=4), config_base=false, statist
le_manager=(close_idle_time=100000, close_scan_interval=10, close_handle_minimum=250), statisti
oss),"}
{"t":{"$date":"2021-01-31T00:56:54.395+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx"
95788][3708:140713908197088], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 20 thr
{"t":{"$date":"2021-01-31T00:56:54.631+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx"

```

Run mongo Shell

Step 13: Now we are going to connect our server (mongod) with the mongo shell. So, keep that mongod window and open a new command prompt window and write **mongo**. Now, our mongo shell will successfully connect to the mongod.

Important Point: Please do not close the mongod window if you close this window your server will stop working and it will not able to connect with the mongo shell.

Check the installation

```

Microsoft Windows [Version 10.0.22621.2438]
(c) Microsoft Corporation. All rights reserved.

C:\Users\chand>mongo --version
db version v7.0.2
Build Info: {
  "version": "7.0.2",
  "gitVersion": "02b1c855e1302209ef00dda6ba3ef8749dd8b62a",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}

C:\Users\chand>mongo
Current Mongosh Log ID: 603f1916e9d0c00a0037cceb
Connecting to: mongod://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB: 7.0.2
Using Mongosh: 2.0.2

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2023-10-30T07:59:04.200+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

test> |

```

C:\Users\chand>mongo --version

db version v7.0.2

Build Info: {

"version": "7.0.2",


```

"gitVersion": "02b3c655e1302209ef046da6ba3ef6749dd0b62a",
"modules": [],
"allocator": "tcmalloc",
"environment": {
  "distmod": "windows",
  "distarch": "x86_64",
  "target_arch": "x86_64"
}
}

```

C:\Users\chand>mongosh

Current Mongosh Log ID: 653f1916e9d0c40a0037cceb

Connecting

to: mongod://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2

Using MongoDB: 7.0.2

Using Mongosh: 2.0.2

For mongosh info see: <https://docs.mongodb.com/mongodb-shell/>

The server generated these startup warnings when booting

2023-10-30T07:59:04.208+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted.

Practical No. 1B: Perform the CRUD Operations

Aim: Perform the CRUD Operations Using

- ❖ MongoDB Database
- ❖ CouchDB Database

Performing CRUD in MongoDB

- Create the Database.

```
test> use masood
switched to db masood
masood>
```

- Display Name of Database.

```
masood> db
masood
masood> |
```

- List down all the database.

```
masood> show dbs
admin      40.00 KiB
config     108.00 KiB
employee   72.00 KiB
local      96.00 KiB
prac2      72.00 KiB
```

- Create a Collection in database.

```
masood> db.createCollection("Student")
{ ok: 1 }
```

- Insert data in collection using "insertOne()" Command.

```
masood> db.Student.insertOne({"roll no":1,"name":"Masood","email":"masood@email.com"})
{
  acknowledged: true,
  insertedId: ObjectId("6701004a9e0f9bf8935683a1")
}
masood> db.Student.insertOne({"roll no":2,"name":"Asad","email":"asad@email.com"})
{
  acknowledged: true,
  insertedId: ObjectId("670100979e0f9bf8935683a2")
}
masood> db.Student.insertOne({"roll no":3,"name":"Ahad","email":"ahad@email.com"})
{
  acknowledged: true,
  insertedId: ObjectId("670100a99e0f9bf8935683a3")
}
```

- Finding data in collection using " find() " Command.

```
masood> db.Student.find()
[
  {
    _id: ObjectId("6701000e9e0f9bf8935683a0"), 'roll no': 1 },
  {
    _id: ObjectId("6701004a9e0f9bf8935683a1"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  },
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ahad',
    email: 'ahad@email.com'
  }
]
```

- Finding data in collection using " find().pretty() " Command.

```
masood> db.Student.find().pretty()
[
  {
    _id: ObjectId("6701000e9e0f9bf8935683a0"), 'roll no': 1 },
  {
    _id: ObjectId("6701004a9e0f9bf8935683a1"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  },
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ahad',
    email: 'ahad@email.com'
  }
]
```

- Insert Multiple data in collection using " insertMany() " Command.

```
masood> db.Student.insertMany([{"roll no":4,"name":"Albar","email":"albar@email.com"},
{"roll no":5,"name":"Bilal","email":"bilal@email.com"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("670102049e0f9bf8935683a4"),
    '1': ObjectId("670102049e0f9bf8935683a5")
  }
}
masood> db.Student.insertMany([{"roll no":6,"name":"Rohan","email":"rohan@email.com"},
{"roll no":7,"name":"Rohit","email":"rohit@email.com"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6701024f9e0f9bf8935683a6"),
    '1': ObjectId("6701024f9e0f9bf8935683a7")
  }
}
```

- Finding data in collection using " find().pretty() " Command.

```

masood> db.Student.find().pretty()
[
  {
    _id: ObjectId("6701000e9e0f9bf8935683a0"), 'roll no': 1 },
  {
    _id: ObjectId("6701004a9e0f9bf8935683a1"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  },
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ahad',
    email: 'ahad@email.com'
  },
  {
    _id: ObjectId("670102049e0f9bf8935683a4"),
    'roll no': 4,
    name: 'Albar',
    email: 'albar@email.com'
  },
  {
    _id: ObjectId("670102049e0f9bf8935683a5"),
    'roll no': 5,
    name: 'Bilal',
    email: 'bilal@email.com'
  },
  {
    _id: ObjectId("6701024f9e0f9bf8935683a6"),
    'roll no': 6,
    name: 'Rohan',
    email: 'rohan@email.com'
  },
]

```

- Update data in collection using "updateOne()" Command.

```

masood> db.Student.updateOne({"roll no":3},{ $set: {"name": "Ruhaan"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

- Showing the data after update.

```
masood> db.Student.find({"roll no":3})
[
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ahad@email.com'
  }
]
```

- Updating and finding data by specific value

```
masood> db.Student.find({"roll no":3})
[
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ahad@email.com'
  }
]
masood> db.Student.updateOne({"roll no":3},{ $set:{"email":"ruhaan@email.com"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
masood> db.Student.find({"roll no":3})
[
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com'
  }
]
```

- Using Update command for inserting/adding new data in field.

```

masood> db.Student.updateOne({"roll no":3},{ $set:{"Phone":"9876543214"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
masood> db.Student.find({"roll no":3})
[
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com',
    Phone: '9876543214'
  }
]

```

- Deleting data in collection using " deleteOne() " Command.

```

masood> db.Student.deleteOne({"roll no":6})
{ acknowledged: true, deletedCount: 1 }
masood> db.Student.deleteOne({"roll no":4})
{ acknowledged: true, deletedCount: 1 }
masood> db.Student.find().pretty()
[
  { _id: ObjectId("6701000e9e0f9bf8935683a0"), 'roll no': 1 },
  {
    _id: ObjectId("6701004a9e0f9bf8935683a1"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  },
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com',
    Phone: '9876543214'
  },
  {
    _id: ObjectId("670102049e0f9bf8935683a5"),
    'roll no': 5,
    name: 'Bilal',
    email: 'bilal@email.com'
  },
  {
    _id: ObjectId("6701024f9e0f9bf8935683a7"),
    'roll no': 7,
    name: 'Rohit',
    email: 'rohit@email.com'
  }
]

```

- Deleting Multiple data in collection using " deleteMany() " Command.


```

masood> db.Student.deleteMany({"roll no":1})
{ acknowledged: true, deletedCount: 2 }
masood> db.Student.find().pretty()
[
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com',
    Phone: '9876543214'
  },
  {
    _id: ObjectId("670102049e0f9bf8935683a5"),
    'roll no': 5,
    name: 'Bilal',
    email: 'bilal@email.com'
  },
  {
    _id: ObjectId("6701024f9e0f9bf8935683a7"),
    'roll no': 7,
    name: 'Rohit',
    email: 'rohit@email.com'
  }
]

```

- Drop the collection using " Drop() " Command.

```

masood> db.Student.drop()
true

```

- After drop showing the Collection None.

```

masood> show collections

```

- Drop the Database using " dropDatabase() " Command.

```

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

```

Practical - 2: Interacting with Redis

What is REDIS?

Redis is an open-source, in-memory data structure store that can be used as a database, cache, and message broker. It is known for its speed and versatility, making it a popular choice for various applications that require high-performance data storage and retrieval. Here are some key features and aspects of Redis. Redis full form is Remote Directory Server. Redis is a popular NO-SQL in-memory data structure store that is often used as a database, cache, and message broker. It is widely used by developers and system administrators to improve the performance and scalability of their applications. Redis, short for Remote Dictionary Server, is an open source, in-memory datastore. You can use it as a fast database, application cache solution, streaming engine, and more. Redis is a highly reliable, fast, and high-performing solution thanks to its ability to store data in RAM.

When applications rely on external data sources, they experience a significant latency, limiting their performance. To work around this limitation and improve application performance, you can store and operate on its data in RAM instead, bringing it closer to the application.

This solution is where Redis shines. It stores all the necessary external data in RAM to deliver the highest possible read and write performance. Furthermore, it enables replication to place data physically closer to reduce latency.

Here are some core features of Redis:

- Programmability using Lua and Redis Functions.
- Module API for writing custom Rust, C, and C++ extensions.
- Support for horizontal scalability with hash-based sharding.
- Replication with automatic failover.
- Ability to write to storage devices to survive shutdowns and reboots.

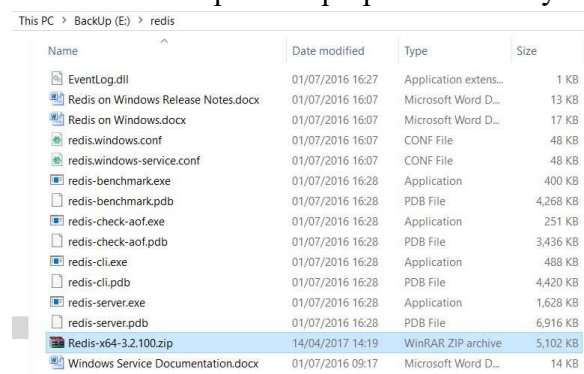
Install Redis on your lab server or local machine.

