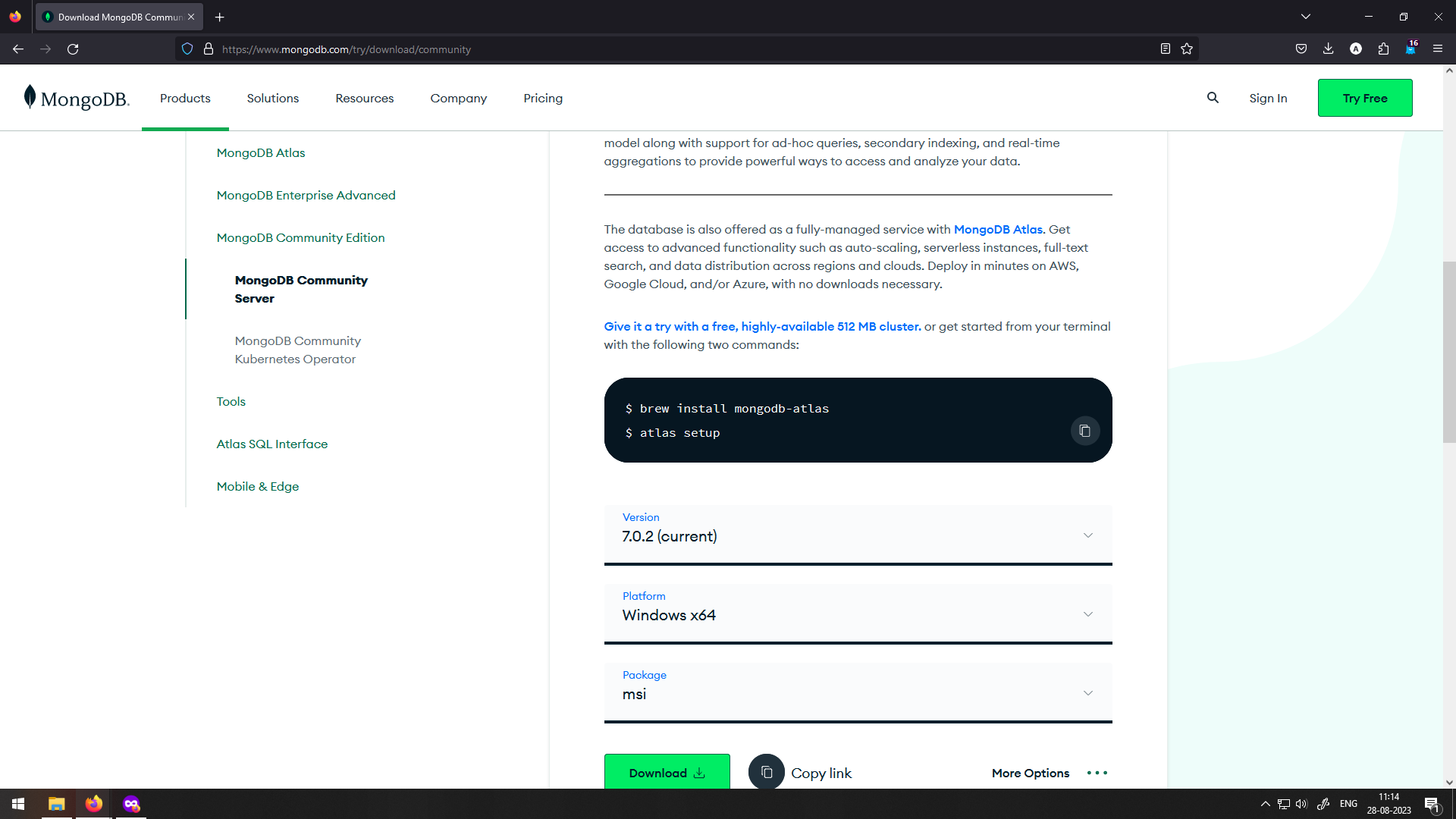
INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr no** | **Practicals** | **Date** | **Signature** |
| **1** | **To Demonstrate Installation Of Mongodb** | **23 – 08 – 2023** |  |
| **2** | **To Perform CRUD Operations on Student Data in MongoDB** | **23 – 08 – 2023** |  |
| **3** | **To demonstrate Aggregation Pipeline in MongoDB.** | **02 – 09 – 2023** |  |
| **4** | **To demonstrate the Usage of MongoDB in Python.** | **02 – 09 – 2023** |  |
| **5** | **To demonstrate Installation of CouchDB.** | **06 – 09 – 2023** |  |
| **6** | **To Perform CRUD Operations in CouchDB.** | **13 – 09 – 2023** |  |
| **7** | **To Create and Utilize Views in CouchDB.** | **17 – 09 – 2023** |  |
| **8** | **HTTP URL Paths and Commands Using CURL to Interact with CouchDB for CRUD Operations.** | **01 – 10 – 2023** |  |
| **9** | **To demonstrate Redis installation and execute its commands to create data** | **07 – 10 – 2023** |  |
| **10** | **To demonstrate cassandra installation and execute queries to create column family data to perform CURD operations** | **13 – 10 – 2023** |  |

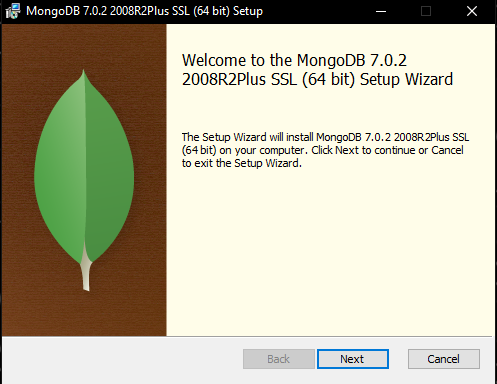
|  |  |
| --- | --- |
| **Practical No. 1** | |
| **Aim :** To Demonstrate Installation of MongoDB | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : 24 - 08 - 2023 | Sign : |

**Procedure :**  Follow these steps to install MongoDB Community Edition

1. Visit the MongoDB download page:<https://www.mongodb.com/try/download/community>
2. Choose the version of MongoDB you want to install.
3. Select the MSI installer package and download it.



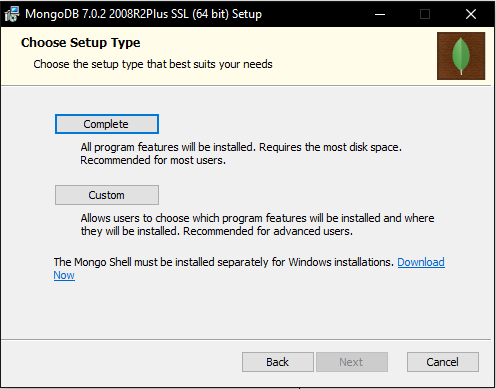
1. Run the installer, pick the desired path and follow the installation wizard.



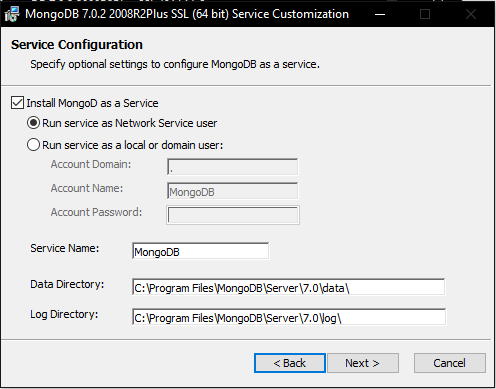
1. Accept the terms in the License Agreement and select Next.



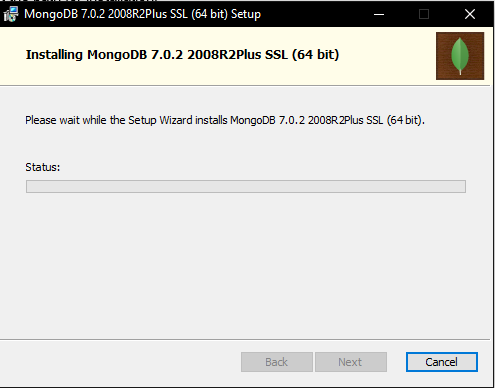
1. Select Complete install.



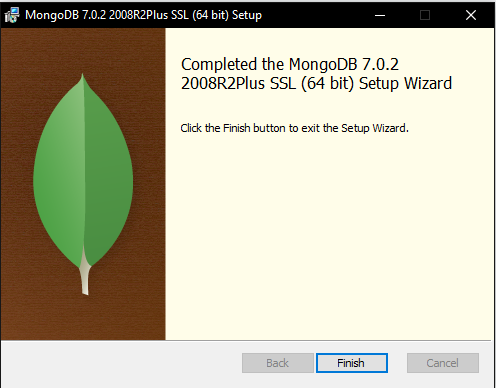
1. Select the path for installation.



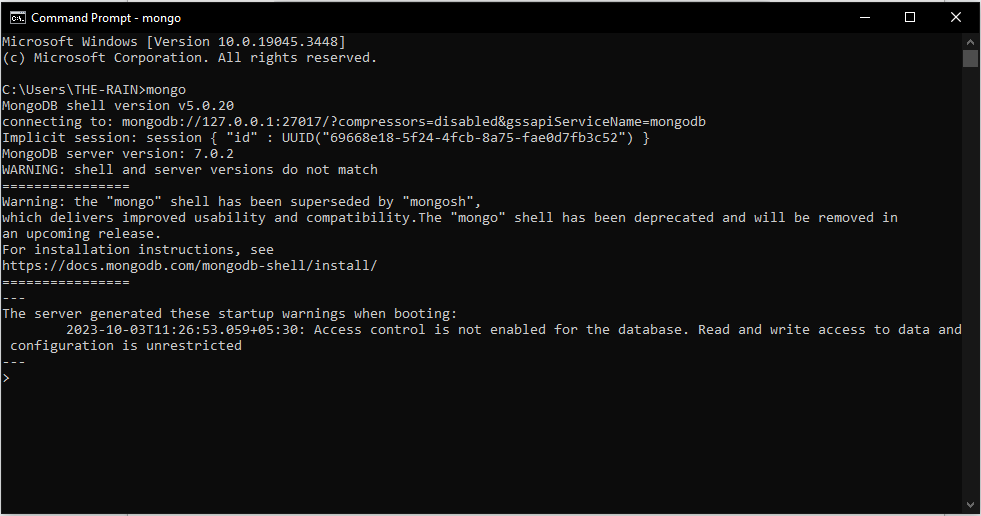
1. Click on next and select install and wait.



1. Click on Finish.



1. Once the installation is completed Open Command Line and type “mongo” or “mongosh” to check if its installed.



1. As a result MongoDB has been successfully installed.

|  |  |
| --- | --- |
| **Practical No. 2** | |
| **Aim :** To Perform CRUD Operations on Student Data in MongoDB | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : 24 - 08 - 2023 | Sign : |

**Procedure :** Performing CRUD (Create, Read, Update, Delete)

### Connect to MongoDB:

**C:\Users\THE-RAIN>mongo**

MongoDB shell version v5.0.20connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("94f23a20-60b1-41e3-958f-04bfde2ad3b1") }

MongoDB server version: 7.0.2

WARNING: shell and server versions do not match

================

Warning: the "mongo" shell has been superseded by "mongosh", which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in an upcoming release. For installation instructions, see

https://docs.mongodb.com/mongodb-shell/install/

================

---

The server generated these startup warnings when booting:

2023-10-03T11:26:53.059+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

### Create a Database:

### > use students;

### switched to db students

### > Db;

### students

### Create (Insert) a Student Record:

### > db.students.insertOne({roll: 01, name: "Calvin",age: 20,marks: 700})

{

"acknowledged" : true,

"insertedId" : ObjectId("651bb8cd73fd2ea5a7f9df13")

}

### 

### > db.students.insertMany([{

### roll: 02, name: 'Karan', age: 21, marks: 750},

### {roll: 03, name: 'Rahul', age: 22, marks:540},

### {roll: 04, name: 'Harsh', age:21, marks: 250},

### {roll: 05, name: 'Ankit', age: 20, marks: 900}]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("651bb96873fd2ea5a7f9df14"),

ObjectId("651bb96873fd2ea5a7f9df15"),

ObjectId("651bb96873fd2ea5a7f9df16"),

ObjectId("651bb96873fd2ea5a7f9df17")

]

}

1. Find Student Record:
   1. Find all documents in the 'Students' collection:

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Find the first document in the 'Students' collection:

**> db.students.findOne();**

{

"\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"),

"roll" : 1,

"name" : "Calvin",

"age" : 20,

"marks" : 700

}

* 1. Find the second document in the 'Students' collection.

**> db.students.find().limit(2);**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

* 1. Display all documents after skipping the first two documents

**> db.students.find().skip(2);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

1. Skip and limit:
   1. displaying them in a pretty format:

**> db.students.find().skip(2).pretty();**

{

"\_id" : ObjectId("651bb96873fd2ea5a7f9df15"),

"roll" : 3,

"name" : "Rahul",

"age" : 22,

"marks" : 540

}

{

"\_id" : ObjectId("651bb96873fd2ea5a7f9df16"),

"roll" : 4,

"name" : "Harsh",

"age" : 21,

"marks" : 250

}

{

"\_id" : ObjectId("651bb96873fd2ea5a7f9df17"),

"roll" : 5,

"name" : "Ankit",

"age" : 20,

"marks" : 900

}

* 1. Skip the first document and limit the result to 1 document.

**> db.students.find().skip(1).limit(1);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

* 1. Skip the first document and limit the result to 2 documents.

**> db.students.find().skip(1).limit(2);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

* 1. Skip the first two documents and limit the result to 2 documents.

**> db.students.find().skip(2).limit(2);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

* 1. Skip the first three documents and limit the result to 1 document.

**> db.students.find().skip(3).limit(1);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

* 1. Skip the first three documents and limit the result to 3 documents."

**> db.students.find().skip(3).limit(3);**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

1. Count: Find the count of documents in the 'students' collection

**> db.students.count();**

5

1. Find all documents in the 'students' collection with:
   1. Roll Number 3:

**> db.students.find({roll: 3});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

* 1. Roll number 4:

**> db.students.find({roll: 4});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

* 1. Roll number 1 or 5:
  2. **> db.students.find({$or:[{roll:{$eq:1}},{roll:{$eq:5}}]});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Roll numbers greater than or equal to 3:

**> db.students.find({roll:{$gte:3}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Roll number 1 and 3:

**> db.students.find({roll:{$in:[1,3]}});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

* 1. Roll numbers other than 1 and 3:

**> db.students.find({roll:{$nin:[1,3]}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Roll numbers other than 4 and 5:

**> db.students.find({$nor:[{roll:{$eq:4}},{roll:{$eq:5}}]});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

1. Find all documents in the 'students' collection with MARKS:
   1. Between 650 and 750 (inclusive).

**> db.students.find({$and:[{marks:{$gte:650}},{marks:{$lte:750}}]});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

* 1. Less than 650?"

**> db.students.find({marks:{$not:{$gte:650}}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

**> db.students.find({marks:{$lte:650}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

* 1. Divisible by 20.

**> db.students.find({marks:{$mod:[20,0]}});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. That exists and is either 600 or 750.