You can choose to download different versions or the latest version of Redis
github.com/MSOpenTech/redis/releases



1. Download either .msi or .zip file, this tutorial will let you download latest zip file Redis-x64-3.2.100.zip.

2. Extract the zip file to prepared directory.



3. Run redis-server.exe, you can either directly run redis-server.exe by clicking or run via command prompt.

This PC > BackUp (E:) > redis

| Name | Date modified | Type | Size |
|-------------------------------------|------------------|-----------------------|----------|
| EventLog.dll | 01/07/2016 16:27 | Application extens... | 1 KB |
| Redis on Windows Release Notes.docx | 01/07/2016 16:07 | Microsoft Word D... | 13 KB |
| Redis on Windows.docx | 01/07/2016 16:07 | Microsoft Word D... | 17 KB |
| redis.windows.conf | 01/07/2016 16:07 | CONF File | 48 KB |
| redis.windows-service.conf | 01/07/2016 16:07 | CONF File | 48 KB |
| redis-benchmark.exe | 01/07/2016 16:28 | Application | 400 KB |
| redis-benchmark.pdb | 01/07/2016 16:28 | PDB File | 4,268 KB |
| redis-check-aof.exe | 01/07/2016 16:28 | Application | 251 KB |
| redis-check-aof.pdb | 01/07/2016 16:28 | PDB File | 3,436 KB |
| redis-cli.exe | 01/07/2016 16:28 | Application | 488 KB |
| redis-cli.pdb | 01/07/2016 16:28 | PDB File | 4,420 KB |
| redis-server.exe | 01/07/2016 16:28 | Application | 1,628 KB |
| redis-server.pdb | 01/07/2016 16:28 | PDB File | 6,916 KB |
| Redis-x64-3.2.100.zip | 14/04/2017 14:19 | WinRAR ZIP archive | 5,102 KB |
| Windows Service Documentation.docx | 01/07/2016 09:17 | Microsoft Word D... | 14 KB |

4. Run redis-cli.exe, after successfully running the redis-server. You can access it and test commands by running redis-cli.exe

E:\redis\redis-cli.exe

```
127.0.0.1:6379>
```

5. PING command is used to test if a connection is still alive

E:\redis\redis-cli.exe

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ping "hello world"
"hello world"
127.0.0.1:6379>
```

- 1) Install Redis on your local machine or lab server.**
- 2) Store and retrieve data in Redis using various data structures like strings, lists, and sets.**
- 3) Implement basic Redis commands for data manipulation and retrieval**

GET & SET For adding and view the Inserted Data

Redis – Strings

```
127.0.0.1:6379> set company_name nextgenpixel
```

OK

```
127.0.0.1:6379> get company_name
```

"nextgenpixel"

```
127.0.0.1:6379> setex brand_name 10 pixel
```

OK

```
127.0.0.1:6379> get brand_name
```

"pixel"

```
127.0.0.1:6379> get brand_name
```

(nil)

```
127.0.0.1:6379> set company_id 22
```

OK

```
127.0.0.1:6379> keys *
```

1) "company_id"

2) "company_name"

3) "name"

del to delete the Data

```
127.0.0.1:6379> del name
```

(integer) 1

```
127.0.0.1:6379> keys *
```

1) "company_id"

2) "company_name"

TTL stands for Time to Live. In Redis, it's a way to set an expiry time on a key

```
127.0.0.1:6379> ttl head
```

```
(integer) -2
```

```
127.0.0.1:6379> expire head 10
```

```
(integer) 0
```

```
127.0.0.1:6379> set phone_no 92959252952
```

```
OK
```

keys* to show all keys

```
127.0.0.1:6379> keys *
```

```
1) "phone_no"
```

```
2) "company_id"
```

```
3) "company_name"
```

```
127.0.0.1:6379> expire phone_no 10
```

```
(integer) 1
```

```
127.0.0.1:6379> keys *
```

```
1) "phone_no"
```

```
2) "company_id"
```

```
3) "company_name"
```

```
127.0.0.1:6379> keys *
```

```
1) "company_id"
```

```
2) "company_name"
```

Set the with expiration time

```
127.0.0.1:6379> set address mumbai EX 5
```

```
OK
```

```
127.0.0.1:6379> get address
```

```
(nil)
```

```
127.0.0.1:6379> append company_name .co.in
```

```
(integer) 18
```

```
127.0.0.1:6379> get company_name
```

```
"nextgenpixel.co.in"
```

Update the Data

```
127.0.0.1:6379> append company_name website
```

```
(integer) 25
```

```
127.0.0.1:6379> get company_name
```

```
"nextgenpixel.co.inwebsite"
```

Shows the length

```
127.0.0.1:6379> strlen company_name
```

```
(integer) 25
```

```
127.0.0.1:6379> keys *
```

```
1) "company_id"
```

```
2) "company_name"
```

Delete the Key

```
127.0.0.1:6379> del company_id
```

```
(integer) 1
```

```
127.0.0.1:6379> keys *
```

```
1) "company_name"
```

Delete all Existing Keys

```
127.0.0.1:6379> flushall
```

```
OK
```

```
127.0.0.1:6379> keys *
```

```
(empty list or set)
```



```

127.0.0.1:6379> set company_name nextgenpixel
OK
127.0.0.1:6379> get company_name
"nextgenpixel"
127.0.0.1:6379> setex brand_name 10 pixel
OK
127.0.0.1:6379> get brand_name
"pixel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 21
OK
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone_no 92900252902
OK
127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> expire phone_no 10
(integer) 1
127.0.0.1:6379> keys *
1) "phone_no"

```

```

127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> set address mumbai Ex 5
OK
127.0.0.1:6379> get address
(nil)
127.0.0.1:6379> append company_name .co.in
(integer) 18
127.0.0.1:6379> get company_name
"nextgenpixel.co.in"
127.0.0.1:6379> append company_name - website
(error) ERR wrong number of arguments for 'append' command
127.0.0.1:6379> append company_name website
(integer) 25
127.0.0.1:6379> get company_name
"nextgenpixel.co.inwebsite"
127.0.0.1:6379> strlen company_name
(integer) 25
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> del company_id
(integer) 1
127.0.0.1:6379> keys *
1) "company_name"
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>

```

Redis for LIST

Redis – Lists Redis Lists are simply lists of strings, sorted by insertion order. You can add elements in Redis lists in the head or the tail of the list. Maximum length of a list is 232 - 1 elements (4294967295, more than 4 billion of elements per list).

LPUSH - Prepends one or multiple values to a list

```
127.0.0.1:6379> LPUSH compay_name nextgenpixel
```

```
(integer) 1
```

```
127.0.0.1:6379> LPUSH compay_name adobe
```

```
(integer) 2
```

```
127.0.0.1:6379> LPUSH compay_name frolikpixel
```

```
(integer) 3
```

LRANGE – Shows data from the specific

```
127.0.0.1:6379> LRANGE company_name 0 10
```

```
1) "nextgenpixel,adobe,amazon"
```

```
127.0.0.1:6379> LPUSH brand_name manyavar centric xbox
```

```
(integer) 3
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox"
```

```
2) "centric"
```

```
3) "manyavar"
```

RPUSH – (Updates) Appends one or multiple values to a listlist

```
127.0.0.1:6379> RPUSH brand_name web_engage
```

```
(integer) 4
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox"
```

```
2) "centric"
```

```
3) "manyavar"
```

```
4) "web_engage"
```

```
127.0.0.1:6379> LINDEX brand_name 2
```

```
"manyavar"
```

LSET – Sets the value of an element in a list by its index

```
127.0.0.1:6379> LSET brand_name 1 xbox360
```

```
OK
```

LRANGE - Gets a range of elements from a list

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox"
```

```
2) "xbox360"
```

```
3) "manyavar"
```

```
4) "web_engage"
```

LPOP - Removes and gets the first element in a list

```
127.0.0.1:6379> LPOP brand_name
```

```
"xbox"
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox360"
```

```
2) "manyavar"
```

```
3) "web_engage"
```

```
127.0.0.1:6379> RPOP brand_name
```

```
"web_engage"
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox360"
```

```
2) "manyavar"
```

BLPOP - Removes and gets the first element in a list, or blocks until one is available

```
127.0.0.1:6379> BLPOP brand_name 5
```

```
1) "brand_name"
```

```
2) "xbox360"
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "manyavar"
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "manyavar"
```

```
127.0.0.1:6379> LPUSH brand_name blubirch samsung
```

```
(integer) 3
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "samsung"
```

```
2) "blubirch"
```

```
3) "manyavar"
```

```
127.0.0.1:6379>
```

```

127.0.0.1:6379> set company_name rastgeeriel
OK
127.0.0.1:6379> get company_name
"rastgeeriel"
127.0.0.1:6379> set brand_name 16 jiel
OK
127.0.0.1:6379> get brand_name
"jiel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 22
OK
127.0.0.1:6379> keys +
3) "company_id"
3) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys +
3) "company_id"
3) "company_name"
127.0.0.1:6379> set head
(integer) -1
127.0.0.1:6379> expire head 16
(integer) 0
127.0.0.1:6379> set phone_no 000002002
OK
127.0.0.1:6379> keys +
3) "phone_no"
3) "company_id"
3) "company_name"
127.0.0.1:6379> expire phone_no 16
(integer) 1
127.0.0.1:6379> keys +
3) "company_id"
3) "company_name"
3) "phone_no"
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> set company_name rastgeeriel

```

```

127.0.0.1-5770- LFWG company_name testipwired
(listener) 1
127.0.0.1-5770- LFWG company_name adobe
(listener) 2
127.0.0.1-5770- LFWG company_name freilipwired
(listener) 3
127.0.0.1-5770- LKAGUC company_name 0 10
2) "testipwired.adobe.nazare"
127.0.0.1-5770- LFWG brand_name alkyacac catric ibea
(listener) 0
127.0.0.1-5770- LKAGUC brand_name 0 10
1) "adac"
2) "catric"
3) "nazyacac"
127.0.0.1-5770- KFWU brand_name web_engage
(listener) 4
127.0.0.1-5770- LKAGUC brand_name 0 10
2) "adac"
2) "catric"
3) "nazyacac"
4) "web_engage"
127.0.0.1-5770- LKAGUC brand_name 2
"nazyacac"
127.0.0.1-5770- LKST brand_name 3 abx300
04
127.0.0.1-5770- LKAGUC brand_name 0 10
2) "adac"
2) "catric"
3) "nazyacac"
4) "web_engage"
127.0.0.1-5770- LFWG brand_name
"adac"
127.0.0.1-5770- LKAGUC brand_name 0 10
2) "catric"
3) "nazyacac"

```

[illegible]

REDIS -SETs

Redis – Sets

Redis Sets are an unordered collection of unique strings. Unique means sets does not allow repetition of data in a key. In Redis set add, remove, and test for the existence of members in $O(1)$ (constant time regardless of the number of

elements contained inside the Set). The maximum length of a list is $2^{32} - 1$ elements (4294967295, more than 4 billion of elements per set).

```
127.0.0.1:6379> set company_name nextgenpixel
```

OK

```
127.0.0.1:6379> get company_name
```

"nextgenpixel"

```
127.0.0.1:6379> setex brand_name 10 pixel
```

OK

```
127.0.0.1:6379> get brand_name
```

"pixel"

```
127.0.0.1:6379> get brand_name
```

(nil)

```
127.0.0.1:6379> set company_id 22
```

OK

```
127.0.0.1:6379> keys *
```

1) "company_id"

2) "company_name"

3) "name"

```
127.0.0.1:6379> del name
```

(integer) 1

```
127.0.0.1:6379> keys *
```

1) "company_id"

2) "company_name"

```
127.0.0.1:6379> ttl head
```

(integer) -2

```
127.0.0.1:6379> expire head 10
```

(integer) 0

```
127.0.0.1:6379> set phone_no 92959252952
```

OK

```
127.0.0.1:6379> keys *
```

1) "phone_no"

2) "company_id"

3) "company_name"

127.0.0.1:6379> expire phone_no 10

(integer) 1

127.0.0.1:6379> keys *

SADD - Adds one or more members to a set

127.0.0.1:6379> SADD company_name bluebirch zentaix biretail

(integer) 3

SMEMBERS - Gets all the members in a set

127.0.0.1:6379> SMEMBERS company_name

1) "biretail"

2) "zentaix"

3) "bluebirch"