**> db.students.find({marks:{$exists:true,$in:[600,750]}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

* 1. That exist and are either 900 or 1750

**> db.students.find({marks:{$exists:true,$in:[900,1750]}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

1. REGEX: Find all documents in the 'students' collection with names:
   1. Starting with the letter 'A':

**> db.students.find({name:{$regex:/^A/}});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Ending with the letter 'n':

**> db.students.find({name:{$regex:/n$/}});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

* 1. Ending with the letter 'h':

**> db.students.find({name:{$regex:/h$/}});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

1. Sorting: Find all documents in the 'students' collection, sorted by:
   1. Name in ascending order:

**> db.students.find().sort({name:1});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

* 1. Roll number in ascending order:

**> db.students.find().sort({roll:1});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Marks in descending order:

**> db.students.find().sort({marks:-1});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

* 1. Roll number in descending order:

**> db.students.find().sort({roll:-1});**

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Karan", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

1. UPDATE Query :
   1. Update the document with roll number 2 to change the name to 'Varad':

**> db.students.update({roll:2},{$set:{name:"Varad"}});**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Varad", "age" : 21, "marks" : 750 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Update all documents with marks 750 to change their marks to 770:

**> db.students.update({marks:750},{$set:{marks:770}});**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Varad", "age" : 21, "marks" : 770 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 900 }

* 1. Update all documents with marks 900 or 600 to change their marks to 500:

**> db.students.updateMany({marks:{$in:[600,900]}},{$set:{marks:500}});**

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df14"), "roll" : 2, "name" : "Varad", "age" : 21, "marks" : 770 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 500 }

1. Delete Query:
   1. Delete the document with roll number 2:

**> db.students.deleteOne({roll:2});**

{ "acknowledged" : true, "deletedCount" : 1 }

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df17"), "roll" : 5, "name" : "Ankit", "age" : 20, "marks" : 500 }

* 1. Delete all documents with marks 500:

**> db.students.deleteMany({marks:500});**

{ "acknowledged" : true, "deletedCount" : 1 }

**>db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

1. Insert Queries:

**>db.students.insert({roll:2,name:'Ankit',marks:'750',hobbies:['writing','reading','singing']});**

**>db.students.insert({roll:5,name:'Apurva',marks:'700',hobbies:['gaming','sleeping','singing']}); >db.students.insert({roll:6,name:'Sahil',marks:'750',hobbies:['gaming','Gym','dancing']});**

**> db.students.find();**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df16"), "roll" : 4, "name" : "Harsh", "age" : 21, "marks" : 250 }

{ "\_id" : ObjectId("651bcf3973fd2ea5a7f9df18"), "roll" : 2, "name" : "Ankit", "marks" : "750", "hobbies" : [ "writing", "reading", "singing" ] }

{ "\_id" : ObjectId("651bcf4d73fd2ea5a7f9df19"), "roll" : 6, "name" : "Sahil", "marks" : "750", "hobbies" : [ "gaming", "Gym", "dancing" ] }

{ "\_id" : ObjectId("651bcf7873fd2ea5a7f9df1a"), "roll" : 5, "name" : "Apurva", "marks" : "700", "hobbies" : [ "gaming", "sleeping", "singing" ] }

1. Find Queries: Find all documents in the 'students' collection where:
   1. Hobbies include both 'gaming' and 'dancing':

**> db.students.find({hobbies:{$all:['gaming', 'dancing']}});**

{ "\_id" : ObjectId("651bcf4d73fd2ea5a7f9df19"), "roll" : 6, "name" : "Sahil", "marks" : "750", "hobbies" : [ "gaming", "Gym", "dancing" ] }

* 1. Marks are greater than 500:

**> db.students.find({$expr:{$gt:["$marks",500]}});**

{ "\_id" : ObjectId("651bb8cd73fd2ea5a7f9df13"), "roll" : 1, "name" : "Calvin", "age" : 20, "marks" : 700 }

{ "\_id" : ObjectId("651bb96873fd2ea5a7f9df15"), "roll" : 3, "name" : "Rahul", "age" : 22, "marks" : 540 }

{ "\_id" : ObjectId("651bcf3973fd2ea5a7f9df18"), "roll" : 2, "name" : "Ankit", "marks" : 750, "hobbies" : [ "writing", "reading", "singing" ] }

{ "\_id" : ObjectId("651bcf4d73fd2ea5a7f9df19"), "roll" : 6, "name" : "Sahil", "marks" : 750, "hobbies" : [ "gaming", "Gym", "dancing" ] }

{ "\_id" : ObjectId("651bcf7873fd2ea5a7f9df1a"), "roll" : 5, "name" : "Apurva", "marks" : 700, "hobbies" : [ "gaming", "sleeping", "singing" ] }

Greater than or equal to 750:

> db.students.find({$expr:{$gte:["$marks",750]}});

{ "\_id" : ObjectId("651bcf3973fd2ea5a7f9df18"), "roll" : 2, "name" : "Ankit", "marks" : 750, "hobbies" : [ "writing", "reading", "singing" ] }

{ "\_id" : ObjectId("651bcf4d73fd2ea5a7f9df19"), "roll" : 6, "name" : "Sahil", "marks" : 750, "hobbies" : [ "gaming", "Gym", "dancing" ] }

1. Create New DataBase

**> use uni**

**> db.uni.insert([{country : 'Spain', city : 'Salamanca', name : 'USAL', location : { type :'Point', coordinates : [ -5.6722512, 40.9607792 ] }, students : [ { year : 2014, number : 24774 },{ year : 2015, number : 23166 }, { year : 2016, number : 21913 },{ year : 2017, number : 21715}], { country : 'Spain', city : 'Salamanca', name : 'UPSA', location : { type : 'Point',coordinates : [ -5.6691191, 40.9631732 ] }, students : [ { year : 2014, number : 4788 }, { year :2015, number : 4821 }, { year : 2016, number : 6550 }, { year : 2017, number : 6125 }]}]);**

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 2,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

1. New DB courses
   1. **> use courses**

**> db.courses.insert([ { university : 'USAL', name : 'Computer Science', level : 'Excellent' }, {university : 'USAL', name : 'Electronics', level : 'Intermediate' }, {university : 'USAL', name :'Communication', level : 'Excellent' }]);**

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 3,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

* 1. **> db.courses.find();**

{ "\_id" : ObjectId("65198934627e32f113e6e7d9"), "university" : "USAL", "name" : "Computer Science", "level" : "Excellent" }

{ "\_id" : ObjectId("65198934627e32f113e6e7da"), "university" : "USAL", "name" : "Electronics","level" : "Intermediate" }

{ "\_id" : ObjectId("65198934627e32f113e6e7db"), "university" : "USAL", "name" : "Communication", "level" : "Excellent" }

1. Aggregation Queries:
   1. **> db.uni.find();**

{ "\_id" : ObjectId("65198819627e32f113e6e7d7"), "country" : "Spain", "city" :"Salamanca","name" : "USAL", "location" : { "type" : "Point", "coordinates" : [ -5.6722512, 40.9607792 ] },"students" : [ { "year" : 2014, "number" : 24774 }, { "year" : 2015, "number" : 23166 }, { "year" :2016, "number" : 21913 }, { "year" : 2017, "number" : 21715 } ] }

{ "\_id" : ObjectId("65198819627e32f113e6e7d8"), "country" : "Spain", "city" : "Salamanca","name" : "UPSA", "location" : { "type" : "Point", "coordinates" : [ -5.6691191, 40.9631732 ] },"students" : [ { "year" : 2014, "number" : 4788 }, { "year" : 2015, "number" : 4821 }, { "year" :2016, "number" : 6550 }, { "year" : 2017, "number" : 6125 } ] }

* 1. Aggregate to find documents matching the country 'Spain' and city 'Salamanca':

**> db.uni.aggregate([{$match:{country:'Spain',city:'Salamanca'}}]);**

{ "\_id" : ObjectId("65198819627e32f113e6e7d7"), "country" : "Spain", "city" : "Salamanca","name" : "USAL", "location" : { "type" : "Point", "coordinates" : [ -5.6722512, 40.9607792 ] },"students" : [ { "year" : 2014, "number" : 24774 }, { "year" : 2015, "number" : 23166 }, { "year" :2016, "number" : 21913 }, { "year" : 2017, "number" : 21715 } ] }

{ "\_id" : ObjectId("65198819627e32f113e6e7d8"), "country" : "Spain", "city" : "Salamanca","name" : "UPSA", "location" : { "type" : "Point", "coordinates" : [ -5.6691191, 40.9631732 ] },"students" : [ { "year" : 2014, "number" : 4788 }, { "year" : 2015, "number" : 4821 }, { "year" :2016, "number" : 6550 }, { "year" : 2017, "number" : 6125 } ] }

* 1. Aggregate to project only the 'country' field:

**> db.uni.aggregate([{$project:{country:1}}]);**

{ "\_id" : ObjectId("65198819627e32f113e6e7d7"), "country" : "Spain" }

{ "\_id" : ObjectId("65198819627e32f113e6e7d8"), "country" : "Spain" }

* 1. Aggregate to project 'country' and 'city' fields:

**> db.uni.aggregate([{$project:{country:1,city:1}}]);**

{ "\_id" : ObjectId("65198819627e32f113e6e7d7"), "country" : "Spain", "city" : "Salamanca" }

{ "\_id" : ObjectId("65198819627e32f113e6e7d8"), "country" : "Spain", "city" : "Salamanca" }

* 1. Aggregate to project 'country', 'city', and 'name' fields:

**> db.uni.aggregate([{$project:{country:1,city:1,name:1}}]);**

{ "\_id" : ObjectId("65198819627e32f113e6e7d7"), "country" : "Spain", "city" : "Salamanca","name" : "USAL" }

{ "\_id" : ObjectId("65198819627e32f113e6e7d8"), "country" : "Spain", "city" : "Salamanca","name" : "UPSA" }

* 1. Aggregate to group documents by 'name' and calculate the total number of documents for each group:

**> db.uni.aggregate([{$group:{\_id:'$name',totalDocs:{$sum:1}}}]);**

{ "\_id" : "USAL", "totalDocs" : 1 }

{ "\_id" : "UPSA", "totalDocs" : 1 }

* 1. Aggregate to project 'country', 'city', and 'name' fields while excluding the '\_id' field:

**> db.uni.aggregate([{$project:{\_id:0,country:1,city:1,name:1}}]);**

{ "country" : "Spain", "city" : "Salamanca", "name" : "USAL" }

{ "country" : "Spain", "city" : "Salamanca", "name" : "UPSA" }

* 1. Aggregate to group documents by 'name', calculate the total number of documents for each group, and output the result to a new collection 'mydata':

**>db.uni.aggregate([{$group:{\_id:'$name',totalDocs:{$sum:1}}},{$out:'mydata'}]);**

**> db.mydata.find();**

{ "\_id" : "UPSA", "totalDocs" : 1 }

{ "\_id" : "USAL", "totalDocs" : 1 }

* 1. Aggregate to match documents with 'name' equal to 'USAL', unwind the 'students' array,and pretty print the result:

**>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'}]).pretty()**;

{

"\_id" : ObjectId("651bda7f73fd2ea5a7f9df1e"),

"country" : "Spain",

"city" : "Salamanca",

"name" : "USAL",

"location" : {

"type" : "Point",

"coordinates" : [

-5.6722512,

40.9607792

]

},

"students" : {

"year" : 2014,

"number" : 24774

}

}

{

"\_id" : ObjectId("651bda7f73fd2ea5a7f9df1e"),

"country" : "Spain",

"city" : "Salamanca",

"name" : "USAL",

"location" : {

"type" : "Point",

"coordinates" : [

-5.6722512,

40.9607792

]

},

"students" : {

"year" : 2015,

"number" : 23166

}

}

{

"\_id" : ObjectId("651bda7f73fd2ea5a7f9df1e"),

"country" : "Spain",

"city" : "Salamanca",

"name" : "USAL",

"location" : {

"type" : "Point",

"coordinates" : [

-5.6722512,

40.9607792

]

},

"students" : {

"year" : 2016,

"number" : 21913

}

}

{

"\_id" : ObjectId("651bda7f73fd2ea5a7f9df1e"),

"country" : "Spain",

"city" : "Salamanca",

"name" : "USAL",

"location" : {

"type" : "Point",

"coordinates" : [

-5.6722512,

40.9607792

]

},

"students" : {

"year" : 2017,

"number" : 21715

}

}

* 1. Aggregate to match documents with 'name' equal to 'USAL', unwind the 'students' array, project the 'year' and 'number' fields excluding '\_id', and pretty print the result:

**>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'},{$project:{\_id:0,'students.year':1,'students.number':1}}]).pretty();**

{ "students" : { "number" : 24774 } }

{ "students" : { "number" : 23166 } }

{ "students" : { "number" : 21913 } }

{ "students" : { "number" : 21715 } }

* 1. **>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'},{$project:{\_id:0,'students.year':1,'students.number':1}},{$sort:{'students.number':-1}}]).pretty();**

{ "students" : { "number" : 24774 } }

{ "students" : { "number" : 23166 } }

{ "students" : { "number" : 21913 } }

{ "students" : { "number" : 21715 } }

**>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'},{$project:{\_id:0,'students.year':1,'students.number':1}},{$sort:{'students.number':1}}]).pretty();**

{ "students" : { "number" : 21715 } }

{ "students" : { "number" : 21913 } }

{ "students" : { "number" : 23166 } }

{ "students" : { "number" : 24774 } }

**>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'},{$project:{\_id:0,'students.year':1,'students.number':1}},{$sort:{'students.number':1}},{$limit:1}]).pretty();**

{ "students" : { "number" : 21715 } }

**>db.uni.aggregate([{$match:{name:'USAL'}},{$unwind:'$students'},{$project:{\_id:0,'students.year':1,'students.number':1}},{$sort:{'students.number':1}},{$skip:2},{$limit:1}]),pretty();**

{ "students" : { "number" : 23166 } }

**>db.uni.aggregate([{$unwind:'$students'},{$count:'totalDocs'}]);**

{ "totalDocs" : 8 }

**>db.uni.aggregate([{$project:{\_id:0,name:1,marks:1}},{$group:{\_id:'$name',totalDocs:{$sum:'$marks'}}}]);**

{ "\_id" : "UPSA", "totalDocs" : 0 }

{ "\_id" : "USAL", "totalDocs" : 0 }

**>db.uni.aggregate([{$project:{\_id:0,name:1,marks:1}},{$group:{\_id:'$name',totalDocs:{$sum:1}}}]);**

{ "\_id" : "UPSA", "totalDocs" : 1 }

{ "\_id" : "USAL", "totalDocs" : 1 }

|  |  |
| --- | --- |
| **Practical No. 3** | |
| **Aim :** To demonstrate Aggregation Pipeline in MongoDB. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : 02 – 09 – 2023 | Sign : |

Create a mongodb schema with name inventory and contains the following attributes.