SCARD - Gets the number of members in a set

127.0.0.1:6379> SCARD company_name

(integer) 3

127.0.0.1:6379> SADD brand_name posiflex

(integer) 1

127.0.0.1:6379> SADD brand_name dinglix

(integer) 1

127.0.0.1:6379> SADD brand_name spacex

(integer) 1

127.0.0.1:6379> SADD brand_name valience

(integer) 1

127.0.0.1:6379> SMEMBERS brand_name

1) "valience"

2) "dinglix"

3) "spacex"

4) "posiflex"

```
127.0.0.1:6379> SCARD brand_name
```

```
(integer) 4
```

```
127.0.0.1:6379> SCARD brand_name key
```

```
(error) ERR wrong number of arguments for 'scard' command
```

```
127.0.0.1:6379> SCARD brand_name
```

```
(integer) 4
```

SISMEMBER - Determines if a given value is a member of a set

```
127.0.0.1:6379> SISMEMBER brand_name spacex
```

```
(integer) 1
```

SREM - Removes one or more members from a set

```
127.0.0.1:6379> SREM spacex
```

```
(error) ERR wrong number of arguments for 'srem' command
```

```
127.0.0.1:6379> SREM brand_name spacex
```

```
(integer) 1
```

SMEMBERS - Gets all the members in a set

```
127.0.0.1:6379> SMEMBERS brand_name
```

```
1) "valience"
```

```
2) "dinglix"
```

```
3) "posiflex"
```

del to remove

```
127.0.0.1:6379> del brand_name
```

```
(integer) 1
```

```
127.0.0.1:6379> SMEMBERS brand_name
```

```
(empty list or set)
```

```
127.0.0.1:6379>
```



```

107 # W 1:07700 LGH101 company_name whole
108 linktag3 2
109 # W 1:10000 LGH101 company_name fullinplace
110 linktag3 3
111 # W 1:1:07700 LGH101 company_name 0 10
112 # "recompilable_astable_manager"
113 # W 1:07700 LGH101 brand_name manufacturer contact whole
114 linktag3 1
115 # W 1:07700 LGH101 brand_name 0 10
116 # "whole"
117 # "whole"
118 # "whole"
119 # W 1:1:07700 LGH101 brand_name web-engage
120 linktag3 3
121 # W 1:1:07700 LGH101 brand_name 0 10
122 # "whole"
123 # "whole"
124 # "web-engage"
125 # W 1:1:07700 LGH101 brand_name 2
126 # "whole"
127 # W 1:07700 LGH101 company_name blackbox portable Rieftail
128 linktag3 3
129 # W 1:07700 LGH101 company_name
130 # "blackbox"
131 # "blackbox"
132 # "blackbox"
133 # W 1:1:07700 LGH101 company_name
134 linktag3 3
135 # W 1:1:07700 LGH101 brand_name profile
136 linktag3 3
137 # W 1:1:07700 LGH101 brand_name directly
138 linktag3 3
139 # W 1:1:07700 LGH101 brand_name square
140 linktag3 3

```

[illegible]

```

127.0.0.1:6379> set company_name nextgenpixel
OK
127.0.0.1:6379> get company_name
"nextgenpixel"
127.0.0.1:6379> setex brand_name 10 pixel
OK
127.0.0.1:6379> get brand_name
"pixel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 22
OK
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone_no 92089202902
OK
127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> expire phone_no 10
(integer) 1
127.0.0.1:6379> keys *
127.0.0.1:6379> LPUSH compay_name nextgenpixel

```

Practical No. 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 a CRUD operations and query data from Apache Cassandra.

What is Apache Cassandra?

Apache Cassandra is a well-established wide-column NoSQL database. It uses a column-based storage model to capture large amounts of unstructured data.

Cassandra focuses on operating in a distributed cluster of commodity servers and boasts high-availability and flexible horizontal scaling. Apache Cassandra is a highly scalable and distributed NoSQL database system designed to handle large amounts of data across multiple commodity servers without any single point of failure. It is an open-source project managed by the Apache Software Foundation. Cassandra was initially developed by Facebook and later open-sourced as Apache Cassandra in 2008.

Key features of Apache Cassandra include:

- **Distributed Architecture:** Cassandra is designed to distribute data across multiple nodes in a cluster, providing high availability and fault tolerance. This makes it suitable for handling large-scale distributed data.
- **No Single Point of Failure:** Cassandra does not have a single point of failure. Each node in the cluster is independent, and data is replicated across nodes to ensure resilience against hardware failures.
- **Scalability:** Cassandra is horizontally scalable, meaning you can add more nodes to the cluster to handle increased data and traffic. This makes it well-suited for large and growing datasets.
- **High Write and Read Throughput:** Cassandra is optimized for high write and read throughput, making it suitable for applications that require fast data writes and reads.
- **Flexible Schema:** Cassandra has a flexible schema model, allowing developers to define the structure of data on a per-query basis. This flexibility is particularly useful for applications with evolving data requirements.
- **Query Language (CQL):** Cassandra Query Language (CQL) is a SQL-like language used for interacting with Cassandra databases. It provides a familiar syntax for developers who are accustomed to working with relational databases.
- **Tunable Consistency:** Cassandra allows users to tune the consistency level for read and write operations. This means that developers can choose between strong consistency and eventual consistency based on their application's requirements.
- **Support for Multi-Datcenter Replication:** Cassandra supports multi-datcenter deployments, allowing data to be replicated across geographically distributed locations for improved availability and disaster recovery.

Cassandra is commonly used in applications that require high availability, fault tolerance, and scalability, such as real-time big data analytics, time-series data, and recommendation engines. It has found adoption in various industries, including finance, healthcare, and e-commerce.

Install and configure Apache Cassandra in a lab environment.

data, and recommendation engines. It has found adoption in various industries, including finance, healthcare, and e-commerce. Dependencies: Apache Cassandra requires Java 8 to run on a Windows system. Additionally, the Cassandra command-line shell (cqlsh) is dependent on Python 2.7 to work correctly.

To be able to install Cassandra on Windows, first you need to:

1. Download and Install Java 8 and set environment variables.
2. Download and install Python 2.7 and set environment variables.

If you already have these dependencies installed, check your version of Python and Java. If you have Java 8 and Python 2.7. feel free to move on to the third section of this guide.

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

1. Install Java 8 on Windows.

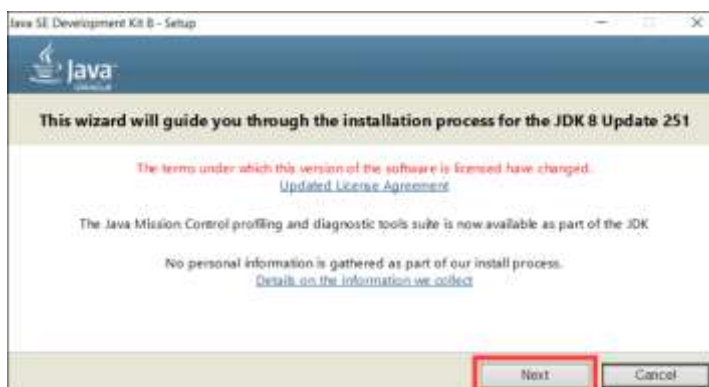
The Java development kit contains all the tools and software you need to run applications written in Java. It is a prerequisite for software solutions such as Apache Cassandra.

Download Oracle JDK 8 (Java Development Kit)

1. Visit the official Oracle download page and download the Oracle JDK 8 software package.
2. Scroll down and locate the Java SE Development Kit 8u251 for Windows x64 download link. The Java 8 download starts automatically after signup.



3. Once the download is complete, double-click the downloaded executable file. Select Next on the initial installation screen.



3. The following section allows you to select optional features and define the location of the installation folder. Accept the default settings and take note of the full path to the installation folder, C: Program FilesJavajdk1.8.0_251. Once you are ready to proceed with the installation, click Next.



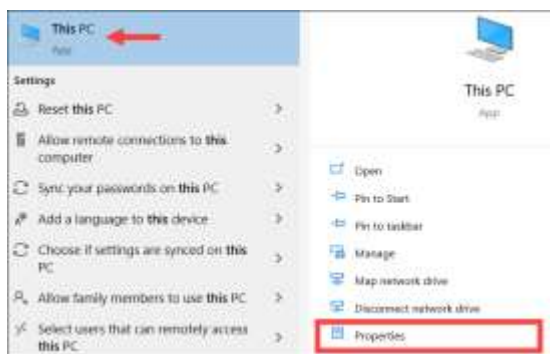
5. The installation process can take several minutes. Select Close once the process is completed.



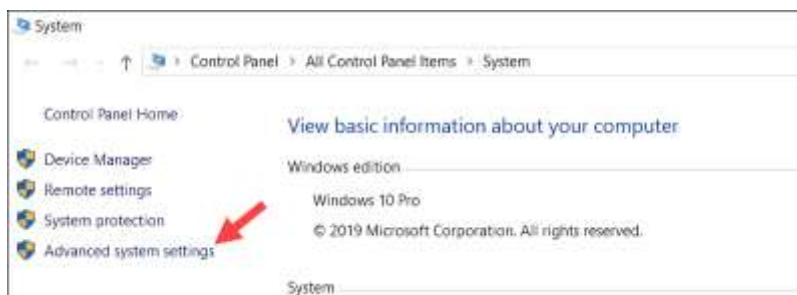
Configure Environment Variables for Java 8

It is vital to configure the environment variables in Windows and define the correct path to the Java 8 installation folder.

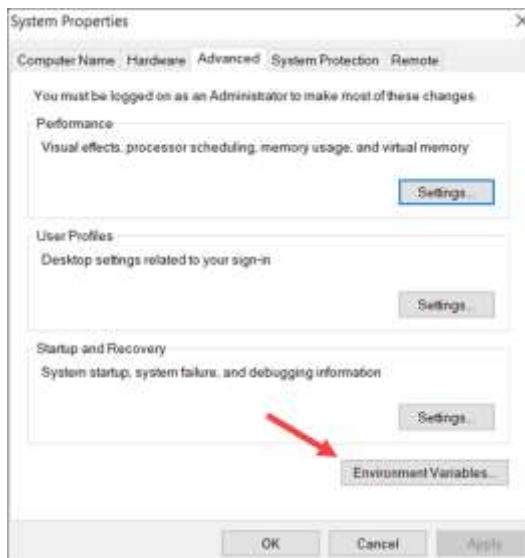
1. Navigate to This PC > Properties.



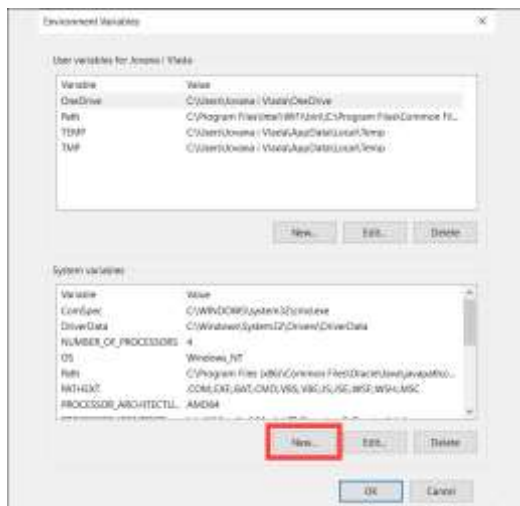
2. Select Advanced system settings.