1. Item name

2. Quantity

3. Size (height, width and unit of measure)

4. Quality (having grade from A – E)

5. Instock (warehouse [Eg: w1, w2], quantity)

Insert queries:

**>db.inventory.insert({iname:"Airbrush", quant:12, size:{height:12,width:12,unit:"cm"},**

**qual:"A",instock:[{whouse:"W1",quantity:8},{whouse:"W4",quantity:4}]});**

**>db.inventory.insert({iname:"Ball", quant:30, size:{height:2,width:2,unit:"cm"},**

**qual:"D",instock:[{whouse:"W4",quantity:20},{whouse:"W2",quantity:10}]});**

**>db.inventory.insertMany([{iname:"Chairs", quant:6, size:{height:0.5,width:0.5,unit:"m"},**

**qual:"C",instock:[{whouse:"W3",quantity:3},{whouse:"W7",quantity:2},{whouse:**

**"W10",quantity:1}]} , {iname:"journal", quant:100, size:{height:24,width:16,unit:"cm"},**

**qual:"B",instock:[{whouse:"W1",quantity:10},{whouse:"W2",quantity:60},{whouse:**

**"W5",quantity:10},{whouse: "W10",quantity:20}]} , {iname:"Erasers ", quant:500,**

**size:{height:40,width:20,unit:"mm"},qual:"D",instock:[{whouse:"W4",quantity:100},{whouse:"W3",quantity:100},{whouse: "W5",quantity:300}]}]);**

**>db.inventory.insertMany([{iname:"Duster", quant:40, size:{height:0.2,width:0.1,unit:"m"},**

**qual:"E",instock:[{whouse:"W1",quantity:12},{whouse:"W2",quantity:12},{whouse:**

**"W10",quantity:26}]} , {iname:"Papers", quant:1000, size:{height:24,width:16,unit:"cm"},**

**qual:"C",instock:[{whouse:"W4",quantity:500},{whouse:"W6",quantity:500}]} , {iname:"Mouse",quant:30, size:{height:10,width:5,unit:"cm"},**

**qual:"A",instock:[{whouse:"W10",quantity:10},{whouse:"W9",quantity:10},{whouse:**

**"W8",quantity:10}]}]);>db.inventory.insertMany([{iname:"Keyboard", quant:42, size:{height:0.2,width:0.5,unit:"m"},qual:"A",instock:[{whouse:"W8",quantity:12},{whouse:"W6",quantity:13},{whouse:"W3",quantity:17}]} , {iname:"Stand", quant:350, size:{height:24,width:30,unit:"cm"},qual:"B",instock:[{whouse:"W4",quantity:250},{whouse:"W6",quantity:100}]}]);**

Questions:

1. Find all items having quantity less than or equal to 10

**> db.inventory.find({quant:{$lte:10}});**

[{

\_id: ObjectId("639ad50a60dc5989073cca6a"),iname: 'Chairs',quant: 6,

size: { height: 0.5, width: 0.5, unit: 'm' },qual: 'C',

instock: [{ whouse: 'W3', quantity: 3 },{ whouse: 'W7', quantity: 2 },{ whouse: 'W10', quantity:1

}]}]

1. Find all items having quality as A and display name and quality only

**> db.inventory.find({qual:"A"},{iname:1,qual:1});**

[{\_id: ObjectId("639ad32160dc5989073cca68"), iname: 'Airbrush',qual: 'A'},

{\_id: ObjectId("639ad5d660dc5989073cca6f"),iname: 'Mouse',qual: 'A'},

{\_id: ObjectId("639ad66e60dc5989073cca70"),iname: 'Keyboard',qual: 'A'}]

OR

**> db.inventory.find({qual:"A"},{iname:1,qual:1,\_id:0});**

[{ iname: 'Airbrush', qual: 'A' },{ iname: 'Mouse', qual: 'A' },{ iname: 'Keyboard', qual: 'A' }]

1. Having instock warehouse as 1 and display names, instock warehouse and in stock quantity

**> db.inventory.find({"instock.whouse":{$eq:"W1"}},{iname:1,\_id:0, "instock.whouse":1});**

[{ iname: 'Airbrush',instock: [ { whouse: 'W1' }, { whouse: 'W4' } ]},{iname: 'journal',instock: [{ whouse: 'W1' },{ whouse: 'W2' },{ whouse: 'W5' },{ whouse: 'W10' }]},{iname: 'Duster',instock: [ { whouse: 'W1' }, { whouse: 'W2' }, { whouse: 'W10' } ] }]

1. Having quality as B and E and display their names and quality

**> db.inventory.find({$or:[{qual:"B"}, {qual:"E"}]},{iname:1,\_id:0, qual:1});**

[{ iname: 'journal', qual: 'B' },{ iname: 'Duster', qual: 'E' },{ iname: 'Stand', qual: 'B' }]

1. Having in stock quantity between 20 and 40 and display names, quantity and instock data

**> db.inventory.find({$and:[{"instock.quantity":{$lte:40}},**

{"instock.quantity":{$gte:20}}]},{iname:1,\_id:0, qual:1, "instock":1});

[{iname: 'Ball', qual: 'D', instock: [ { whouse: 'W4', quantity: 20 }, { whouse: 'W2', quantity: 10 } ] }, {iname: 'journal', qual: 'B', instock: [ { whouse: 'W1', quantity: 10 }, { whouse: 'W2', quantity: 60 }, {whouse: 'W5', quantity: 10 }, { whouse: 'W10', quantity: 20 } ] }, { iname: 'Duster', qual: 'E', instock: [ {whouse: 'W1', quantity: 12 }, { whouse: 'W2', quantity: 12 }, { whouse: 'W10', quantity: 26 } ] }]

1. Find all items starting with S.

**> db.inventory.find({iname:{$regex:'S'}}, {iname:1, quant:1, \_id:0});**

[ { iname: 'Stand', quant: 350 } ]

1. Find all inventory item names having instock warehouse as w2 and w5 and w7, show name,warehouse number, except id

**> db.inventory.find({"instock.whouse":{$in:["W2", "W5", "W7"]}},{iname:1,\_id:0,**

**"instock.whouse":1});**

[

{

iname: 'Ball',

instock: [

{ whouse: 'W4' },

{ whouse: 'W2' }

]

},

{

iname: 'Chairs',

instock: [

{ whouse: 'W3' },

{ whouse: 'W7' },

{ whouse: 'W10' }

]

},

{

iname: 'Journal',

instock: [

{ whouse: 'W1' },

{ whouse: 'W2' },

{ whouse: 'W5' },

{ whouse: 'W10' }

]

},

{

iname: 'Erasers',

instock: [

{ whouse: 'W4' },

{ whouse: 'W3' },

{ whouse: 'W5' }

]

},

{

iname: 'Duster',

instock: [

{ whouse: 'W1' },

{ whouse: 'W2' },

{ whouse: 'W10' }

]

}

]

1. Find all inventory item names having instock warehouse not as w2 and w5 and w7, show name,warehouse number, except id

**> db.inventory.find({"instock.whouse":{$nin:["W2", "W5", "W7"]}},{iname:1,\_id:0, "instock.whouse":1});**

[

{iname: 'Airbrush',instock: [ { whouse: 'W1' }, { whouse: 'W4' } ]},

{ iname: 'Papers', instock: [ { whouse: 'W4' }, { whouse: 'W6' } ] },

{iname: 'Mouse',instock: [ { whouse: 'W10' }, { whouse: 'W9' }, { whouse: 'W8' } ]},

{iname: 'Keyboard',instock: [ { whouse: 'W8' }, { whouse: 'W6' }, { whouse: 'W3' } ]},

{ iname: 'Stand', instock: [ { whouse: 'W4' }, { whouse: 'W6' } ] }

]

1. Item names not having instock quantity greater than equal to 50

**>db.inventory.find({"instock.quantity":{$not:{$gte:50}}},{iname:1,\_id:0, qual:1, "instock":1});**

[

{ iname: 'Airbrush', qual: 'A', instock: [ { whouse: 'W1', quantity: 8 }, { whouse: 'W4', quantity: 4 } ] },{ iname: 'Ball', qual: 'D', instock: [ { whouse: 'W4', quantity: 20 }, { whouse: 'W2', quantity: 10 } ] },{ iname: 'Chairs', qual: 'C', instock: [ { whouse: 'W3', quantity: 3 }, { whouse: 'W7', quantity: 2 }, {whouse: 'W10', quantity: 1 } ] },

{ iname: 'Duster', qual: 'E', instock: [ { whouse: 'W1', quantity: 12 }, { whouse: 'W2', quantity: 12 }, {whouse: 'W10', quantity: 26 } ] },

{ iname: 'Mouse', qual: 'A', instock: [ { whouse: 'W10', quantity: 10 }, { whouse: 'W9', quantity: 10 }, {whouse: 'W8', quantity: 10 } ] },

{ iname: 'Keyboard', qual: 'A', instock: [ { whouse: 'W8', quantity: 12 }, { whouse: 'W6', quantity: 13 }, {whouse: 'W3', quantity: 17 } ] }

]

1. Quality as E and quantity greater than equal to 50

**> db.inventory.find({$and:[{qual:"E"}, {quant:{$gte:20}}]},{iname:1,\_id:0, qual:1, quant:1});**

[ { iname: 'Duster', quant: 40, qual: 'E' } ]

1. Count of inventory having unit of measure as metre

**> db.inventory.find({"size.unit":"m"}).count();**

3

1. Display first 3 documents

**> db.inventory.find().limit(3);**

[{ \_id: ObjectId("639ad32160dc5989073cca68"), iname: 'Airbrush', quant: 12, size: { height: 12, width:12, unit: 'cm' }, qual: 'A', instock: [ { whouse: 'W1', quantity: 8 }, { whouse: 'W4', quantity: 4 } ] },{ \_id: ObjectId("639ad50260dc5989073cca69"), iname: 'Ball', quant: 30, size: { height: 2, width: 2, unit:'cm' }, qual: 'D', instock: [ { whouse: 'W4', quantity: 20 }, { whouse: 'W2', quantity: 10 } ] },{ \_id: ObjectId("639ad50a60dc5989073cca6a"), iname: 'Chairs', quant: 6, size: { height: 0.5, width: 0.5,unit: 'm' }, qual: 'C', instock: [ { whouse: 'W3', quantity: 3 }, { whouse: 'W7', quantity: 2 }, { whouse:'W10', quantity: 1 } ] }]

1. Display the total quantity of all items.
   1. **> db.inventory.aggregate([{$group:{\_id:null, "Sum of quantity":{$sum:"$quant"}}}]);**

[ { \_id: null, 'Sum of quantity': 2110 } ]

* 1. Display the average quantity of items present in each document within the collection.

**> db.inventory.aggregate([{$group:{\_id:null, "Average of quantity":{$avg:"$quant"}}}]);**

[ { \_id: null, 'Average of quantity': 211 } ]

* 1. Display which item has the lowest quantity within the collection.

**> db.inventory.aggregate([{$group:{\_id:null, "Minimum quantity":{$min:"$quant"}}}]);**

[ { \_id: null, 'Minimum quantity': 6 } ]

* 1. Display which item has the highest quantity within the collection.

**> db.inventory.aggregate([{$group:{\_id:null, "Maximum quantity":{$max:"$quant"}}}]);**

[ { \_id: null, 'Minimum quantity': 6 } ]

* 1. Display all the items having unit in cm using $match stage.

**> db.inventory.aggregate([{$match:{"size.unit":"cm"}}, {$project:{iname:1, quant:1}}]);**

{ "\_id" : ObjectId("63a01c2d8065354d1932e7d1"), "iname" : "Stapler", "quant" : 100 }

{ "\_id" : ObjectId("63a01c2d8065354d1932e7d2"), "iname" : "Paper", "quant" : 100 }

{ "\_id" : ObjectId("63a01c2d8065354d1932e7d3"), "iname" : "Eraser", "quant" : 30 }

{ "\_id" : ObjectId("63a01c2d8065354d1932e7d4"), "iname" : "Calculator", "quant" : 5 }

1. Match items having quantity greater than equal to 40 quantity

**> db.inventory.aggregate([{$match: {quant: {$gte:40}}}]);**

[

{

\_id: ObjectId("639ad50a60dc5989073cca6b"),

iname: 'Journal',

quant: 100,

size: { height: 24, width: 16, unit: 'cm' },

qual: 'B',

instock: [

{ whouse: 'W1', quantity: 10 },

{ whouse: 'W2', quantity: 60 },

{ whouse: 'W5', quantity: 10 },

{ whouse: 'W10', quantity: 20 }

]

},

{

\_id: ObjectId("639ad50a60dc5989073cca6c"),

iname: 'Erasers', quant: 500,

size: { height: 40, width: 20, unit: 'mm' },

qual: 'D',

instock: [

{ whouse: 'W4', quantity: 100 },

{ whouse: 'W3', quantity: 100 },

{ whouse: 'W5', quantity: 300 }

]

},

{

\_id: ObjectId("639ad5d660dc5989073cca6e"),

iname: 'Papers',

quant: 1000,

size: { height: 24, width: 16, unit: 'cm' },

qual: 'C',

instock: [

{ whouse: 'W4', quantity: 500 },

{ whouse: 'W6', quantity: 500 }

]

},

{

\_id: ObjectId("639ad66e60dc5989073cca70"),

iname: 'Keyboard',

quant: 42,

size: { height: 0.2, width: 0.5, unit: 'm' },

qual: 'A',

instock: [

{ whouse: 'W8', quantity: 12 },

{ whouse: 'W6', quantity: 13 },

{ whouse: 'W3', quantity: 17 }

]

},

{

\_id: ObjectId("639ad66e60dc5989073cca71"),

iname: 'Stand',

quant: 350,

size: { height: 24, width: 30, unit: 'cm' },

qual: 'B',

instock: [

{ whouse: 'W4', quantity: 250 },

{ whouse: 'W6', quantity: 100 }

]

}

]

1. Items having Unit of Measure as metre and display item name and size

**> db.inventory.aggregate([{$match: {"size.unit":"m"}}, {$project: {iname:1,\_id:0, "size.unit":1}}]);**

[ { iname: 'Chairs', size: { unit: 'm' } },

{ iname: 'Duster', size: { unit: 'm' } },

{ iname: 'Keyboard', size: { unit: 'm' } }]

1. Display all inventory items using $match in aggregation

**> db.inventory.aggregate([{$match: {}}]);**

[

{

\_id: ObjectId("639ad32160dc5989073cca68"),

iname: 'Airbrush',

quant: 12,

size: { height: 12, width: 12, unit: 'cm' },

qual: 'A',

instock: [

{ whouse: 'W1', quantity: 8 },

{ whouse: 'W4', quantity: 4 }

]

},

{

\_id: ObjectId("639ad50260dc5989073cca69"),

iname: 'Ball',

quant: 30,

size: { height: 2, width: 2, unit: 'cm' },

qual: 'D',

instock: [

{ whouse: 'W4', quantity: 20 },

{ whouse: 'W2', quantity: 10 }

]

},

{

\_id: ObjectId("639ad50a60dc5989073cca6a"),

iname: 'Chairs',

quant: 6,

size: { height: 0.5, width: 0.5, unit: 'm' },

qual: 'C',

instock: [

{ whouse: 'W3', quantity: 3 },

{ whouse: 'W7', quantity: 2 },

{ whouse: 'W10', quantity: 1 }

]

},

{

\_id: ObjectId("639ad50a60dc5989073cca6b"),

iname: 'Journal',

quant: 100,

size: { height: 24, width: 16, unit: 'cm' },

qual: 'B',

instock: [

{ whouse: 'W1', quantity: 10 },

{ whouse: 'W2', quantity: 60 },

{ whouse: 'W5', quantity: 10 },

{ whouse: 'W10', quantity: 20 }

]

},

{

\_id: ObjectId("639ad5d660dc5989073cca6d"),

iname: 'Duster',

quant: 40,

size: { height: 0.2, width: 0.1, unit: 'm' },

qual: 'E',

instock: [

{ whouse: 'W1', quantity: 12 },

{ whouse: 'W2', quantity: 12 },

{ whouse: 'W10', quantity: 26 }

]

},

{

\_id: ObjectId("639ad5d660dc5989073cca6e"),

iname: 'Papers',

quant: 1000,

size: { height: 24, width: 16, unit: 'cm' },

qual: 'C',

instock: [

{ whouse: 'W4', quantity: 500 },

{ whouse: 'W6', quantity: 500 }

]

},

{

\_id: ObjectId("639ad5d660dc5989073cca6f"),

iname: 'Mouse',

quant: 30,

size: { height: 10, width: 5, unit: 'cm' },

qual: 'A',

instock: [

{ whouse: 'W10', quantity: 10 },

{ whouse: 'W9', quantity: 10 },

{ whouse: 'W8', quantity: 10 }

]

},

{

\_id: ObjectId("639ad66e60dc5989073cca70"),

iname: 'Keyboard',

quant: 42,

size: { height: 0.2, width: 0.5, unit: 'm' },

qual: 'A',

instock: [

{ whouse: 'W8', quantity: 12 },

{ whouse: 'W6', quantity: 13 },

{ whouse: 'W3', quantity: 17 }

]

},

{

\_id: ObjectId("639ad66e60dc5989073cca71"),

iname: 'Stand',

quant: 350,

size: { height: 24, width: 30, unit: 'cm' },

qual: 'B',

instock: [

{ whouse: 'W4', quantity: 250 },

{ whouse: 'W6', quantity: 100 }

]

}

]