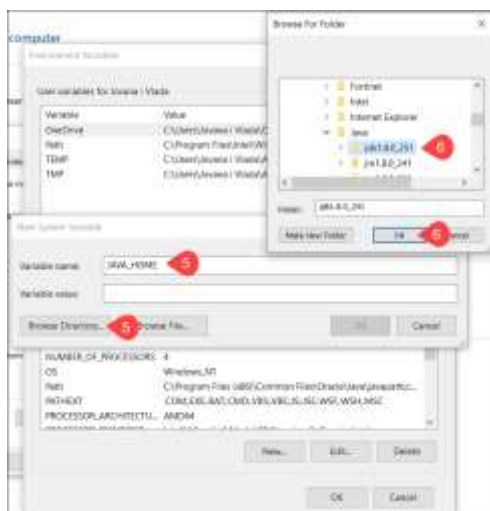
3. Click the Environment Variables... button.



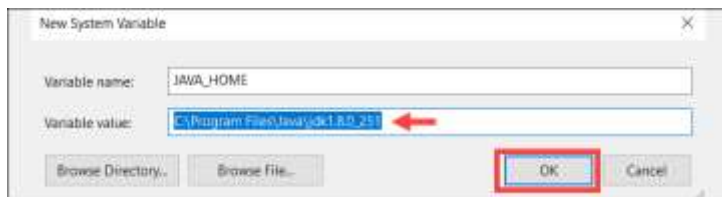
4. Select New in the System Variable section.



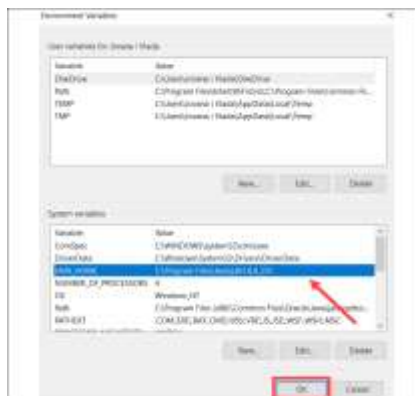
5. Enter JAVA_HOME for the new variable name. Select the Variable value field and then the Browse Directory option.



6. Navigate to This PC > Local Disk C: > Program Files > Java > jdk1.8.0_251 and select OK.
7. Once the correct path to the JDK 8 installation folder has been added to the JAVA_HOME system variable, click OK.



8. You have successfully added the JAVA_HOME system variable with the correct JDK 8 path to the variable list. Select OK in the main Environment Variables window to complete the process.



Step 2: Install and Configure Python 2.7 on Windows

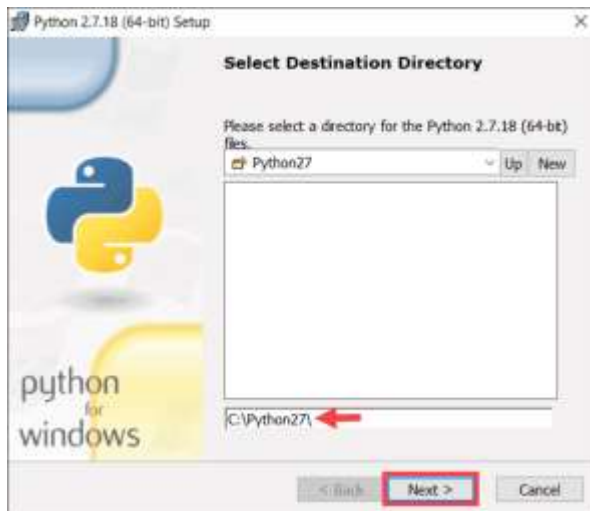
Users interact with the Cassandra database by utilizing the cqlsh bash shell. You need to install Python 2.7 for cqlsh to handle user requests properly.

Install Python 2.7 on Windows

1. Visit the Python official download page and select the Windows x64 version link.



2. Define if you would like Python to be available to all users on this machine or just for your user account and select Next.
3. Specify and take note of the Python installation folder location. Feel free to leave the default location C:\Python27 by clicking Next.



4. The following step allows you to customize the Python installation package. Select Next to continue the installation using the default settings.

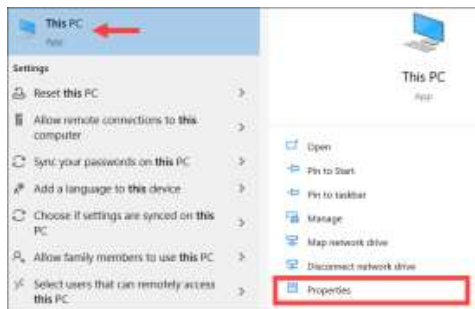


5. The installation process takes a few moments. Once it is complete, select Finish to conclude the installation process.

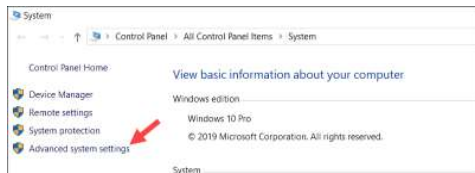


Edit Environment Variable for Python 2.7

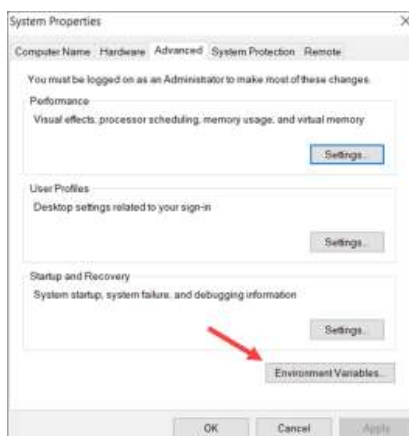
1. Navigate to This PC > Properties



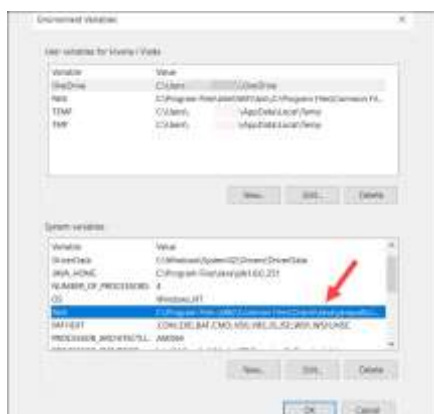
2. Select the Advanced system settings option.



3. Click Environment Variable.



4. Double-click on the existing Path system variable.



5. Select New and then Browse to locate the Python installation folder quickly. Once you have confirmed that the path is correct, click OK.



6. Add the Python 2.7 path to the Path system variable by selecting OK.



Step 3: Download and Set Up Apache Cassandra

Download and Extract Cassandra tar.gz Folder

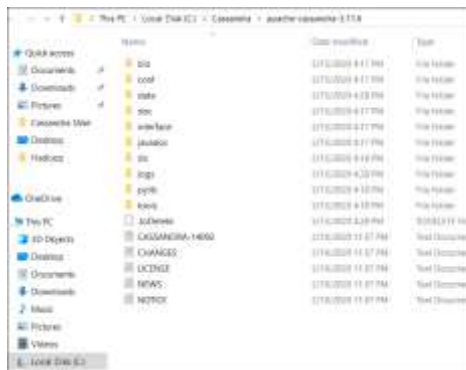
1. Visit the official Apache Cassandra Download page and select the version you would prefer to download. Currently, the latest available version is 3.11.6.



2. Click the suggested Mirror download link to start the download process.



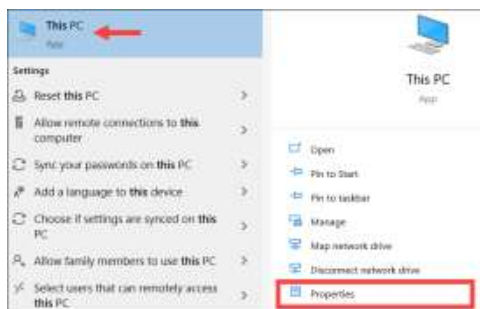
3. Unzip the compressed tar.gz folder using a compression tool such as 7-Zip or WinZip. In this example, the compressed folder was unzipped, and the content placed in the C:\Cassandra\apache-cassandra-3.11.6 folder.



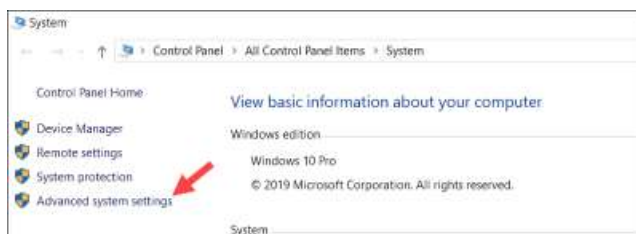
Configure Environment Variables for Cassandra

Set up the environment variables for Cassandra to enable the database to interact with other applications and operate on Windows.

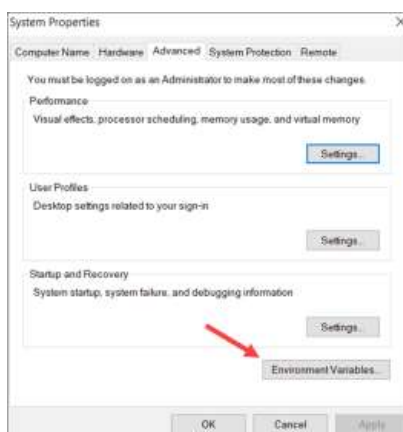
1. Go to This PC > Properties.



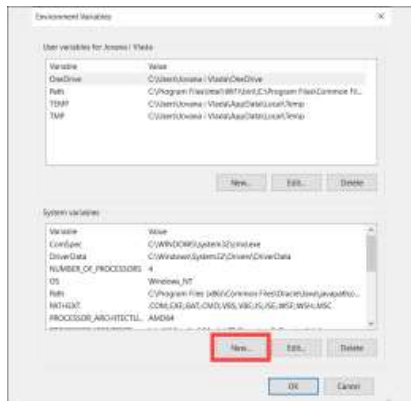
2. Go to Advanced system settings.



3. Click the Environment Variables... button.



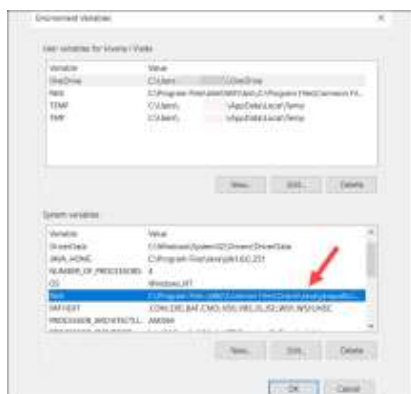
4. Add a completely new entry by selecting the New option.



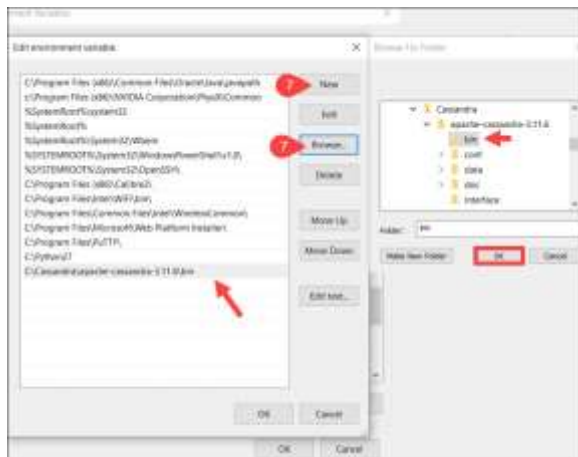
5. Type CASSANDRA_HOME for Variable name, then for the Variable value column select the location of the unzipped Apache Cassandra folder. Based on the previous steps, the location is C:\Cassandraapache-cassandra-3.11.



6. Once you have confirmed that the location is correct, click OK.



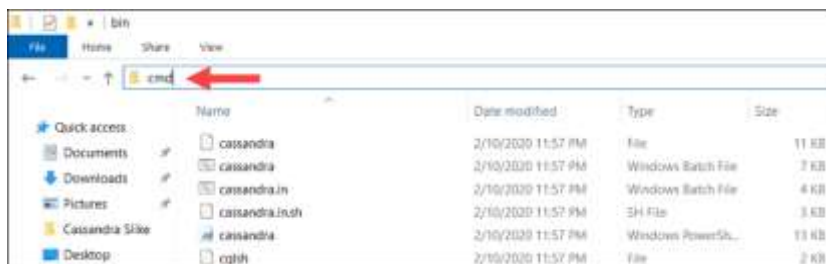
7. Select New and then Browse. In this instance, you need to add the full path to the bin folder located within the Apache Cassandra folder, C:\Cassandraapache-cassandra-3.11.6bin.



8. Hit the OK button and then again OK to save the edited variables.

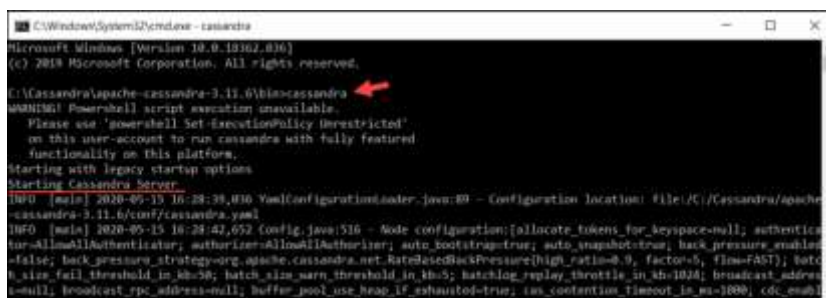
Step 4: Start Cassandra from Windows CMD

Navigate to the Cassandra bin folder. Start the Windows Command Prompt directly from within the bin folder by typing `cmd` in the address bar and pressing Enter.



Type the following command to start the Cassandra server:

Cassandra The system proceeds to start the Cassandra Server.



Step 5: Access Cassandra cqlsh from Windows CMD

While the initial command prompt is still running open a new command line prompt from the same bin folder. Enter the following command to access the Cassandra cqlsh bash shell:

Cqlsh

```

C:\Windows\System32\cmd.exe - cqlsh
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Cassandra\apache-cassandra-3.11.6\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 3.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>

```

You now have access to the Cassandra shell and can proceed to issue basic database commands to your Cassandra server.

- b. Create a keyspace and define a table schema.

Open Ubuntu



Open terminal and type cqlsh

```

hp@ubuntu: ~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.5 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh>

```

```

hp@ubuntu:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 4.0.0 | Cassandra 4.0.5 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> create keyspace bilal with replication={'class': 'SimpleStrategy', 'replica
tion_factor': 3};

Warnings :
Your replication factor 3 for keyspace bilal is higher than the number of nodes
1

cqlsh> use bilal;
cqlsh:bilal> describe bilal;

CREATE KEYSPACE bilal WITH replication = {'class': 'SimpleStrategy', 'replicatio
n_factor': '3'} AND durable_writes = true;
cqlsh:bilal>

```

- c. Insert data into the table.
- c. Perform a CRUD operations and query data from Apache Cassandra.

```

cqlsh:bilal> create table students(rollno int primary key, name text, age int,
house text);
cqlsh:bilal> describe students;

CREATE TABLE bilal.students (
  rollno int PRIMARY KEY,
  age int,
  house text,
  name text
) WITH additional_write_policy = '99p'
  AND bloom_filter_fp_chance = 0.01
  AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
  AND cdc = false
  AND comment = ''
  AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredC
ompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
  AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassan
dra.io.compress.LZ4Compressor'}
  AND crc_check_chance = 1.0
  AND default_time_to_live = 0
  AND extensions = {}
  AND gc_grace_seconds = 864000
  AND max_index_interval = 2048
  AND memtable_flush_period_in_ms = 0
  AND min_index_interval = 128
  AND read_repair = 'BLOCKING'
  AND speculative_retry = '99p';
cqlsh:bilal>

```

Cassandra Create Table

Commands:


```

hp@ubuntu: ~
cqlsh:bilal> insert into students (rollno,name,age,house) values(1,'Bilal',24,
'Agni');
cqlsh:bilal> insert into students (rollno,name,age,house) values(2,'Keshav',22
,'Vayu');
cqlsh:bilal> insert into students (rollno,name,age,house) values(3,'Masood',23
,'Jal');
cqlsh:bilal> insert into students (rollno,name,age,house) values(4,'Shubham',2
4,'Prthvi');

```

Commands:

Read

```

hp@ubuntu: ~
cqlsh:bilal> select * from students;

rollno | age | house | name
-----+-----+-----+-----
      1 |  24 |   Agni |  Bilal
      2 |  22 |   Vayu | Keshav
      4 |  24 | Prthvi | Shubham
      3 |  23 |    Jal | Masood

(4 rows)

```

Update the table:

Commands:

```

hp@ubuntu: ~
cqlsh:bilal> update students set age=23 where rollno=4;
cqlsh:bilal> select * from students;

rollno | age | house | name
-----+-----+-----+-----
      1 |  24 |   Agni |  Bilal
      2 |  22 |   Vayu | Keshav
      4 |  23 | Prthvi | Shubham
      3 |  23 |    Jal | Masood

(4 rows)
cqlsh:bilal>

```

Delete

Commands:


```

hp@ubuntu: ~
cqlsh:bilal> delete from students where rollno=2;
cqlsh:bilal> select * from students;

rollno | age | house | name
-----+-----+-----+-----
1 | 24 | Agni | Bilal
4 | 23 | Prthvi | Shubham
3 | 23 | Jal | Masood

(3 rows)
cqlsh:bilal>

```

Drop table

Commands:

```

hp@ubuntu: ~
cqlsh:bilal> drop table students;
cqlsh:bilal> describe students;
'students' not found in keyspace 'bilal'
cqlsh:bilal>

```

Drop Keyspace

Commands:

```

hp@ubuntu: ~
cqlsh:bilal> drop keyspace bilal;
cqlsh:bilal> drop keyspace bilal;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Keyspace
'bilal' doesn't exist"
cqlsh:bilal>

```

Practical No. 5A: Querying MongoDB

Aim: Querying MongoDB

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

❖ **Checking all Existing Data**

❖ Finding data in collection using " find().pretty() " Command.

```
masood> db.Student.find().pretty()
[
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com',
    Phone: '9876543214'
  },
  {
    _id: ObjectId("670102049e0f9bf8935683a5"),
    'roll no': 5,
    name: 'Bilal',
    email: 'bilal@email.com'
  },
  {
    _id: ObjectId("6701024f9e0f9bf8935683a7"),
    'roll no': 7,
    name: 'Rohit',
    email: 'rohit@email.com'
  },
  {
    _id: ObjectId("670107e69e0f9bf8935683a8"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  },
  {
    _id: ObjectId("6701080f9e0f9bf8935683a9"),
    'roll no': 4,
    name: 'Zaid',
    email: 'zaid@email.com'
  },
]
```

❖ Finding Specific Data

```
masood> db.Student.find({"roll no":3})
[
  {
    _id: ObjectId("670100a99e0f9bf8935683a3"),
    'roll no': 3,
    name: 'Ruhaan',
    email: 'ruhaan@email.com',
    Phone: '9876543214'
  }
]
masood> db.Student.find({"name":"a"})

masood> db.Student.find({"roll no":5})
[
  {
    _id: ObjectId("670102049e0f9bf8935683a5"),
    'roll no': 5,
    name: 'Bilal',
    email: 'bilal@email.com'
  }
]
masood> db.Student.find({"roll no":1})
[
  {
    _id: ObjectId("670107e69e0f9bf8935683a8"),
    'roll no': 1,
    name: 'Masood',
    email: 'masood@email.com'
  }
]
```

❖ Finding Specific Data using \$in

```
masood> db.Student.find({"roll no":{"$in":[4,6,8]}})
[
  {
    _id: ObjectId("6701080f9e0f9bf8935683a9"),
    'roll no': 4,
    name: 'Zaid',
    email: 'zaid@email.com'
  },
  {
    _id: ObjectId("670108269e0f9bf8935683aa"),
    'roll no': 6,
    name: 'Sana',
    email: 'sana@email.com'
  },
  {
    _id: ObjectId("6701083d9e0f9bf8935683ab"),
    'roll no': 8,
    name: 'Arhan',
    email: 'arhan@email.com'
  }
]
```

❖ **Finding Data using \$or**

❖ db.Student.find({\$or:[{"roll no":{"gte":3}},{"name":"Zaid"}]})

```
masood> db.Student.find({$or:[{"roll no":{"gte":3}},{"name":"Zaid"}]})
[
  {
    _id: ObjectId("6701080f9e0f9bf8935683a9"),
    'roll no': 4,
    name: 'Zaid',
    email: 'zaid@email.com'
  }
]
```

❖ **Finding Specific Data using \$eq which is EQUAL**

❖ db.Student.find({\$or:[{"roll no":2},{"name":"Zaid"}]})

```
masood> db.Student.find({$or:[{"roll no":2},{"name":"Zaid"}]})
[
  {
    _id: ObjectId("670100979e0f9bf8935683a2"),
    'roll no': 2,
    name: 'Asad',
    email: 'asad@email.com'
  },
  {
    _id: ObjectId("6701080f9e0f9bf8935683a9"),
    'roll no': 4,
    name: 'Zaid',
    email: 'zaid@email.com'
  }
]
```

Practical No. 6: Redis Data Manipulation

AIM: Redis Data Manipulation

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

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ping "Hello Masood"
"Hello Masood"
127.0.0.1:6379>
```

```
127.0.0.1:6379> set Masood NewGen
OK
127.0.0.1:6379> get Masood
"NewGen"
```

```
127.0.0.1:6379> setex Masood 10 NewGen2
OK
127.0.0.1:6379> get Masood
"NewGen2"
127.0.0.1:6379> get Masood
"NewGen2"
127.0.0.1:6379> get Masood
(nil)
```

```
127.0.0.1:6379> keys *
1) "compay_name1"
2) "b_name"
3) "company_name"
4) "brand_name"
5) "Masood"
6) "company"
```

```
127.0.0.1:6379> del b_name
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
```

```

127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> get head
(nil)
127.0.0.1:6379> set head ajhgfdh
OK
127.0.0.1:6379> get head
"ajhgfdh"
127.0.0.1:6379> ttl head
(integer) -1
127.0.0.1:6379> expire head 10
(integer) 1
127.0.0.1:6379> ttl head
(integer) 7
127.0.0.1:6379> ttl head
(integer) 4
127.0.0.1:6379> ttl head
(integer) 0

```