1. Item having names: Chairs, journal, duster and display its: Name quantity unit

**> db.inventory.aggregate([{$match: {iname: {$in: ["Chairs","journal","Duster"]}}}, {$project:**

**{iname:1,\_id:0, quant:1, "size.unit":1}}]);**

[{ iname: 'Chairs', quant: 6, size: { unit: 'm' } },{ iname: 'journal', quant: 100, size: { unit: 'cm' } },{ iname:'Duster', quant: 40, size: { unit: 'm' } }]

1. Item names not having: Chairs, journal, duster and display its: Name quantity unit

**> db.inventory.aggregate([{$match: {iname: {$nin: ["Chairs","journal","Duster"]}}}, {$project:{iname:1,\_id:0, quant:1, "size.unit":1}}]);**

[ { iname: 'Airbrush', quant: 12, size: { unit: 'cm' } },

{ iname: 'Ball', quant: 30, size: { unit: 'cm' } },

{ iname: 'Erasers ', quant: 500, size: { unit: 'mm' } },

{ iname: 'Papers', quant: 1000, size: { unit: 'cm' } },

{ iname: 'Stand', quant: 350, size: { unit: 'cm' } }]

1. Display all items having instock quantity between 50 and 100, display item name, unit of measure and instock details.

**> db.inventory.aggregate([{$match: {$and:[{"instock.quantity":{$lte:100}}, {"instock.quantity":{$gte:50}}]}}, {$project: {iname:1,\_id:0,"size.unit":1,"instock":1}}]);**

[{ iname: 'journal', size: { unit: 'cm' }, instock: [ { whouse: 'W1', quantity: 10 }, { whouse: 'W2', quantity: 60

}, { whouse: 'W5', quantity: 10 }, { whouse: 'W10', quantity: 20 } ] },

{ iname: 'Erasers ', size: { unit: 'mm' }, instock: [ { whouse: 'W4', quantity: 100 }, { whouse: 'W3', quantity:

100 }, { whouse: 'W5', quantity: 300 } ] },

{ iname: 'Stand', size: { unit: 'cm' }, instock: [ { whouse: 'W4', quantity: 250 }, { whouse: 'W6', quantity: 100

} ] }]

1. Display all items sorted on item names

**> db.inventory.aggregate([{$match: {}}, {$sort: {iname:1}}]);**

[ { \_id: ObjectId("639ad32160dc5989073cca68"), iname: 'Airbrush', quant: 12, size: { height: 12, width: 12,

unit: 'cm' }, qual: 'A', instock: [ { whouse: 'W1', quantity: 8 }, { whouse: 'W4', quantity: 4 } ] }, { \_id:

ObjectId("639ad50260dc5989073cca69"), iname: 'Ball', quant: 30, size: { height: 2, width: 2, unit: 'cm' },

qual: 'D', instock: [ { whouse: 'W4', quantity: 20 }, { whouse: 'W2', quantity: 10 } ] }, { \_id:

ObjectId("639ad50a60dc5989073cca6a"), iname: 'Chairs', quant: 6, size: { height: 0.5, width: 0.5, unit: 'm'

}, qual: 'C', instock: [ { whouse: 'W3', quantity: 3 }, { whouse: 'W7', quantity: 2 }, { whouse: 'W10',

quantity: 1 } ] }, { \_id: ObjectId("639ad5d660dc5989073cca6d"), iname: 'Duster', quant: 40, size: { height:

0.2, width: 0.1, unit: 'm' }, qual: 'E', instock: [ { whouse: 'W1', quantity: 12 }, { whouse: 'W2', quantity: 12 },

{ whouse: 'W10', quantity: 26 } ] }, { \_id: ObjectId("639ad50a60dc5989073cca6c"), iname: 'Erasers ', quant:

500, size: { height: 40, width: 20, unit: 'mm' }, qual: 'D', instock: [ { whouse: 'W4', quantity: 100 }, {

whouse: 'W3', quantity: 100 }, { whouse: 'W5', quantity: 300 } ] }, { \_id:

ObjectId("639ad66e60dc5989073cca70"), iname: 'Keyboard', quant: 42, size: { height: 0.2, width: 0.5, unit:

'm' }, qual: 'A', instock: [ { whouse: 'W8', quantity: 12 }, { whouse: 'W6', quantity: 13 }, { whouse: 'W3',

quantity: 17 } ] }, { \_id: ObjectId("639ad5d660dc5989073cca6f"), iname: 'Mouse', quant: 30, size: { height:

10, width: 5, unit: 'cm' }, qual: 'A', instock: [ { whouse: 'W10', quantity: 10 }, iname: 'Papers', quant: 1000,

size: { height: 24, width: 16, unit: 'cm' }, qual: 'C', instock: [ { whouse: 'W4', quantity: 500 }, { whouse:

'W6', quantity: 500 } ] }, { \_id: ObjectId("639ad66e60dc5989073cca71"), iname: 'Stand', quant: 350, size: {

height: 24, width: 30, unit: 'cm' }, qual: 'B', instock: [ { whouse: 'W4', quantity: 250 }, { whouse: 'W6',

quantity: 100 } ] }, { \_id: ObjectId("639ad50a60dc5989073cca6b"), iname: 'journal', quant: 100, size: {

height: 24, width: 16, unit: 'cm' }, qual: 'B', instock: [ { whouse: 'W1', quantity: 10 }, { whouse: 'W2',

quantity: 60 }, { whouse: 'W5', quantity: 10 }, { whouse: 'W10', quantity: 20 } ] }]

1. Display all items having quantity greater than equal to 70 sorted by their quantity in descending order and display their iname and quantity

**> db.inventory.aggregate([{$match: {quant:{$gte:70}}}, {$sort: {quant:-1}}, {$project:{iname:1,quant:1, \_id:0}}]);**

[ { iname: 'Papers', quant: 1000 },

{ iname: 'Erasers ', quant: 500 },

{ iname: 'Stand', quant: 350 },

{ iname: 'journal', quant: 100 }]

1. Adding additional records to perform group aggregation on:

**>db.inventory.insertMany([{ iname: 'Airbrush', quant: 55, size: { height: 52, width: 26, unit: 'm' },qual: 'D', instock: [ { whouse: 'W2', quantity: 65 }, { whouse: 'W4', quantity: 8 } ] }, { iname: 'Ball',quant: 39, size: { height: 7, width: 5, unit: 'cm' }, qual: 'D', instock: [ { whouse: 'W4', quantity: 82 }, {whouse: 'W2', quantity: 12 } ] }, { iname: 'Chairs', quant: 47, size: { height: 48, width: 20, unit: 'm' },qual: 'C', instock: [ { whouse: 'W3', quantity: 25 }, { whouse: 'W7', quantity: 3 }, { whouse: 'W10',quantity: 8 } ] }, { iname: 'journal', quant: 120, size: { height: 24, width: 16, unit: 'cm' }, qual: 'B',instock: [ { whouse: 'W1', quantity: 85 }, { whouse: 'W2', quantity: 20 }, { whouse: 'W5', quantity: 18}, { whouse: 'W10', quantity: 23 } ] }, { iname: 'Erasers ', quant: 58, size: { height: 45, width: 70, unit:'mm' }, qual: 'D', instock: [ { whouse: 'W4', quantity: 550 }, { whouse: 'W3', quantity: 140 }, { whouse:'W5', quantity: 340 } ] }, { iname: 'Duster', quant: 40, size: { height: 0.2, width: 0.1, unit: 'm' }, qual:'E', instock: [ { whouse: 'W1', quantity: 12 }, { whouse: 'W2', quantity: 12 }, { whouse: 'W10',quantity: 26 } ] }, { iname: 'Papers', quant: 1250, size: { height: 24, width: 16, unit: 'cm' }, qual: 'C',instock: [ { whouse: 'W4', quantity: 700 }, { whouse: 'W6', quantity: 570 } ] }, { iname: 'Mouse',**

**quant:30, size: { height: 10, width: 5, unit: 'cm' }, qual: 'A', instock: [ { whouse: 'W10', quantity: 10 }, {whouse: 'W9', quantity: 10 }, { whouse: 'W8', quantity: 10 } ] }, { iname: 'Keyboard', quant: 42, size: {height: 0.2, width: 0.5, unit: 'm' }, qual: 'A', instock: [ { whouse: 'W8', quantity: 12 }, { whouse: 'W6',quantity: 13 }, { whouse: 'W3', quantity: 17 } ] }, { iname: 'Stand', quant: 350, size: { height: 24,width: 30, unit: 'cm' }, qual: 'B', instock: [ { whouse: 'W4', quantity: 250 }, { whouse: 'W6', quantity:**

**100 } ] } ]);**

1. Display all items with name journals

**> db.inventory.aggregate([{$match: {iname:"journal"}}]);**

[{ \_id: ObjectId("639ad50a60dc5989073cca6b"), iname: 'journal', quant: 100, size: { height: 24, width: 16,unit: 'cm' }, qual: 'B', instock: [ { whouse: 'W1', quantity: 10 }, { whouse: 'W2', quantity: 60 }, { whouse:'W5', quantity: 10 }, { whouse: 'W10', quantity: 20 } ] }, { \_id: ObjectId("63a018397235907b7ffe3859"),iname: 'journal', quant: 120, size: { height: 24, width: 16, unit: 'cm' }, qual: 'B', instock: [ { whouse: 'W1',quantity: 85 }, { whouse: 'W2', quantity: 20 }, { whouse: 'W5', quantity: 18 }, { whouse: 'W10', quantity: 23

}] }]

1. Display all inventory item names grouped by their name

**> db.inventory.aggregate([{$group:{\_id:"$iname"}}]);**

[ { \_id: 'Ball' }, { \_id: 'Duster' }, { \_id: 'Chairs' }, { \_id: 'Airbrush' }, { \_id: 'Erasers ' }, { \_id: 'Mouse' },

{ \_id: 'Papers' }, { \_id: 'journal' }, { \_id: 'Keyboard' }, { \_id: 'Stand' }]

1. Count the number of items in a particular group, by item name

**> db.inventory.aggregate([{$group:{\_id: "$iname", "Total number of items are: ": {$sum:1}}}]);**

[{ \_id: 'Chairs', 'Total number of items are: ': 2 },

{ \_id: 'Papers', 'Total number of items are: ': 2 },

{ \_id: 'journal', 'Total number of items are: ': 2 },

{ \_id: 'Airbrush', 'Total number of items are: ': 2 },

{ \_id: 'Keyboard', 'Total number of items are: ': 2 }]

1. Count the number of items in a particular group, by item name and sort them in ascending order

**> db.inventory.aggregate([{$group:{\_id: "$iname", count: {$sum:1}}}, {$sort: {\_id:1}}]);**

[{ \_id: 'Airbrush', count: 2 },

{ \_id: 'Ball', count: 2 },

{ \_id: 'Chairs', count: 2 },

{ \_id: 'Duster', count: 2 },

{ \_id: 'Erasers ', count: 2 },

{ \_id: 'Keyboard', count: 2 },

{ \_id: 'Mouse', count: 2 },

{ \_id: 'Papers', count: 2 },

{ \_id: 'Stand', count: 2 },

{ \_id: 'journal', count: 2 }]

1. Count the number of items in a particular group, by item name and sort them in descending order.

**> db.inventory.aggregate([{$group:{\_id: "$iname", count: {$sum:1}}}, {$sort: {\_id:-1}}]);**

[{ \_id: 'journal', count: 2 },

{ \_id: 'Stand', count: 2 },

{ \_id: 'Papers', count: 2 },

{ \_id: 'Mouse', count: 2 },

{ \_id: 'Keyboard', count: 2 },

{ \_id: 'Erasers ', count: 2 },

{ \_id: 'Duster', count: 2 },

{ \_id: 'Chairs', count: 2 },

{ \_id: 'Ball', count: 2 },

{ \_id: 'Airbrush', count: 2 }]

1. Display total quantities of all items grouped by item name and sort ascendingly

**> db.inventory.aggregate([{$group:{\_id: "$iname", count: {$sum:"$quant"}}}, {$sort: {\_id:1}}]);**

[{ \_id: 'Airbrush', count: 67 },

{ \_id: 'Ball', count: 69 },

{ \_id: 'Chairs', count: 53 },

{ \_id: 'Duster', count: 80 },

{ \_id: 'Erasers ', count: 558 },

{ \_id: 'Keyboard', count: 84 },

{ \_id: 'Mouse', count: 60 },

{ \_id: 'Papers', count: 2250 },

{ \_id: 'Stand', count: 700 },

{ \_id: 'journal', count: 220 }]

1. Display total quantities of all items grouped by item name and sort ascendingly having count gte 80

**> db.inventory.aggregate([{$group:{\_id: "$iname", count: {$sum:"$quant"}}}, {$sort: {\_id:1}},{$match: {count:{$gte:80}}}]);**

[{ \_id: 'Duster', count: 80 },

{ \_id: 'Erasers ', count: 558 },

{ \_id: 'Keyboard', count: 84 },

{ \_id: 'Papers', count: 2250 },

{ \_id: 'Stand', count: 700 },

{ \_id: 'journal', count: 220 }]

1. Display all names with their quantity, using project first and then match gte 50

**> db.inventory.aggregate([{$project: {\_id:0, iname:1, quant:1}}, {$match: {quant:{$gte:50}}}]);**

[{ iname: 'journal', quant: 100 },

{ iname: 'Erasers ', quant: 500 },

{ iname: 'Papers', quant: 1000 },

{ iname: 'Stand', quant: 350 },

{ iname: 'Airbrush', quant: 55 },

{ iname: 'journal', quant: 120 },

{ iname: 'Erasers ', quant: 58 },

{ iname: 'Papers', quant: 1250 },

{ iname: 'Stand', quant: 350 }]

1. Display all names with their quantity, using project first and then match gte 50, group and count the number of items by item name sorted on their total count

**> db.inventory.aggregate([{$project: {\_id:0, iname:1, quant:1}}, {$match: {quant:{$gte:50}}},{$group:{\_id: "$iname", count: {$sum:1}}}, {$sort:{count:1}}]);**

[{ \_id: 'Airbrush', count: 1 },

{ \_id: 'journal', count: 2 },

{ \_id: 'Erasers ', count: 2 },

{ \_id: 'Papers', count: 2 },

{ \_id: 'Stand', count: 2 }]

1. Display all names with their quantity, using project first and then match gte 50, group and calculate the number of items by item name sorted on their total quantity

**> db.inventory.aggregate([{$project: {\_id:0, iname:1, quant:1}}, {$match: {quant:{$gte:50}}},{$group:{\_id: “$iname”, count: {$sum:”$quant”}}}, {$sort:{count:1}}]);**

[ { \_id: 'Airbrush', count: 55 },

{ \_id: 'journal', count: 220 },

{ \_id: 'Erasers ', count: 558 },

{ \_id: 'Stand', count: 700 },

{ \_id: 'Papers', count: 2250 }]

1. Group by name as well as quantity

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname",Quantity:"$quant"}, totalcount:{$sum:1}}}]);**

[{ \_id: { Name: 'Chairs', Quantity: 6 }, totalcount: 1 },

{ \_id: { Name: 'Stand', Quantity: 350 }, totalcount: 3 },

{ \_id: { Name: 'journal', Quantity: 120 }, totalcount: 2 },

{ \_id: { Name: 'Erasers ', Quantity: 500 }, totalcount: 1 },

{ \_id: { Name: 'Duster', Quantity: 40 }, totalcount: 3 },

{ \_id: { Name: 'Airbrush', Quantity: 55 }, totalcount: 2 },

{ \_id: { Name: 'journal', Quantity: 100 }, totalcount: 1 },

{ \_id: { Name: 'Ball', Quantity: 30 }, totalcount: 1 },

{ \_id: { Name: 'Papers', Quantity: 1250 }, totalcount: 2 },

{ \_id: { Name: 'Erasers ', Quantity: 58 }, totalcount: 2 },

{ \_id: { Name: 'Papers', Quantity: 1000 }, totalcount: 1 },

{ \_id: { Name: 'Mouse', Quantity: 30 }, totalcount: 3 },

{ \_id: { Name: 'Keyboard', Quantity: 42 }, totalcount: 3 },

{ \_id: { Name: 'Airbrush', Quantity: 12 }, totalcount: 1 },

{ \_id: { Name: 'Ball', Quantity: 39 }, totalcount: 2 },

{ \_id: { Name: 'Chairs', Quantity: 47 }, totalcount: 2 }]