```

127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone 9988774455
OK

```

```

127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"

```

```

127.0.0.1:6379> expire phone 10
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"
127.0.0.1:6379>
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"

```

```
127.0.0.1:6379> set add mum EX 5
OK
127.0.0.1:6379> get add
"mum"
127.0.0.1:6379> get add
(nil)
```

```
127.0.0.1:6379> set add mum EX 5
OK
127.0.0.1:6379> get add
"mum"
127.0.0.1:6379> get add
(nil)
127.0.0.1:6379> append Masood Project2024
(integer) 19
127.0.0.1:6379> get Masood
"NewGen45Project2024"
```

```
127.0.0.1:6379> append Masood website
(integer) 26
127.0.0.1:6379> get Masood
"NewGen45Project2024website"
```

```
127.0.0.1:6379> strlen Masood
(integer) 26
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
```

```
127.0.0.1:6379> del compay_name1
(integer) 0
127.0.0.1:6379> del Masood
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "company"
127.0.0.1:6379> del compay_name1
(integer) 1
127.0.0.1:6379> keys *
1) "company_name"
2) "brand_name"
3) "company"
```

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

```
127.0.0.1:6379> LPUSH Masood nextGentime
(integer) 1
127.0.0.1:6379> LPUSH Masood qwerty
(integer) 2
127.0.0.1:6379> LPUSH Masood asdfgh
(integer) 3
127.0.0.1:6379> LPUSH Masood zxcvbnm
(integer) 4
```

```
127.0.0.1:6379> LRANGE Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "qwerty"
4) "nextGentime"
```

```
127.0.0.1:6379> RPUSH Masood Bilal
(integer) 5
127.0.0.1:6379> LRANGE Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "qwerty"
4) "nextGentime"
5) "Bilal"
```

```
127.0.0.1:6379> LINDEX Masood 2
"qwerty"
127.0.0.1:6379> LINDEX Masood 4
"Bilal"
127.0.0.1:6379> LINDEX Masood 3
"nextGentime"
```

```
127.0.0.1:6379> LSET Masood 2 Khan
OK
127.0.0.1:6379> LRANGE Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "Khan"
4) "nextGentime"
5) "Bilal"
```



```
127.0.0.1:6379> RPOP Masood
"Bilal"
127.0.0.1:6379> LRange Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "Khan"
4) "nextGentime"
```

```
127.0.0.1:6379> LPOP Masood
"zxcvbnm"
127.0.0.1:6379> LRange Masood 0 10
1) "asdfgh"
2) "Khan"
3) "nextGentime"
```

```
127.0.0.1:6379> BLPOP Masood 10
1) "Masood"
2) "asdfgh"
127.0.0.1:6379> LRange Masood 0 10
1) "Khan"
2) "nextGentime"
127.0.0.1:6379> LRange Masood 0 10
1) "Khan"
2) "nextGentime"
```

```
127.0.0.1:6379> LPUSH Masood Iphone Samsung
(integer) 4
127.0.0.1:6379> LRange Masood 0 10
1) "Samsung"
2) "Iphone"
3) "Khan"
4) "nextGentime"
```

```
127.0.0.1:6379> set BiluBhai keshubhai
OK
127.0.0.1:6379> get BiluBhai
"keshubhai"
127.0.0.1:6379> set prathuBhai 10 vivekubhai
(error) ERR syntax error
127.0.0.1:6379> setex prathuBhai 10 vivekubhai
OK
127.0.0.1:6379> get prathuBhai
(nil)
```

```
127.0.0.1:6379> Keys *
1) "BiluBhai"
2) "Masood"
127.0.0.1:6379> |
```

```

127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> get head
(nil)
127.0.0.1:6379> set head ajhgfdh
OK
127.0.0.1:6379> get head
"ajhgfdh"
127.0.0.1:6379> ttl head
(integer) -1
127.0.0.1:6379> expire head 10
(integer) 1
127.0.0.1:6379> ttl head
(integer) 7
127.0.0.1:6379> ttl head
(integer) 4
127.0.0.1:6379> ttl head
(integer) 0

```

```

127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone 9988774455
OK

```

```

127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"

```

SADD - Adds one or more members to a set

127.0.0.1:6379> SADD company_name bluebirch zentaix biretail

(integer) 3

SMEMBERS - Gets all the members in a set

127.0.0.1:6379> SMEMBERS company_name

1) "biretail";

2) "zentaix";

3) "bluebirch";

SCARD - Gets the number of members in a set

127.0.0.1:6379> SCARD company_name

(integer) 3

127.0.0.1:6379> SADD brand_name posiflex

(integer) 1

127.0.0.1:6379> SADD brand_name dinglix

(integer) 1

127.0.0.1:6379> SADD brand_name spacex

(integer) 1

127.0.0.1:6379> SADD brand_name valience

(integer) 1

127.0.0.1:6379> SMEMBERS brand_name

1) "valience"

2) "dinglix"

3) "spacex"

4) "posiflex"

127.0.0.1:6379> SCARD brand_name

(integer) 4

127.0.0.1:6379> SCARD brand_name key

(error) ERR wrong number of arguments for 'scard' command

127.0.0.1:6379> SCARD brand_name

(integer) 4

SISMEMBER - Determines if a given value is a member of a set

127.0.0.1:6379> SISMEMBER brand_name spacex

(integer) 1

SREM - Removes one or more members from a set

127.0.0.1:6379> SREM spacex

Practical No. 7: Implementing Indexing in MongoDB

Aim: Implementing Indexing in MongoDB

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

A: Create an index on a specific field in a MongoDB collection

- ❖ **First check Before Creating Index**
- ❖ `db.students.getIndexes()`

```
masood> db.Student.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
masood> |
```

- ❖ **Creating Index**
- ❖ `db.staff.createIndex()`

```
masood> db.Student.createIndex({"roll no":3})
roll no_3
masood> db.Student.createIndex({"roll no":1})
roll no_1
masood> db.Student.createIndex({"roll no":5})
roll no_5
```

- ❖ **Get the Created Indexes**
- ❖ `db.staff.getIndexes()`

```
masood> db.Student.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { 'roll no': 3 }, name: 'roll no_3' },
  { v: 2, key: { 'roll no': 1 }, name: 'roll no_1' },
  { v: 2, key: { 'roll no': 5 }, name: 'roll no_5' }
]
```

- ❖ **Dropping or Deleting Indexe's**
- ❖ `db.staff.dropIndex()`

```
masood> db.Student.dropIndex({"roll no":5})
{ nIndexesWas: 4, ok: 1 }
masood> db.Student.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { 'roll no': 3 }, name: 'roll no_3' },
  { v: 2, key: { 'roll no': 1 }, name: 'roll no_1' }
]
```

B: Measure the impact of indexing on query performance

- **Before Creating Index Checking (IT will scan from all documents to check)**
- `db.staff.find({}).explain("executionStats")`

```
masood> db.Student.find({"email":"zaid@email.com"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'masood.Student',
    indexFilterSet: false,
    parsedQuery: { email: { '$eq': 'zaid@email.com' } } },
    queryHash: 'CFBD0524',
    planCacheKey: 'CFBD0524',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { email: { '$eq': 'zaid@email.com' } } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 16,
    totalKeysExamined: 0,
    totalDocsExamined: 8,
    executionStages: {
      stage: 'COLLSCAN',
      filter: { email: { '$eq': 'zaid@email.com' } } },
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 9,
      advanced: 1,
      needTime: 7,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 8
    }
  }
}
```

```

},
command: {
  find: 'Student',
  filter: { email: 'zaid@email.com' },
  '$db': 'masood'
},
serverInfo: {
  host: 'DESKTOP-HD6N8CK',
  port: 27017,
  version: '7.0.5',
  gitVersion: '7809d71e84e314b497f282ea8aa06d7ded3eb205'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted'
},
ok: 1
}

```

- **After Creating Index Checking (It will rapidly show the result by scanning specific Document)**
- `db.staff.find({}).explain("executionStats")`

```

masood> db.Student.find({"roll no":3}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'masood.Student',
    indexFilterSet: false,
    parsedQuery: { 'roll no': { '$eq': 3 } },
    queryHash: 'C4D6C4FF',
    planCacheKey: 'B1A044A9',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { 'roll no': 3 },
        indexName: 'roll no_3',
        isMultiKey: false,
        multiKeyPaths: { 'roll no': [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { 'roll no': [ '[3, 3]' ] }
      }
    },
    rejectedPlans: [
      {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { 'roll no': 1 },
          indexName: 'roll no_1',
          isMultiKey: false,
          multiKeyPaths: { 'roll no': [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,

```

```

        indexVersion: 2,
        direction: 'forward',
        indexBounds: { 'roll no': [ '[3, 3]' ] }
    }
}
]
},
executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 67,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
        stage: 'FETCH',
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        works: 3,
        advanced: 1,
        needTime: 0,
        needYield: 0,
        saveState: 1,
        restoreState: 1,
        isEOF: 1,
        docsExamined: 1,
        alreadyHasObj: 0,
        inputStage: {
            stage: 'IXSCAN',
            nReturned: 1,
            executionTimeMillisEstimate: 0,
            works: 2,
            advanced: 1,
            needTime: 0,
            needYield: 0,
            saveState: 1,
            restoreState: 1,
            isEOF: 1,
            keyPattern: { 'roll no': 3 },
            indexName: 'roll no_3',
            isMultiKey: false,

```



```

    multiKeyPaths: { 'roll no': [] },
    isUnique: false,
    isSparse: false,
    isPartial: false,
    indexVersion: 2,
    direction: 'forward',
    indexBounds: { 'roll no': [ '[3, 3]' ] },
    keysExamined: 1,
    seeks: 1,
    dupsTested: 0,
    dupsDropped: 0
  }
},
command: { find: 'Student', filter: { 'roll no': 3 }, '$db': 'masood' },
serverInfo: {
  host: 'DESKTOP-HD6N8CK',
  port: 27017,
  version: '7.0.5',
  gitVersion: '7809d71e84e314b497f282ea8aa06d7ded3eb205'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted'
},
ok: 1
}

```

Practical No. 8: Data Storage in Redis

AIM: Data Storage in Redis

- a) Implement caching functionality using Redis as a cache store.
- b) Store and retrieve data from Redis cache using appropriate commands.

GET & SET For adding and view the Inserted Data

Redis – Strings

```
127.0.0.1:6379> set company_name nextgenpixel
```

OK

```
127.0.0.1:6379> get company_name
```

"nextgenpixel"

```
127.0.0.1:6379> setex brand_name 10 pixel
```

OK

```
127.0.0.1:6379> get brand_name
```

"pixel"

```
127.0.0.1:6379> get brand_name
```

(nil)

```
127.0.0.1:6379> set company_id 22
```

OK

```
127.0.0.1:6379> keys *
```

4) "company_id"

5) "company_name"

6) "name"

del to delete the Data

```
127.0.0.1:6379> del name
```

(integer) 1

```
127.0.0.1:6379> keys *
```

3) "company_id"

4) "company_name"

TTL stands for Time to Live. In Redis, it's a way to set an expiry time on a key