1. Calculate total quantity of items grouped by item name and quantity

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname", Quantity:"$quant"},**

**count:{$sum:1},totalquantity:{$sum:"$quant"}}}]);**

[{ \_id: { Name: 'Chairs', Quantity: 6 }, count: 1, totalquantity: 6 },

{ \_id: { Name: 'Stand', Quantity: 350 },count: 3,totalquantity: 1050},

{\_id: { Name: 'journal', Quantity: 120 },count: 2,totalquantity: 240 },

{\_id: { Name: 'Erasers ', Quantity: 500 },count: 1,totalquantity: 500},

{\_id: { Name: 'Duster', Quantity: 40 },count: 3,totalquantity: 120},

{\_id: { Name: 'Airbrush', Quantity: 55 },count: 2,totalquantity: 110},

{\_id: { Name: 'journal', Quantity: 100 },count: 1,totalquantity: 100},

{ \_id: { Name: 'Ball', Quantity: 30 }, count: 1, totalquantity: 30 },

{\_id: { Name: 'Papers', Quantity: 1250 }, count: 2,totalquantity: 2500},

{\_id: { Name: 'Erasers ', Quantity: 58 },count: 2,totalquantity: 116}]

1. Calculate total quantity of items grouped by item name and quantity and sort by iname

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname",**

**Quantity:"$quant"},totalquantity:{$sum:"$quant"}}}, {$sort:{"\_id.Name":1}}]);**

[{ \_id: { Name: 'Airbrush', Quantity: 12 }, totalquantity: 12 },

{ \_id: { Name: 'Airbrush', Quantity: 55 }, totalquantity: 110 },

{ \_id: { Name: 'Ball', Quantity: 39 }, totalquantity: 78 },

{ \_id: { Name: 'Ball', Quantity: 30 }, totalquantity: 30 },

{ \_id: { Name: 'Chairs', Quantity: 6 }, totalquantity: 6 },

{ \_id: { Name: 'Chairs', Quantity: 47 }, totalquantity: 94 },

{ \_id: { Name: 'Duster', Quantity: 40 }, totalquantity: 120 },

{ \_id: { Name: 'Erasers ', Quantity: 58 }, totalquantity: 116 },

{ \_id: { Name: 'Erasers ', Quantity: 500 }, totalquantity: 500 },

{ \_id: { Name: 'Keyboard', Quantity: 42 }, totalquantity: 126 },

{ \_id: { Name: 'Mouse', Quantity: 30 }, totalquantity: 90 },

{ \_id: { Name: 'Papers', Quantity: 1000 }, totalquantity: 1000 },

{ \_id: { Name: 'Papers', Quantity: 1250 }, totalquantity: 2500 },

{ \_id: { Name: 'Stand', Quantity: 350 }, totalquantity: 1050 },

{ \_id: { Name: 'journal', Quantity: 120 }, totalquantity: 240 },

{ \_id: { Name: 'journal', Quantity: 100 }, totalquantity: 100 }]

1. Calculate total quantity of items grouped by item name and quantity and sort by iname, display all records except first 2 records.

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname",**

**Quantity:"$quant"},totalquantity:{$sum:"$quant"}}}, {$sort:{"\_id.Name":1}}, {$skip:2}]);**

[ { \_id: { Name: 'Ball', Quantity: 30 }, totalquantity: 30 },

{ \_id: { Name: 'Ball', Quantity: 39 }, totalquantity: 78 },

{ \_id: { Name: 'Chairs', Quantity: 6 }, totalquantity: 6 },

{ \_id: { Name: 'Chairs', Quantity: 47 }, totalquantity: 94 },

{ \_id: { Name: 'Duster', Quantity: 40 }, totalquantity: 120 },

{ \_id: { Name: 'Erasers ', Quantity: 500 }, totalquantity: 500 },

{ \_id: { Name: 'Erasers ', Quantity: 58 }, totalquantity: 116 },

{ \_id: { Name: 'Keyboard', Quantity: 42 }, totalquantity: 126 },

{ \_id: { Name: 'Mouse', Quantity: 30 }, totalquantity: 90 },

{ \_id: { Name: 'Papers', Quantity: 1250 }, totalquantity: 2500 },

{ \_id: { Name: 'Papers', Quantity: 1000 }, totalquantity: 1000 },

{ \_id: { Name: 'Stand', Quantity: 350 }, totalquantity: 1050 },

{ \_id: { Name: 'journal', Quantity: 120 }, totalquantity: 240 },

{ \_id: { Name: 'journal', Quantity: 100 }, totalquantity: 100 }]

1. Calculate total quantity of items grouped by item name and quantity and sort by iname, display all records except first 2 records and limit upto 3.

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname",**

**Quantity:"$quant"},totalquantity:{$sum:"$quant"}}}, {$sort:{"\_id.Name":1}}, {$skip:2},{$limit:3}]);**

[ { \_id: { Name: 'Ball', Quantity: 30 }, totalquantity: 30 },

{ \_id: { Name: 'Ball', Quantity: 39 }, totalquantity: 78 },

{ \_id: { Name: 'Chairs', Quantity: 47 }, totalquantity: 94 }]

1. Average:

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname", Quantity:"$quant"}, count:{$sum:1},totalquantity:{$avg:"$quant"}}}]);**

[{ \_id: { Name: 'Chairs', Quantity: 6 }, count: 1, totalquantity: 6 },

{ \_id: { Name: 'journal', Quantity: 120 }, count: 2, totalquantity: 120 },

{ \_id: { Name: 'Airbrush', Quantity: 12 }, count: 1, totalquantity: 12 },

{ \_id: { Name: 'Mouse', Quantity: 30 }, count: 3, totalquantity: 30 },

{ \_id: { Name: 'Keyboard', Quantity: 42 }, count: 3, totalquantity: 42 },

{ \_id: { Name: 'Papers', Quantity: 1000 }, count: 1, totalquantity: 1000 },

{ \_id: { Name: 'Ball', Quantity: 39 }, count: 2, totalquantity: 39 },

{ \_id: { Name: 'Chairs', Quantity: 47 }, count: 2, totalquantity: 47 },

{ \_id: { Name: 'Erasers ', Quantity: 58 }, count: 2, totalquantity: 58 },

{ \_id: { Name: 'Papers', Quantity: 1250 }, count: 2, totalquantity: 1250 },

{ \_id: { Name: 'Ball', Quantity: 30 }, count: 1, totalquantity: 30 },

{ \_id: { Name: 'journal', Quantity: 100 }, count: 1, totalquantity: 100 },

{ \_id: { Name: 'Erasers ', Quantity: 500 }, count: 1, totalquantity: 500 },

{ \_id: { Name: 'Duster', Quantity: 40 }, count: 3, totalquantity: 40 },

{ \_id: { Name: 'Airbrush', Quantity: 55 }, count: 2, totalquantity: 55 },

{ \_id: { Name: 'Stand', Quantity: 350 }, count: 3, totalquantity: 350 }]

1. Grouped by name and average

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname"}, count: {$sum:1},**

**Average:{$avg:"$quant"}}}]);**

[{ \_id: { Name: 'Ball' }, count: 3, Average: 36 },

{ \_id: { Name: 'Mouse' }, count: 3, Average: 30 },

{ \_id: { Name: 'Stand' }, count: 3, Average: 350 },

{ \_id: { Name: 'Duster' }, count: 3, Average: 40 },

{ \_id: { Name: 'Keyboard' }, count: 3, Average: 42 },

{ \_id: { Name: 'Chairs' }, count: 3, Average: 33.333333333333336 },

{ \_id: { Name: 'Papers' }, count: 3, Average: 1166.6666666666667 },

{ \_id: { Name: 'journal' }, count: 3, Average: 113.33333333333333 },

{ \_id: { Name: 'Erasers ' }, count: 3, Average: 205.33333333333334 },

{ \_id: { Name: 'Airbrush' }, count: 3, Average: 40.666666666666664 }]

OR

**> db.inventory.aggregate([{$group:{\_id: {Name:"$iname"}, count: {$sum:1}, TotalQuantity:{$sum:"$quant"}, Average:{$avg:"$quant"}}}]);**

[ { \_id: { Name: 'Ball' }, count: 3, TotalQuantity: 108, Average: 36 },

{ \_id: { Name: 'Stand' }, count: 3, TotalQuantity: 1050, Average: 350},

{ \_id: { Name: 'Airbrush' }, count: 3, TotalQuantity: 122, Average: 40.666666666666664},

{ \_id: { Name: 'journal' },count: 3,TotalQuantity: 340, Average: 113.33333333333333 },

{ \_id: { Name: 'Erasers ' }, count: 3, TotalQuantity: 616, Average: 205.33333333333334 },

{ \_id: { Name: 'Papers' }, count: 3, TotalQuantity: 3500, Average: 1166.6666666666667 },

{ \_id: { Name: 'Chairs' }, count: 3, TotalQuantity: 100, Average: 33.333333333333336 },

{ \_id: { Name: 'Keyboard' }, count: 3, TotalQuantity: 126, Average: 42 },

{ \_id: { Name: 'Duster' }, count: 3, TotalQuantity: 120, Average: 40 },

{ \_id: { Name: 'Mouse' }, count: 3, TotalQuantity: 90, Average: 3

|  |  |
| --- | --- |
| **Practical No. 4** | |
| **Aim :** To demonstrate the Usage of MongoDB in Python. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : 04 - 09 - 2023 | Sign : |

1. Download and install pymongo driver to access mongodb database and test the same.

**> python –m pip install pymongo**

1. Create a database “customer” in mongodb using python and perform CRUD operations

on the same.

**Input and Output:**

**import pymongo**

**#Creating connection**

**db = pymongo.MongoClient("mongodb://127.0.0.1:27017/")**

**#Create database**

**dbase = db["Students"]**

**print("Database Created")**

**#Output**

**Database Created**

1. **#Add data and check**

**data = dbase["Student"]**

1. **#List all dbs in mongodb**

print("All Databases:"+str(db.list\_database\_names()))

**#Output**

All Databases:['admin', 'config', 'courses', 'local', 'msc1', 'uni']

1. **#Check if database already exists**

dblist = db.list\_database\_names()

if "Students" in dblist:

print("Already Existing")

else:

print("Doesn't Exist")

**Output**

Doesn't Exist

1. **Insert 1 record**

dict1 = {"name":"Calvin","age":12,"marks":"67"}

ins1 = data.insert\_one(dict1)

print("\nInserted Record: "+str(ins1.inserted\_id))

**Output**

Inserted Record: 652024ea67be9c57a48e047e

1. **Insert multiple records**

l1= [{"Name":"Smith","age":21,"marks":102},

{"Name":"Keanu","age":22,"marks":103},

{"Name":"Thomas","age":23,"marks":104},

{"Name":"Percy","age":21,"marks":105},

{"Name":"Creed","age":22,"marks":106},

{"Name":"Griffin","age":23,"marks":107},

{"Name":"Jim","age":21,"marks":104}]

c = data.insert\_many(l1)

print("Records inserted: "+str(c.inserted\_ids))

**Output**

Records inserted: [ObjectId('6520303cefcefe07f0fe38ed'), ObjectId('6520303cefcefe07f0fe38ee'), ObjectId('6520303cefcefe07f0fe38ef'), ObjectId('6520303cefcefe07f0fe38f0'), ObjectId('6520303cefcefe07f0fe38f1'), ObjectId('6520303cefcefe07f0fe38f2'), ObjectId('6520303cefcefe07f0fe38f3')]

1. **Print all docs in collection using find method**

print("Records:\n\n")

for data1 in data.find():

print(data1)

**Output**

Records:

{'\_id': ObjectId('6520303cefcefe07f0fe38ec'), 'Name': 'Calvin', 'age': 22, 'marks': 101}

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Smith', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ee'), 'Name': 'Keanu', 'age': 22, 'marks': 103}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 23, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f0'), 'Name': 'Percy', 'age': 21, 'marks': 105}

{'\_id': ObjectId('6520303cefcefe07f0fe38f1'), 'Name': 'Creed', 'age': 22, 'marks': 106}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 21, 'marks': 104}

1. **Find document with address**

myq = {"Name":"Calvin"}

mydoc1 = data.find(myq)

print("\n\nRecord with Matched Name: ")

for i in mydoc1:

print(i)

**Output**

Record with Matched Name:

{'\_id': ObjectId('6520303cefcefe07f0fe38ec'), 'Name': 'Calvin', 'age': 22, 'marks': 101}

1. **Sort based on name**

sorted1 = data.find().sort("age")

print("\n\nSorted Records: ")

for i in sorted1:

print(i)

**Output**

Sorted Records:

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Smith', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38f0'), 'Name': 'Percy', 'age': 21, 'marks': 105}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 21, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38ec'), 'Name': 'Calvin', 'age': 22, 'marks': 101}

{'\_id': ObjectId('6520303cefcefe07f0fe38ee'), 'Name': 'Keanu', 'age': 22, 'marks': 103}

{'\_id': ObjectId('6520303cefcefe07f0fe38f1'), 'Name': 'Creed', 'age': 22, 'marks': 106}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 23, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

1. **Delete one**

del1 = {"Name":"Percy"}

d1 = data.delete\_one(del1)

var = data.find()

print("\n\nRecords after Deletion: ")

for i in var:

print(i)

print("Records deleted: "+str(d1.deleted\_count))

**Output**

{'\_id': ObjectId('6520303cefcefe07f0fe38ec'), 'Name': 'Calvin', 'age': 22, 'marks': 101}

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Smith', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ee'), 'Name': 'Keanu', 'age': 22, 'marks': 103}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 23, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f1'), 'Name': 'Creed', 'age': 22, 'marks': 106}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 21, 'marks': 104}

Records deleted: 1

1. **Delete many**

del2 = {"age":22}

d2 = data.delete\_many(del2)

var = data.find()

print("\n\nRecords after Deletion: ")

for i in var:

print(i)

print("Records deleted: "+str(d2.deleted\_count))

**Output**

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Smith', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 23, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 21, 'marks': 104}

Records deleted: 3

1. **Update one record**

up1 = {"marks":102}

set1 = {"$set":{"Name":"Michael"}}

u1 = data.update\_one(up1, set1)

var = data.find()

for i in var:

print(i)

print("\n\nUpdated record count: "+str(u1.modified\_count))

**Output**

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Michael', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 23, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 21, 'marks': 104}

Updated record count: 1

1. **Update many**

up2 = {"marks":104}

set2 = {"$set":{"age":31}}

u2 = data.update\_many(up2, set2)

var = data.find()

for i in var:

print(i)

print("\n\nUpdated record count: "+str(u2.modified\_count))

#Output

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Michael', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 31, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

{'\_id': ObjectId('6520303cefcefe07f0fe38f3'), 'Name': 'Jim', 'age': 31, 'marks': 104}

Updated record count: 2

1. **Limit**

var = data.find().limit(3)

print("\n\nLimited Records: ")

for i in var:

print(i)

Output

Limited Records:

{'\_id': ObjectId('6520303cefcefe07f0fe38ed'), 'Name': 'Michael', 'age': 21, 'marks': 102}

{'\_id': ObjectId('6520303cefcefe07f0fe38ef'), 'Name': 'Thomas', 'age': 31, 'marks': 104}

{'\_id': ObjectId('6520303cefcefe07f0fe38f2'), 'Name': 'Griffin', 'age': 23, 'marks': 107}

1. **Drop collection**

var = data.drop()

print("Dropped Collection")

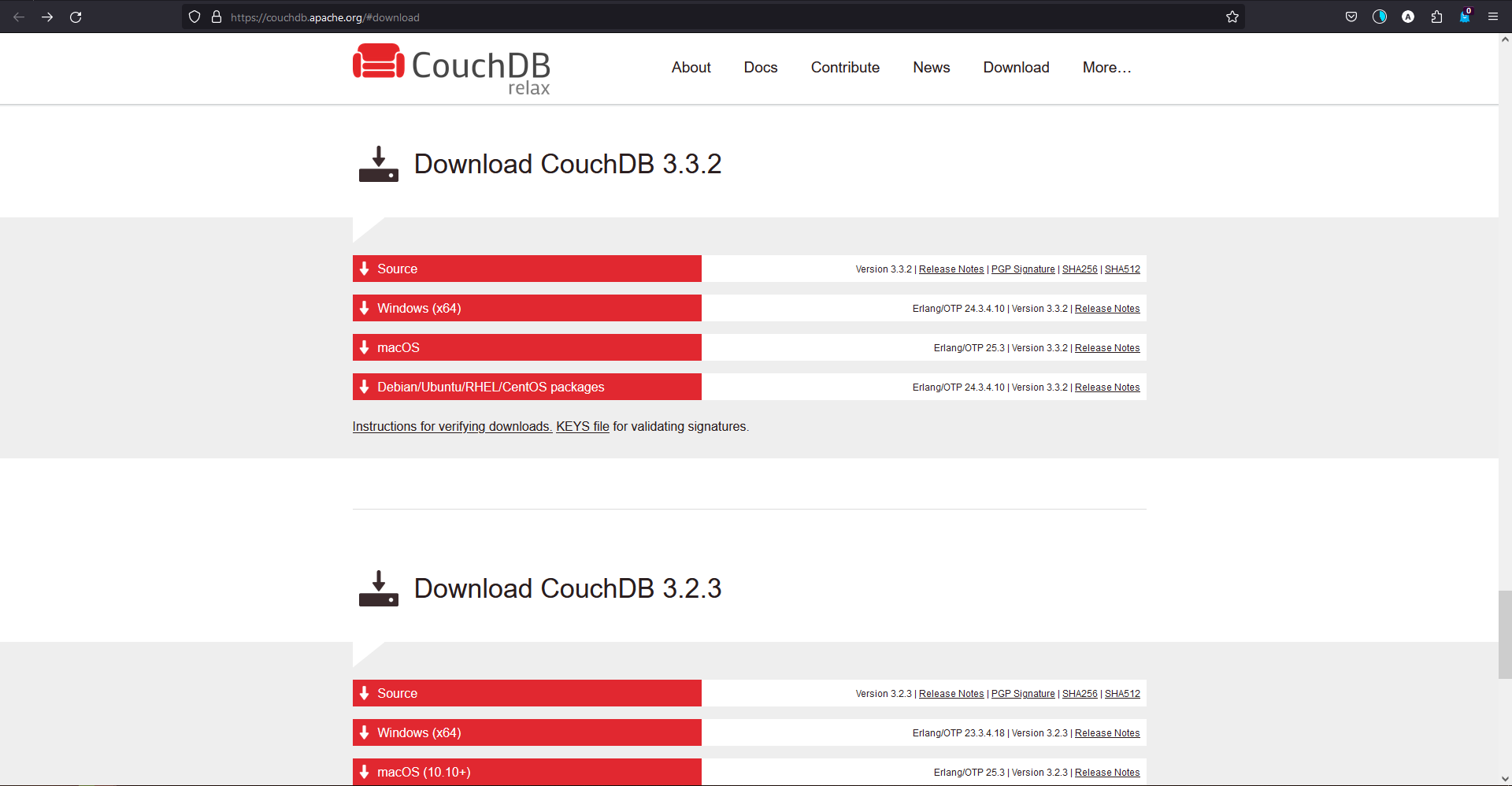
**Output**

Dropped Collection

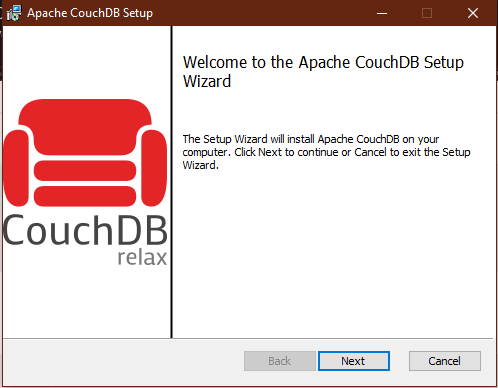
|  |  |
| --- | --- |
| **Practical No. 5** | |
| **Aim :** To Demonstrate Installation of CouchDB | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : | Sign : |

Procedure :

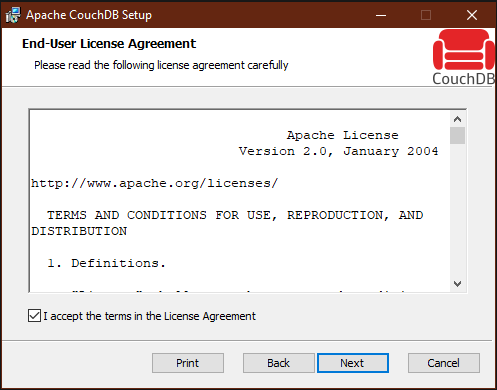
1. **Download CouchDB**:
   1. Go to the official CouchDB website to download the Windows installer:<https://couchdb.apache.org/#download>
   2. Choose the version you want to install and download the installer that matches your Windows version (32-bit or 64-bit).



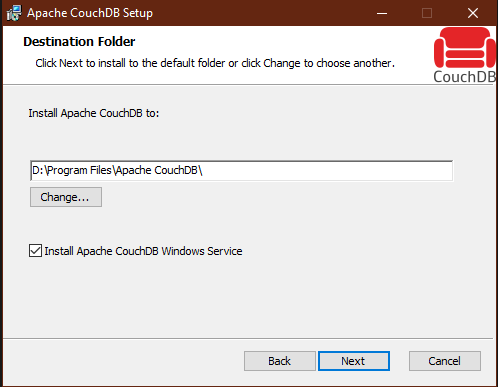
1. **Run the Installer**:
   1. Double-click the downloaded installer to run it.
   2. Click On Next



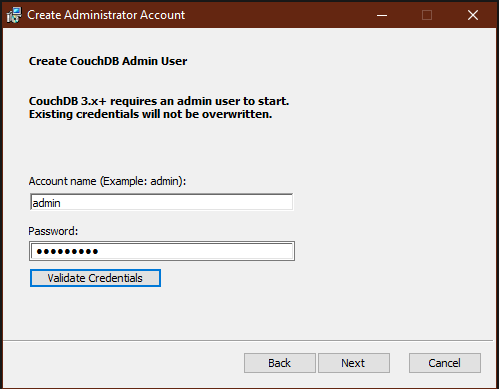
* 1. Click on accept and Next.



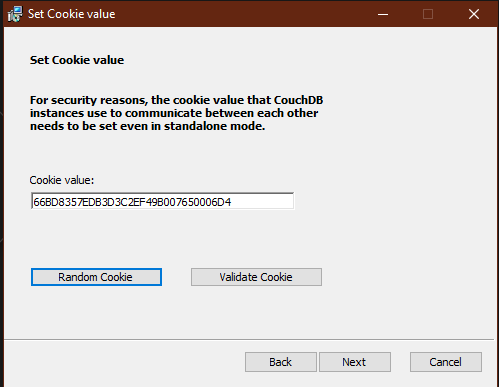
* 1. Select the desired location



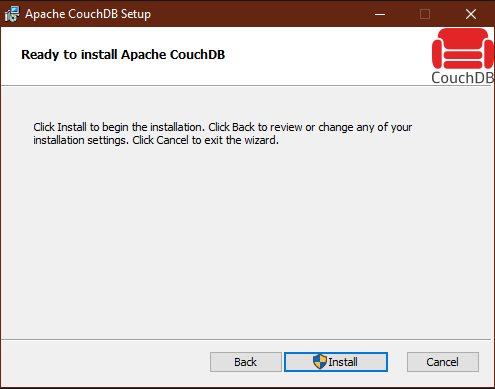
* 1. Create CouchDB admin credentials



* 1. Set the cookie value to a random



* 1. Click on install

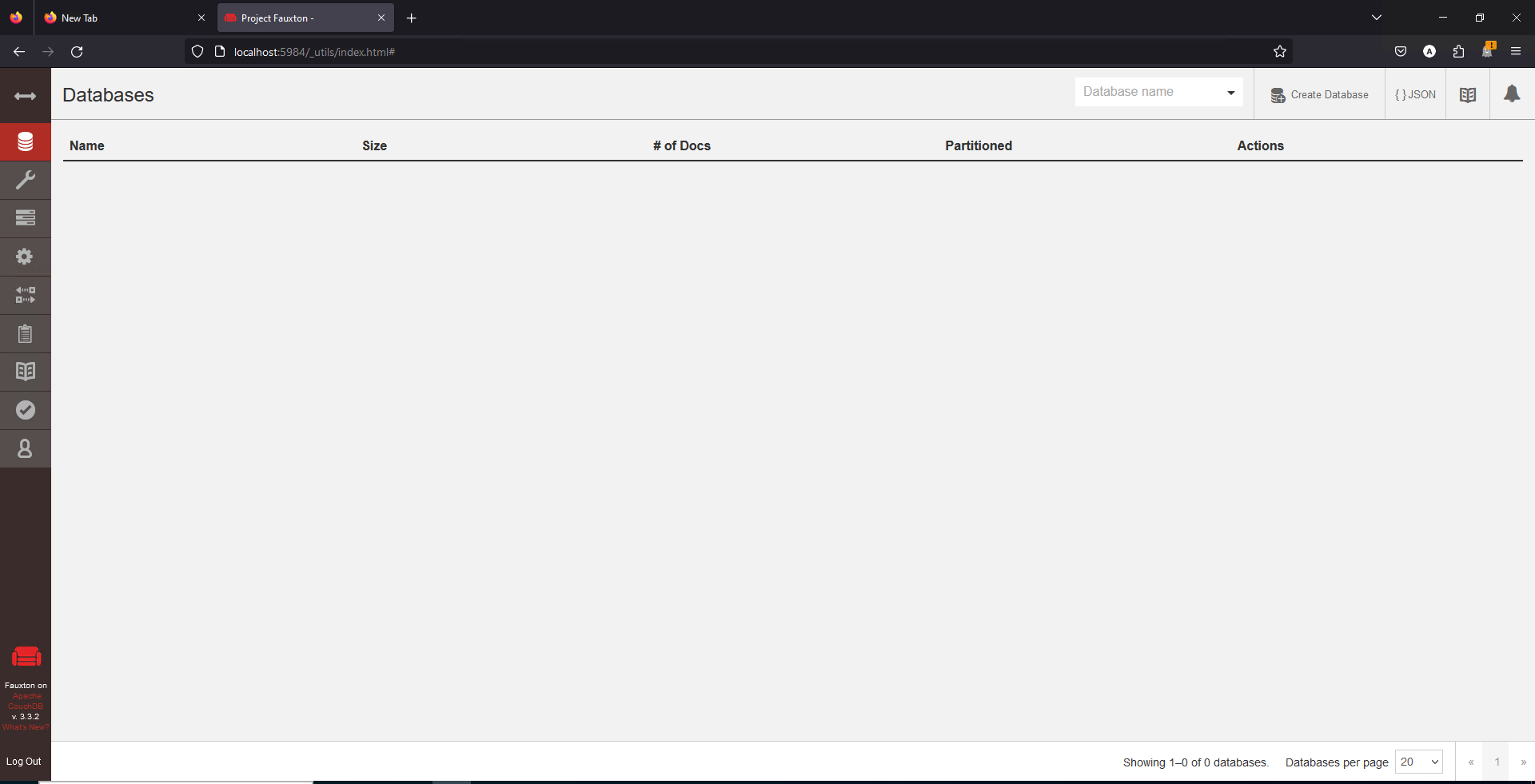


* 1. Once the installation is complete click on finish

1. **Configure CouchDB**:
   1. Now that you've successfully installed CouchDB, After installation, open built-in web interface of CouchDB by visiting the following link: <http://127.0.0.1:5984/>



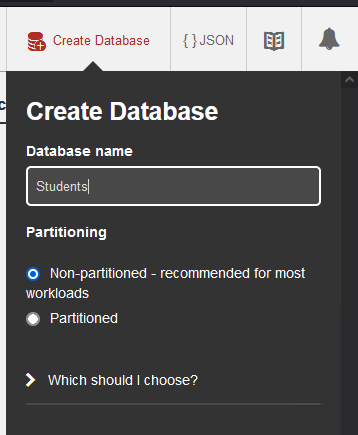
* 1. Open your web browser and enter the following URL: http://127.0.0.1:5984/\_utils/ . and login using the given credentials.



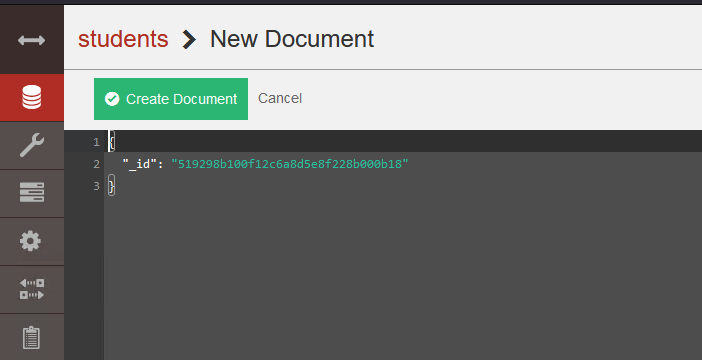
* 1. With this the installation of CouchDB has been successfully completed.