```
127.0.0.1:6379> ttl head
```

```
(integer) -2
```

```
127.0.0.1:6379> expire head 10
```

```
(integer) 0
```

```
127.0.0.1:6379> set phone_no 92959252952
```

```
OK
```

keys* to show all keys

```
127.0.0.1:6379> keys *
```

```
4) "phone_no"
```

```
5) "company_id"
```

```
6) "company_name"
```

```
127.0.0.1:6379> expire phone_no 10
```

```
(integer) 1
```

```
127.0.0.1:6379> keys *
```

```
4) "phone_no"
```

```
5) "company_id"
```

```
6) "company_name"
```

```
127.0.0.1:6379> keys *
```

```
3) "company_id"
```

```
4) "company_name"
```

Set the with expiration time

```
127.0.0.1:6379> set address mumbai EX 5
```

```
OK
```

```
127.0.0.1:6379> get address
```

```
(nil)
```

```
127.0.0.1:6379> append company_name .co.in
```

```
(integer) 18
```

```
127.0.0.1:6379> get company_name
```

```
"nextgenpixel.co.in"
```

Update the Data

```
127.0.0.1:6379> append company_name website
```

```
(integer) 25
```

```
127.0.0.1:6379> get company_name
```

```
"nextgenpixel.co.inwebsite"
```

Shows the length

```
127.0.0.1:6379> strlen company_name
```

```
(integer) 25
```

```
127.0.0.1:6379> keys *
```

```
3) "company_id"
```

```
4) "company_name"
```

Delete the Key

```
127.0.0.1:6379> del company_id
```

```
(integer) 1
```

```
127.0.0.1:6379> keys *
```

```
1) "company_name"
```

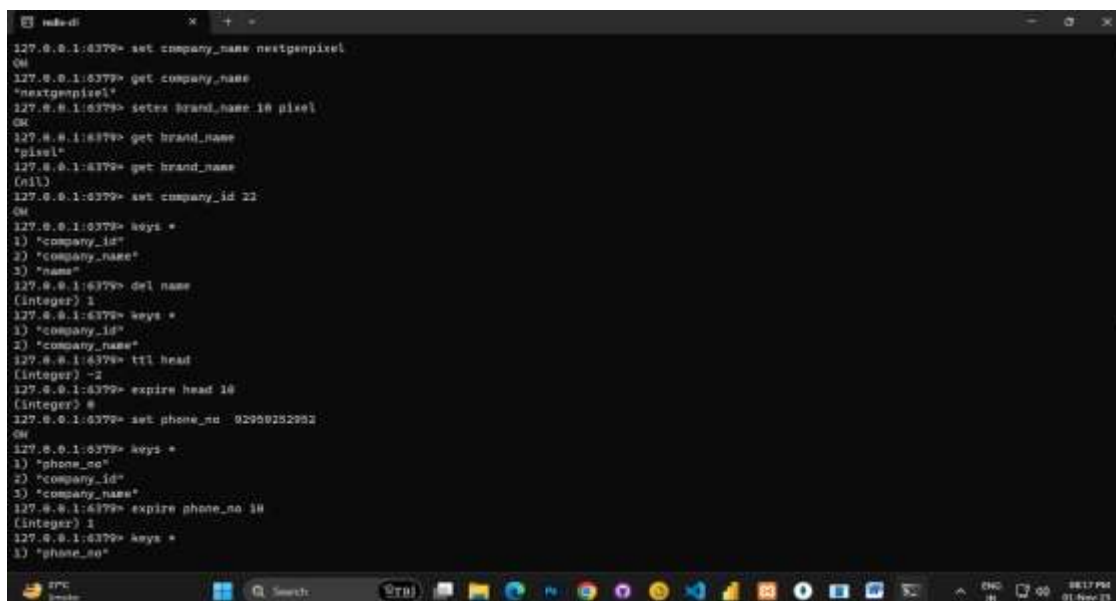
Delete all Existing Keys

```
127.0.0.1:6379> flushall
```

```
OK
```

```
127.0.0.1:6379> keys *
```

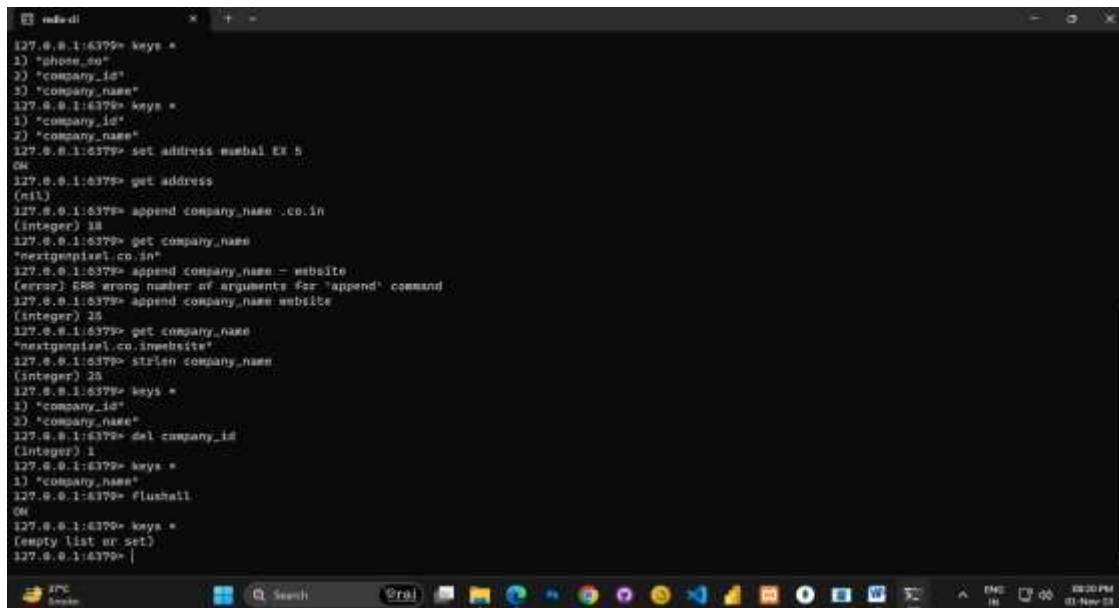
```
(empty list or set)
```



```

127.0.0.1:6379> set company_name nextgenpixel
OK
127.0.0.1:6379> get company_name
"nextgenpixel"
127.0.0.1:6379> setex brand_name 10 pixel
OK
127.0.0.1:6379> get brand_name
"pixel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 22
OK
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone_no 92900252202
OK
127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> expire phone_no 10
(integer) 1
127.0.0.1:6379> keys *
1) "phone_no"

```



```

127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> set address mumbai Ex 5
OK
127.0.0.1:6379> get address
(nil)
127.0.0.1:6379> append company_name .co.in
(integer) 18
127.0.0.1:6379> get company_name
"nextgenpixel.co.in"
127.0.0.1:6379> append company_name - website
(error) ERR wrong number of arguments for 'append' command
127.0.0.1:6379> append company_name website
(integer) 25
127.0.0.1:6379> get company_name
"nextgenpixel.co.inwebsite"
127.0.0.1:6379> strlen company_name
(integer) 25
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> del company_id
(integer) 1
127.0.0.1:6379> keys *
1) "company_name"
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>

```

Redis for LIST

Redis – Lists Redis Lists are simply lists of strings, sorted by insertion order. You can add elements in Redis lists in the head or the tail of the list. Maximum length of a list is 232 - 1 elements (4294967295, more than 4 billion of elements per list).

LPUSH - Prepends one or multiple values to a list

```
127.0.0.1:6379> LPUSH compay_name nextgenpixel
```

```
(integer) 1
```

```
127.0.0.1:6379> LPUSH compay_name adobe
```

```
(integer) 2
```

```
127.0.0.1:6379> LPUSH compay_name frolikpixel
```

```
(integer) 3
```

LRANGE – Shows data from the specific

```
127.0.0.1:6379> LRANGE company_name 0 10
```

```
1) "nextgenpixel,adobe,amazon"
```

```
127.0.0.1:6379> LPUSH brand_name manyavar centric xbox
```

```
(integer) 3
```

```
127.0.0.1:6379> LRANGE brand_name 0 10
```

```
1) "xbox"
```

```
4) "centric"
```

5) "manyavar"

RPUSH – (Updates) Appends one or multiple values to a list

127.0.0.1:6379> RPUSH brand_name web_engage

(integer) 4

127.0.0.1:6379> LRange brand_name 0 10

5) "xbox"

6) "centric"

7) "manyavar"

8) "web_engage"

127.0.0.1:6379> LINDEX brand_name 2

"manyavar"

LSET – Sets the value of an element in a list by its index

127.0.0.1:6379> LSET brand_name 1 xbox360

OK

LRANGE - Gets a range of elements from a list

127.0.0.1:6379> LRange brand_name 0 10

5) "xbox"

6) "xbox360"

7) "manyavar"

8) "web_engage"

LPOP - Removes and gets the first element in a list

127.0.0.1:6379> LPOP brand_name

"xbox"

127.0.0.1:6379> LRange brand_name 0 10

4) "xbox360"

5) "manyavar"

6) "web_engage"

127.0.0.1:6379> RPOP brand_name

"web_engage"

127.0.0.1:6379> LRange brand_name 0 10

1) "xbox360"

2) "manyavar"

BLPOP - Removes and gets the first element in a list, or blocks until one is available

127.0.0.1:6379> BLPOP brand_name 5

3) "brand_name"

4) "xbox360"

127.0.0.1:6379> LRANGE brand_name 0 10

1) "manyavar"

127.0.0.1:6379> LRANGE brand_name 0 10

1) "manyavar"

127.0.0.1:6379> LPUSH brand_name blubirch samsung

(integer) 3

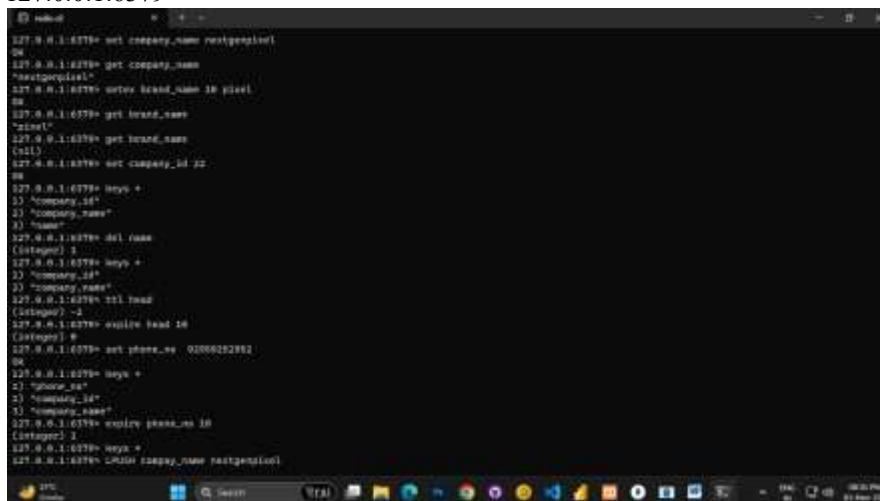
127.0.0.1:6379> LRANGE brand_name 0 10

4) "samsung"

5) "blubirch"

6) "manyavar"

127.0.0.1:6379>



```

127.0.0.1:6379> set company_name nextgenpixel
OK
127.0.0.1:6379> get company_name
"nextgenpixel"
127.0.0.1:6379> setex brand_name 10 pixel
OK
127.0.0.1:6379> get brand_name
"pixel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 22
OK
127.0.0.1:6379> keys +
1) "company_id"
2) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys +
1) "company_id"
2) "company_name"
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone_no 010000252952
OK
127.0.0.1:6379> keys +
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> expire pass_no 10
(integer) 1
127.0.0.1:6379> keys +
127.0.0.1:6379> lpush company_name nextgenpixel

```

[illegible]

REDIS -SETs

Redis – Sets

Redis Sets are an unordered collection of unique strings. Unique means sets does not allow repetition of data in a key. In Redis set add, remove, and test for the existence of members in $O(1)$ (constant time regardless of the number of elements contained inside the Set). The maximum length of a list is $2^{32} - 1$ elements (4294967295, more than 4 billion of elements per set).