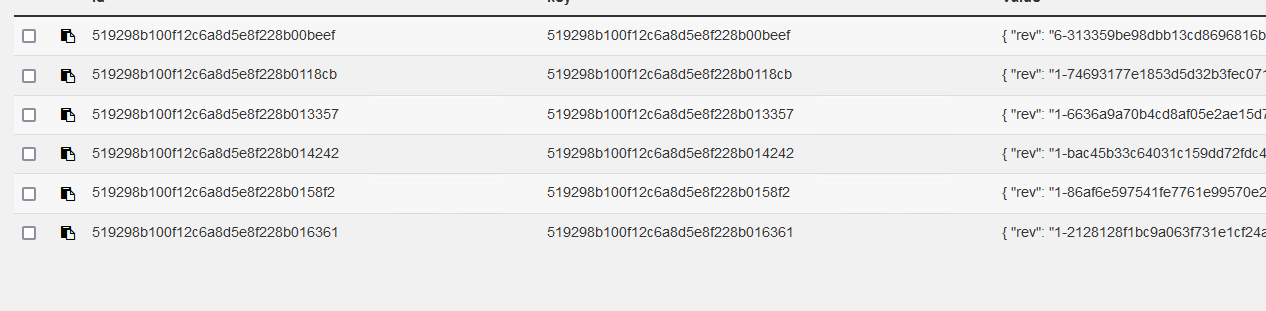
|  |  |
| --- | --- |
| **Practical No. 6** | |
| **Aim :** To Perform CRUD Operations in CouchDB. | |
| Name : Calvin Koshy | Roll No. : |
| Date : | Sign : |

1. Creating a database
   1. Click on the Create Database to manually create a database.

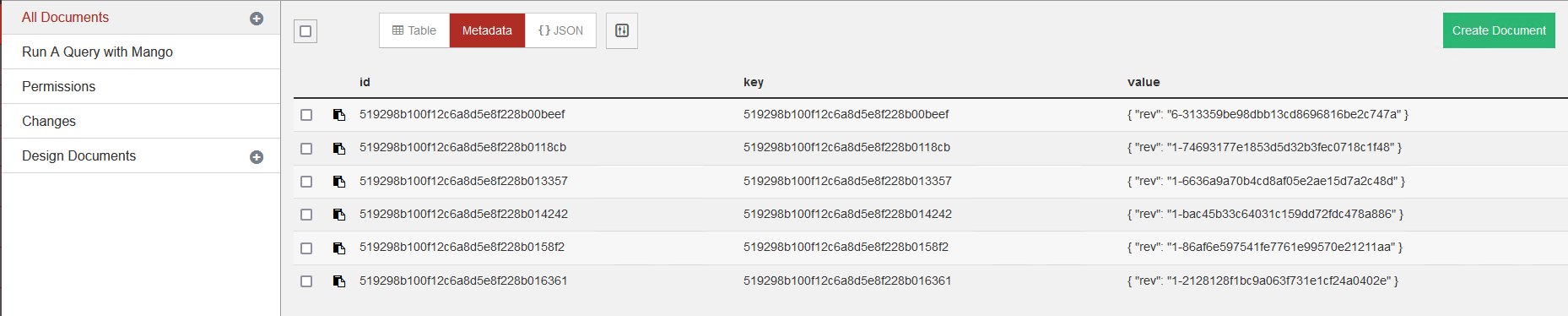


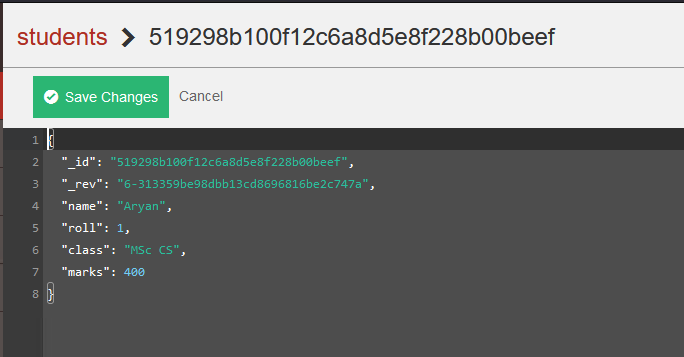
* 1. To Create a document click on the database you created and then click on “create database”



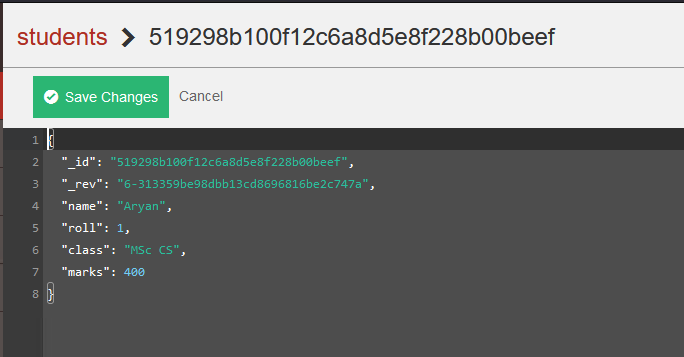


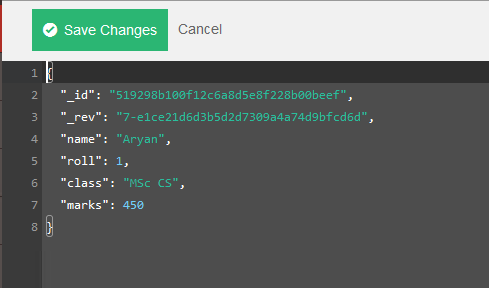
1. Reading database
   1. You can read the document just by clicking on it





1. Updating database
   1. To Change the values of the document you can open the document, do the changes and then hit save changes.

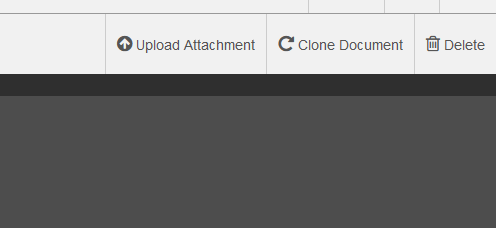


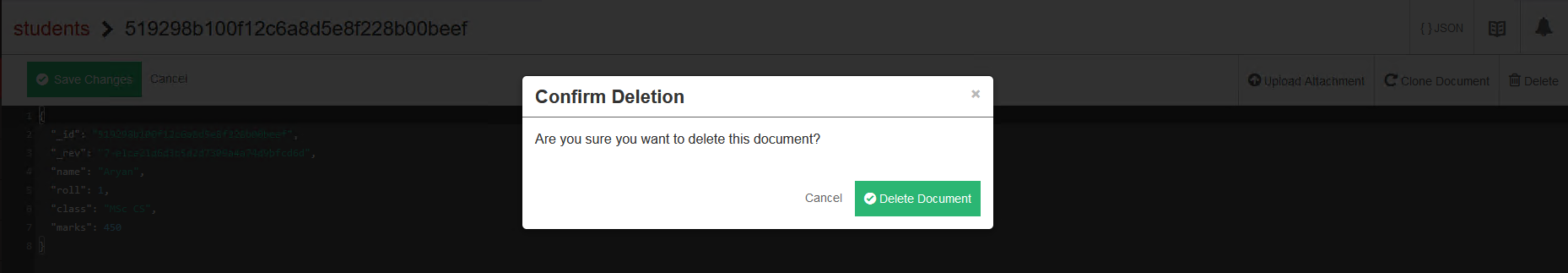


Here I changed the Marks from 400 to 450 simply by just editing the document.

* 1. You can notice that every time you update the document the rev value of the document also changes, The first number of rev indicates how many revisions there have been i.e. how many times it has been updated.

1. Deleting
   1. To delete a document you just have to click on delete on right side





|  |  |
| --- | --- |
| **Practical No. 7** | |
| **Aim :** To Create and Utilize Views in CouchDB. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : | Sign : |

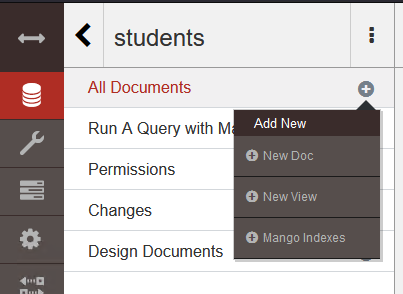
### Creating Views:

To Create View

Open the database

Click on the + next to design document

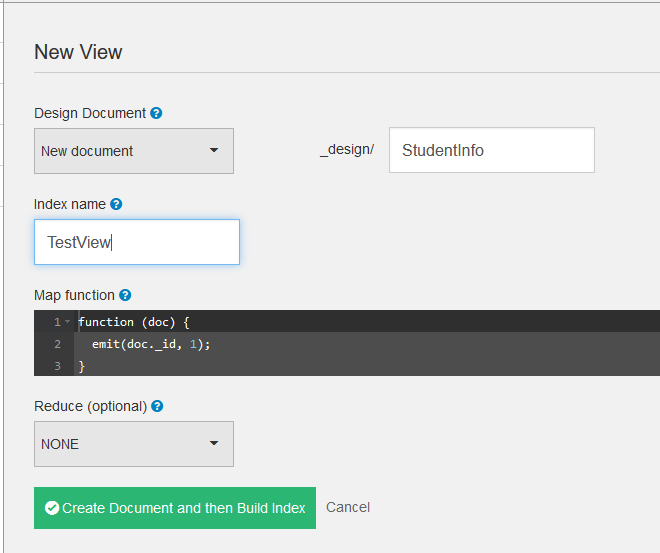
New View



you can add view to an existing design document or create a new one

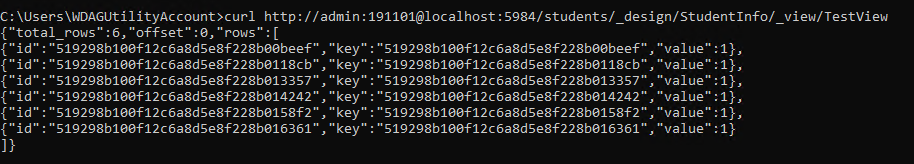
lets create a new one called: "StudentInfo"

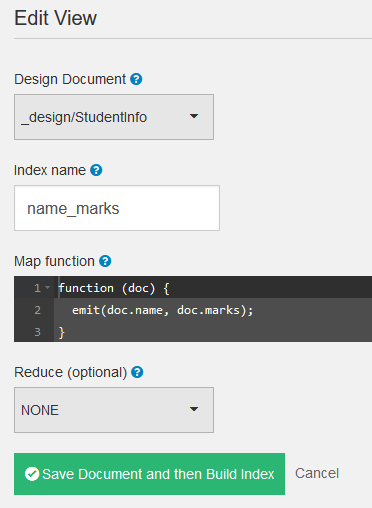
and view called: "TestView”



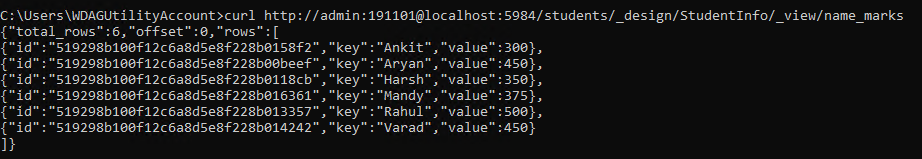
Viewing the created View

curl [http://admin:root@localhost:5984/students/\_design/StudentInfo/\_view/TestView](http://admin:191101@localhost:5984/students/_design/StudentInfo/_view/TestView)



We can create a view just to display the name and marks of the student

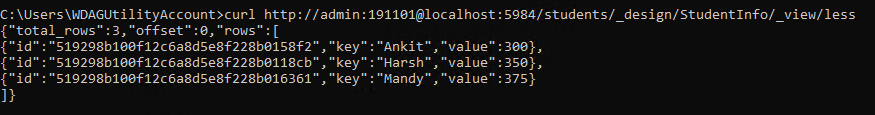
curl http://admin:root@localhost:5984/students/\_design/StudentInfo/\_view/name\_marks



Display Marks less than 450



curl http://admin:191101@localhost:5984/students/\_design/StudentInfo/\_view/less



|  |  |
| --- | --- |
| **Practical No. 8** | |
| **Aim :** HTTP URL Paths and Commands Using CURL to Interact with CouchDB for CRUD Operations. | |
| Name : Calvin Koshy | Roll No. : |
| Date : | Sign : |

1. **Check CouchDB Server Status:**

**curl** [**http://admin:root@localhost:5984/**](http://admin:root@localhost:5984/)

{"couchdb":"Welcome","version":"3.3.2","git\_sha":"11a234070","uuid":"ab8607b548150872f3fa8fe060005633","features":["access-ready","partitioned","pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"The Apache Software Foundation"}}

1. **Create Employee Doument :**

**>curl -X PUT http://admin:root@localhost:5984/emp/001 -d "{\"empid\":\"101\",\"empname\":\"Calvin\",\"empsal\":\"3000\"}"**

{"ok":true,"id":"001","rev":"1-ef6a085cd4f20d4f45f86ee7ada1ddef"}

**>curl -X PUT http://admin:root@localhost:5984/emp/002 -d "{\"empid\":\"106\",\"empname\":\"Durgesh\",\"empsal\":\"2200\"}"**

{"ok":true,"id":"002","rev":"1-9eb38ddee97f2d2abee32f7f9bf71906"}

**>curl -X PUT http://admin:root@localhost:5984/emp/003 -d "{\"empid\":\"107\",\"empname\":\"Apurva\",\"empsal\":\"7000\"}"**

{"ok":true,"id":"003","rev":"1-095b583759d43aad6f9ab295dff4d9f7"}

1. **List all Database**

**>curl -X GET** [**http://admin:root@localhost:5984/\_all\_dbs**](http://admin:root@localhost:5984/_all_dbs)

["emp","students"]

1. **Retrieve Employee Document by ID:**

**>curl -X GET http://admin:root@localhost:5984/emp/001**

{"\_id":"001","\_rev":"1-ef6a085cd4f20d4f45f86ee7ada1ddef","empid":"101","empname":"Calvin","empsal":"3000"}

**>curl -X GET** [**http://admin:root@localhost:5984/emp/003**](http://admin:root@localhost:5984/emp/003)

{"\_id":"003","\_rev":"1-095b583759d43aad6f9ab295dff4d9f7","empid":"107","empname":"Apurva","empsal":"7000"}

1. **List All Documents in the "emp" Database:**

**>curl -X GET** [**http://admin:root@localhost:5984/emp/\_all\_docs**](http://admin:root@localhost:5984/emp/_all_docs)

{"total\_rows":8,"offset":0,"rows":[

{"id":"001","key":"001","value":{"rev":"1-ef6a085cd4f20d4f45f86ee7ada1ddef"}},

{"id":"002","key":"002","value":{"rev":"1-9eb38ddee97f2d2abee32f7f9bf71906"}},

{"id":"003","key":"003","value":{"rev":"1-095b583759d43aad6f9ab295dff4d9f7"}},

{"id":"0905e7412c9a387b11e391481b0046ed","key":"0905e7412c9a387b11e391481b0046ed","value":{"rev":"1-76e095b14ae02d366ba51c828b3949e8"}},

{"id":"0905e7412c9a387b11e391481b00801f","key":"0905e7412c9a387b11e391481b00801f","value":{"rev":"1-c8163d1eb6c80da62e06cd946b50548d"}},

{"id":"0905e7412c9a387b11e391481b009bce","key":"0905e7412c9a387b11e391481b009bce","value":{"rev":"1-28de6d585b44ead2b7a806c0fb4c513a"}},

{"id":"0905e7412c9a387b11e391481b00c5aa","key":"0905e7412c9a387b11e391481b00c5aa","value":{"rev":"1-5cc9484efa44604805938b87b498b8cf"}},

{"id":"0905e7412c9a387b11e391481b00dd85","key":"0905e7412c9a387b11e391481b00dd85","value":{"rev":"1-f6f7719e67447f26ca0588aa09d46fe9"}}

]}

1. **Delete Employee Document with Revision:**

**>curl -X DELETE http://admin:root@localhost:5984/emp/003?rev=1-095b583759d43aad6f9ab295dff4d9f7**

{"ok":true,"id":"003","rev":"2-c3e9b92565857c3e6bbdec062002b470"}

**>curl -X GET http://admin:root@localhost:5984/emp/\_all\_docs**

{"total\_rows":7,"offset":0,"rows":[

{"id":"001","key":"001","value":{"rev":"1-ef6a085cd4f20d4f45f86ee7ada1ddef"}},

{"id":"002","key":"002","value":{"rev":"1-9eb38ddee97f2d2abee32f7f9bf71906"}},

{"id":"0905e7412c9a387b11e391481b0046ed","key":"0905e7412c9a387b11e391481b0046ed","value":{"rev":"1-76e095b14ae02d366ba51c828b3949e8"}},

{"id":"0905e7412c9a387b11e391481b00801f","key":"0905e7412c9a387b11e391481b00801f","value":{"rev":"1-c8163d1eb6c80da62e06cd946b50548d"}},

{"id":"0905e7412c9a387b11e391481b009bce","key":"0905e7412c9a387b11e391481b009bce","value":{"rev":"1-28de6d585b44ead2b7a806c0fb4c513a"}},

{"id":"0905e7412c9a387b11e391481b00c5aa","key":"0905e7412c9a387b11e391481b00c5aa","value":{"rev":"1-5cc9484efa44604805938b87b498b8cf"}},

{"id":"0905e7412c9a387b11e391481b00dd85","key":"0905e7412c9a387b11e391481b00dd85","value":{"rev":"1-f6f7719e67447f26ca0588aa09d46fe9"}}

]}

1. **Update Employee Document with Revision:**

**>curl -X PUT http://admin:admin@localhost:5984/emp/001 -d "{\"empid\":\"101\",\"empname\":\"Calvin\",\"salary\":\"5000\",\"yop\":\"5\",\"\_rev\":\"1-ef6a085cd4f20d4f45f86ee7ada1ddef\"}"**

{"ok":true,"id":"001","rev":"2-3af8394d4fef862b8ecfa245867e1075"}

**>curl -X PUT http://admin:root@localhost:5984/emp/002 -H "Content-Type: application/json" -d "{\"empid\":\"102\",\"empname\":\"Sahil\",\"salary\":\"2000\",\"yop\":\"5\",\"\_rev\":\"1-9eb38ddee97f2d2abee32f7f9bf71906\"}"**

{"ok":true,"id":"002","rev":"2-edf4f3d73dee370083bef1917a37895c"}

|  |  |
| --- | --- |
| **Practical No. 9 A** | |
| **Aim :** To demonstrate Redis installation and execute its commands to create data. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : | Sign : |

There are two ways of installing Redis on windows

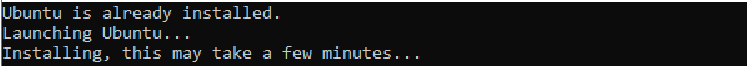
1. A Recommended method which suggests Running redis of WSL (Windows subsystem for linux) as Redis does not support windows officially.
2. A method that’s relatively easy but is not recommended , Which is directly downloading redis on windows through a windows port for Redis.
3. First Method:
   1. Install WSL command :
      1. You can now install everything you need to run WSL with a single command. Open PowerShell or Windows Command Prompt in administrator mode by right-clicking and selecting "Run as administrator", enter the wsl --install command, then restart your machine.