```
127.0.0.1:6379> set company name nextgenpixel
```

OK

```
127.0.0.1:6379> get company name
```

"nextgenpixel"

```
127.0.0.1:6379> setex brand name 10 pixel
```

OK

```
127.0.0.1:6379> get brand name
```


"pixel"

127.0.0.1:6379> get brand_name

(nil)

127.0.0.1:6379> set company_id 22

OK

127.0.0.1:6379> keys *

4) "company_id"

5) "company_name"

6) "name"

127.0.0.1:6379> del name

(integer) 1

127.0.0.1:6379> keys *

3) "company_id"

4) "company_name"

127.0.0.1:6379> ttl head

(integer) -2

127.0.0.1:6379> expire head 10

(integer) 0

127.0.0.1:6379> set phone_no 92959252952

OK

127.0.0.1:6379> keys *

4) "phone_no"

5) "company_id"

6) "company_name"

127.0.0.1:6379> expire phone_no 10

(integer) 1

127.0.0.1:6379> keys *

SADD - Adds one or more members to a set

127.0.0.1:6379> SADD company_name bluebirch zentaix biretail

(integer) 3

SMEMBERS - Gets all the members in a set

```
127.0.0.1:6379> SMEMBERS company_name
```

```
4) "biretail"
```

```
5) "zentaix"
```

```
6) "bluebirch"
```

SCARD - Gets the number of members in a set

```
127.0.0.1:6379> SCARD company_name
```

```
(integer) 3
```

```
127.0.0.1:6379> SADD brand_name posiflex
```

```
(integer) 1
```

```
127.0.0.1:6379> SADD brand_name dinglix
```

```
(integer) 1
```

```
127.0.0.1:6379> SADD brand_name spacex
```

```
(integer) 1
```

```
127.0.0.1:6379> SADD brand_name valience
```

```
(integer) 1
```

```
127.0.0.1:6379> SMEMBERS brand_name
```

```
5) "valience"
```

```
6) "dinglix"
```

```
7) "spacex"
```

```
8) "posiflex"
```

```
127.0.0.1:6379> SCARD brand_name
```

```
(integer) 4
```

```
127.0.0.1:6379> SCARD brand_name key
```

```
(error) ERR wrong number of arguments for 'scard' command
```

```
127.0.0.1:6379> SCARD brand_name
```

```
(integer) 4
```

SISMEMBER - Determines if a given value is a member of a set

```
127.0.0.1:6379> SISMEMBER brand_name spacex
```

```
(integer) 1
```

SREM - Removes one or more members from a set

```
127.0.0.1:6379> SREM spacex
```

(error) ERR wrong number of arguments for 'srem' command

```
127.0.0.1:6379> SREM brand_name spacex
```

(integer) 1

SMEMBERS - Gets all the members in a set

```
127.0.0.1:6379> SMEMBERS brand_name
```

4) "valience"

5) "dinglix"

6) "posiflex"

del to remove

```
127.0.0.1:6379> del brand_name
```

(integer) 1

```
127.0.0.1:6379> SMEMBERS brand_name
```

(empty list or set)

```
127.0.0.1:6379>
```

```

107 # W 1:07700 LGH31 company_name whole
108 linktag3 2
109 # W 1:07700 LGH31 company_name fullindex
110 linktag3 3
111 # W 1:07700 LGH31 company_name 0 10
112 # "retrieving_kent_astate_manager"
113 # W 1:07700 LGH31 brand_name motorway kentast whole
114 linktag3 1
115 # W 1:07700 LGH31 brand_name 0 10
116 # "motor"
117 # "kentast"
118 # "motorway"
119 # W 1:07700 WH31 brand_name web_engage
120 linktag3 3
121 # W 1:07700 LGH31 brand_name 0 10
122 # "motor"
123 # "kentast"
124 # "motorway"
125 # W 1:07700 LGH31 brand_name 2
126 # "motorway"
127 # W 1:07700 WH31 company_name blackbird kentast blacktail
128 linktag3 3
129 # W 1:07700 SH31000 company_name
130 # "blacktail"
131 # "blackbird"
132 # "blacktail"
133 # "blackbird"
134 # W 1:07700 SH31000 company_name
135 linktag3 3
136 # W 1:07700 G310 brand_name profile
137 linktag3 2
138 # W 1:07700 G310 brand_name dirgltk
139 linktag3 2
140 # W 1:07700 WH31 brand_name space
141 linktag3 2

```

[illegible]

```

127.0.0.1:6379> set company_name nextgenpixel
OK
127.0.0.1:6379> get company_name
"nextgenpixel"
127.0.0.1:6379> setex brand_name 18 pixel
OK
127.0.0.1:6379> get brand_name
"pixel"
127.0.0.1:6379> get brand_name
(nil)
127.0.0.1:6379> set company_id 22
OK
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
3) "name"
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> keys *
1) "company_id"
2) "company_name"
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 18
(integer) 0
127.0.0.1:6379> set phone_no 92089102902
OK
127.0.0.1:6379> keys *
1) "phone_no"
2) "company_id"
3) "company_name"
127.0.0.1:6379> expire phone_no 18
(integer) 1
127.0.0.1:6379> keys *
127.0.0.1:6379> LPUSH compay_name nextgenpixel

```

```
127.0.0.1:6379> set Masood NewGen
OK
127.0.0.1:6379> get Masood
"NewGen"
```

```
127.0.0.1:6379> setex Masood 10 NewGen2
OK
127.0.0.1:6379> get Masood
"NewGen2"
127.0.0.1:6379> get Masood
"NewGen2"
127.0.0.1:6379> get Masood
(nil)
```

```
127.0.0.1:6379> keys *
1) "compay_name1"
2) "b_name"
3) "company_name"
4) "brand_name"
5) "Masood"
6) "company"
```

```
127.0.0.1:6379> del b_name
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
```

```
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> get head
(nil)
127.0.0.1:6379> set head ajhgfdh
OK
127.0.0.1:6379> get head
"ajhgfdh"
127.0.0.1:6379> ttl head
(integer) -1
127.0.0.1:6379> expire head 10
(integer) 1
127.0.0.1:6379> ttl head
(integer) 7
127.0.0.1:6379> ttl head
(integer) 4
127.0.0.1:6379> ttl head
(integer) 0
```

```
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone 9988774455
OK
```

```
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"
```

```
127.0.0.1:6379> expire phone 10
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"
127.0.0.1:6379>
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
```

```
127.0.0.1:6379> set add mum EX 5
OK
127.0.0.1:6379> get add
"mum"
127.0.0.1:6379> get add
(nil)
```

```
127.0.0.1:6379> set add mum EX 5
OK
127.0.0.1:6379> get add
"mum"
127.0.0.1:6379> get add
(nil)
127.0.0.1:6379> append Masood Project2024
(integer) 19
127.0.0.1:6379> get Masood
"NewGen45Project2024"
```

```

127.0.0.1:6379> append Masood website
(integer) 26
127.0.0.1:6379> get Masood
"NewGen45Project2024website"

```

```

127.0.0.1:6379> strlen Masood
(integer) 26
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"

```

```

127.0.0.1:6379> del company_name1
(integer) 0
127.0.0.1:6379> del Masood
(integer) 1
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "company"
127.0.0.1:6379> del compay_name1
(integer) 1
127.0.0.1:6379> keys *
1) "company_name"
2) "brand_name"
3) "company"

```

```

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

```

```

127.0.0.1:6379> LPUSH Masood nextGentime
(integer) 1
127.0.0.1:6379> LPUSH Masood qwerty
(integer) 2
127.0.0.1:6379> LPUSH Masood asdfgh
(integer) 3
127.0.0.1:6379> LPUSH Masood zxcvbnm
(integer) 4

```

```
127.0.0.1:6379> LRange Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "qwerty"
4) "nextGentime"
```

```
127.0.0.1:6379> RPush Masood Bilal
(integer) 5
127.0.0.1:6379> LRange Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "qwerty"
4) "nextGentime"
5) "Bilal"
```

```
127.0.0.1:6379> LINDEX Masood 2
"qwerty"
127.0.0.1:6379> LINDEX Masood 4
"Bilal"
127.0.0.1:6379> LINDEX Masood 3
"nextGentime"
```

```
127.0.0.1:6379> LSet Masood 2 Khan
OK
127.0.0.1:6379> LRange Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "Khan"
4) "nextGentime"
5) "Bilal"
```

```
127.0.0.1:6379> RPop Masood
"Bilal"
127.0.0.1:6379> LRange Masood 0 10
1) "zxcvbnm"
2) "asdfgh"
3) "Khan"
4) "nextGentime"
```

```
127.0.0.1:6379> LPop Masood
"zxcvbnm"
127.0.0.1:6379> LRange Masood 0 10
1) "asdfgh"
2) "Khan"
3) "nextGentime"
```



```
127.0.0.1:6379> BLPOP Masood 10
1) "Masood"
2) "asdfgh"
127.0.0.1:6379> LRANGE Masood 0 10
1) "Khan"
2) "nextGentime"
127.0.0.1:6379> LRANGE Masood 0 10
1) "Khan"
2) "nextGentime"
```

```
127.0.0.1:6379> LPUSH Masood Iphone Samsung
(integer) 4
127.0.0.1:6379> LRANGE Masood 0 10
1) "Samsung"
2) "Iphone"
3) "Khan"
4) "nextGentime"
```

```
127.0.0.1:6379> set BiluBhai keshubhai
OK
127.0.0.1:6379> get BiluBhai
"keshubhai"
127.0.0.1:6379> set prathuBhai 10 vivekubhai
(error) ERR syntax error
127.0.0.1:6379> setex prathuBhai 10 vivekubhai
OK
127.0.0.1:6379> get prathuBhai
(nil)
```

```
127.0.0.1:6379> Keys *
1) "BiluBhai"
2) "Masood"
127.0.0.1:6379> |
```

```
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> get head
(nil)
127.0.0.1:6379> set head ajhgfdh
OK
127.0.0.1:6379> get head
"ajhgfdh"
127.0.0.1:6379> ttl head
(integer) -1
127.0.0.1:6379> expire head 10
(integer) 1
127.0.0.1:6379> ttl head
(integer) 7
127.0.0.1:6379> ttl head
(integer) 4
127.0.0.1:6379> ttl head
(integer) 0
```

```
127.0.0.1:6379> ttl head
(integer) -2
127.0.0.1:6379> expire head 10
(integer) 0
127.0.0.1:6379> set phone 9988774455
OK
```

```
127.0.0.1:6379> keys *
1) "compay_name1"
2) "company_name"
3) "brand_name"
4) "Masood"
5) "company"
6) "phone"
```