**wsl --install**

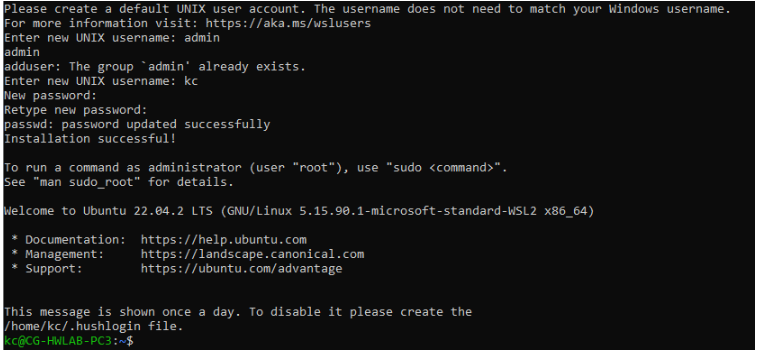
This should take a while , restart the system after completion of

installation.

* 1. After restarting the system :
     1. You will get this pop up window on boot, Where Ubuntu is launched



* + 1. You will be asked to create a new user login:



* + 1. Now with WSL installed , we start with redis installation, add these commands

**curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o**

**/usr/share/keyrings/redis-archive-keyring.gpg**

**echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg]**

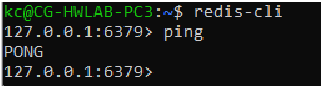
**https://packages.redis.io/deb $(lsb\_release -cs) main" | sudo tee**

**/etc/apt/sources.list.d/redis.list**

* + 1. Now we with redis installed, we start redis server

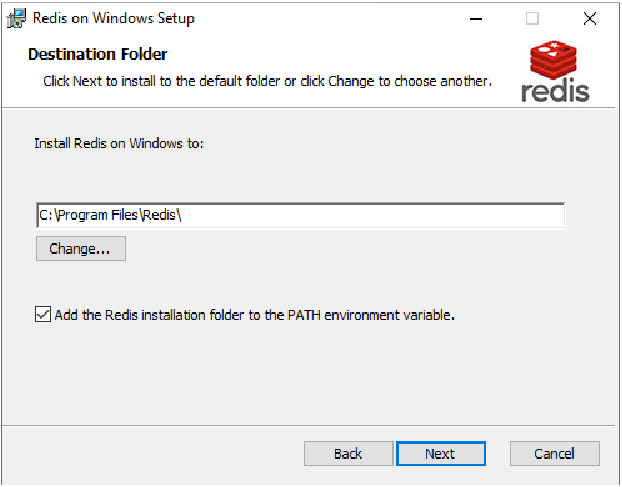
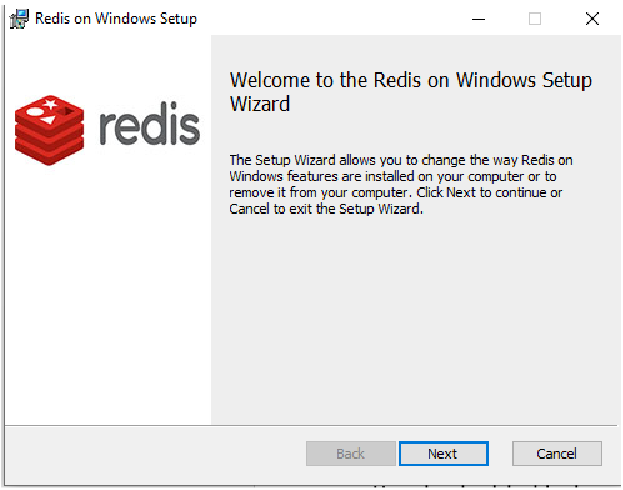


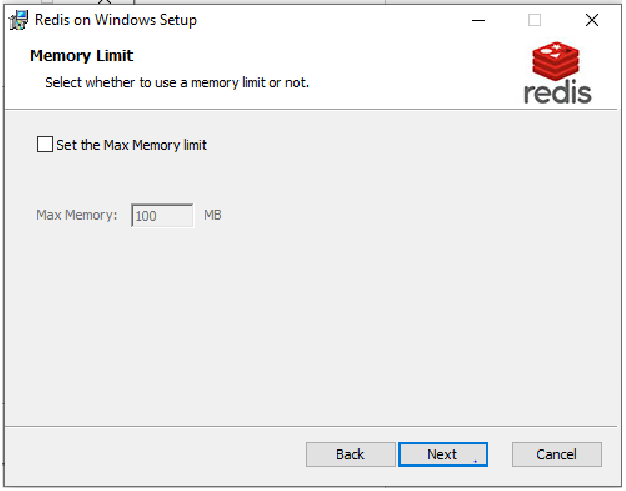
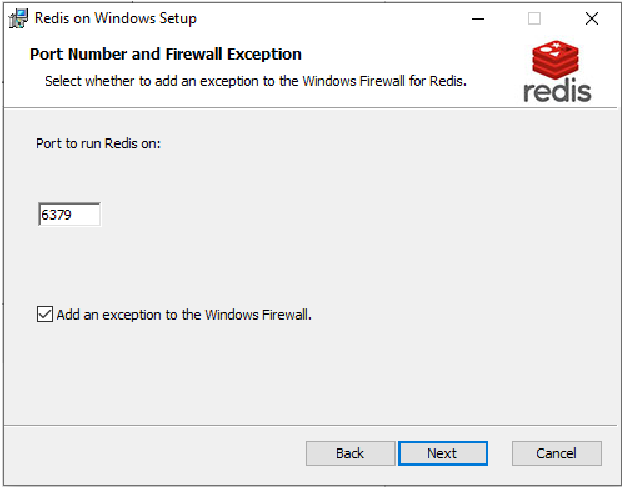
* + 1. Let’s get into redis command line prompt

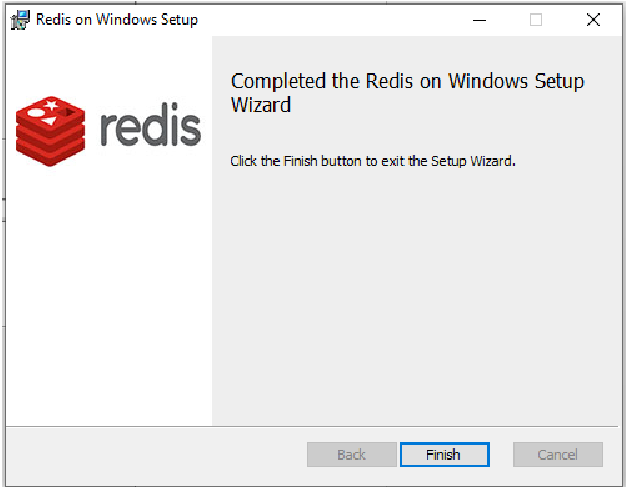


* + 1. This is how you set up redis on window

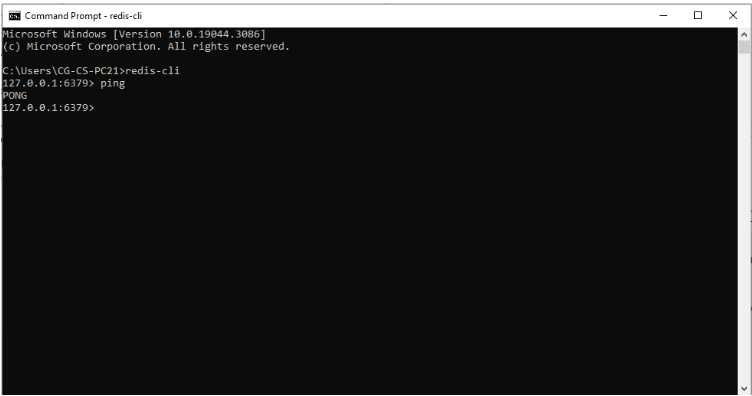
1. Second Method :
   1. You simply visit this site Release 3.0.504 · microsoftarchive/redis · GitHub , download the msi version, you get this window







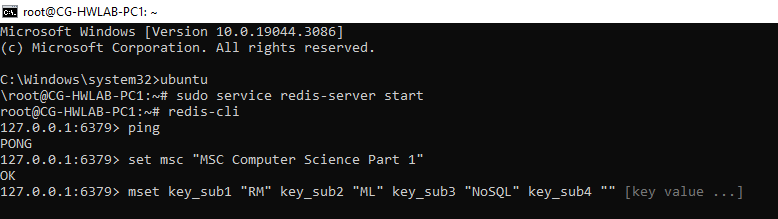
* 1. In your Command prompt



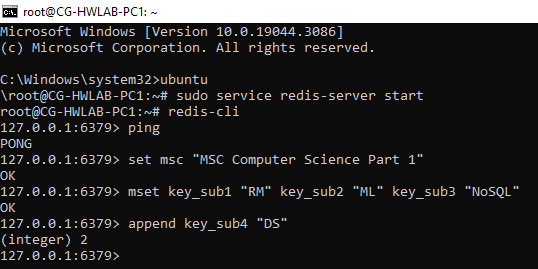
* 1. With this redis has been successfully installed

|  |  |
| --- | --- |
| **Practical No. 9 B** | |
| **Aim :** To demonstrate Redis installation and execute its commands to create data. | |
| Name : Calvin Koshy | Roll No. : |
| Date : | Sign : |

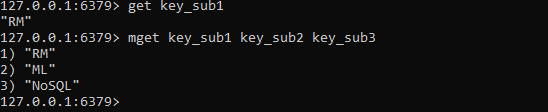
1. Create String using “set”



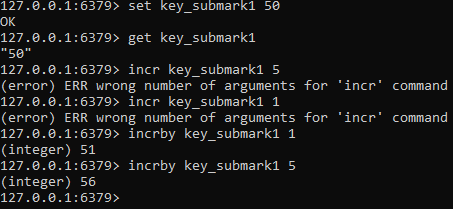
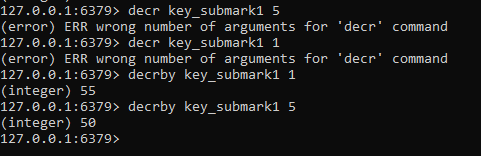
1. Using “append”



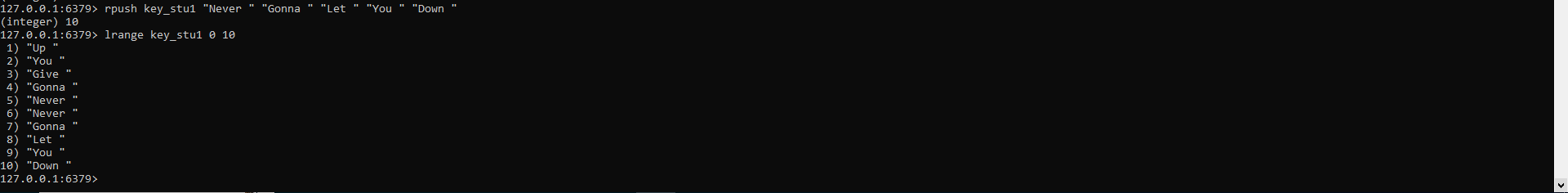
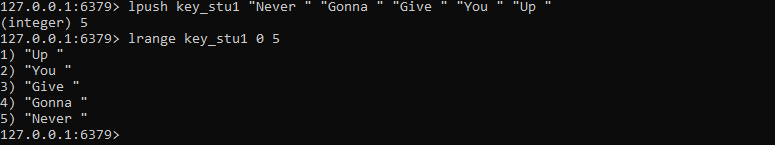
1. Retrieve String using “get” and “mget”



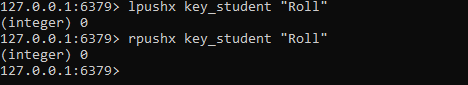
1. Manipulating Strings “incr”/”incrby” and “decr”/”decrby”



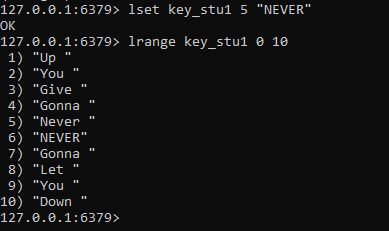
1. Creating Lists “lpush”/”rpush”



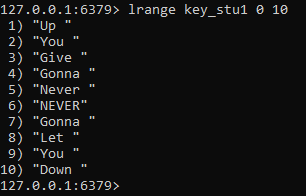
1. Using “lpushx” and ”rpushx”



1. Change existing elements in list “lset”



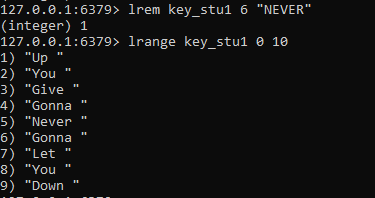
1. Retriever elements from list “lrange”



1. Total length of the list “llen”



1. Removing elements from the list “lrem”



1. Creating Hashes “hset”



1. Set Multiple fields “hsetnx”



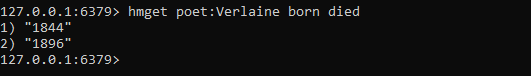
1. Retrieving information from hash



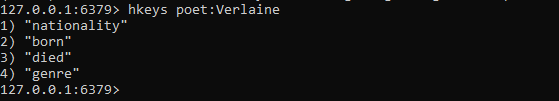
1. Return a field value “hget”



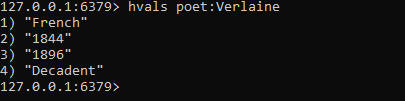
1. Return multiple fields “hmget”



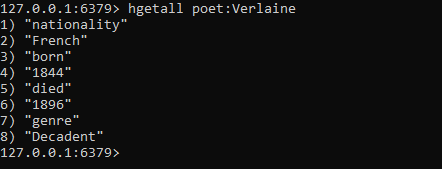
1. Obtain a list “hkeys”



1. Retrieve a list “hvals”



1. Return a associated value in a list “hgettall”



1. Find Length of a field “hlen”



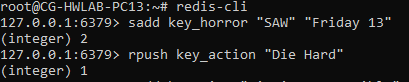
1. Find length of a hash string “hstrlen”



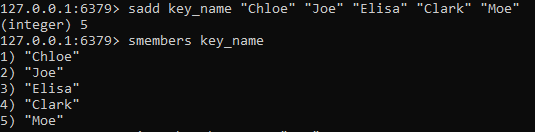
1. Removing fields from hashes ”hdel”



1. Creating Sets



1. Retrieving Members from Sets



1. Checking if a specific value already exists “sismember”



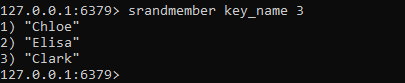
1. Check all the members in set “scard”



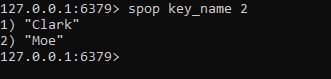
1. Return random elements to set “srandmember”



1. Return Multiple random elements “srandmember key (Integer)”



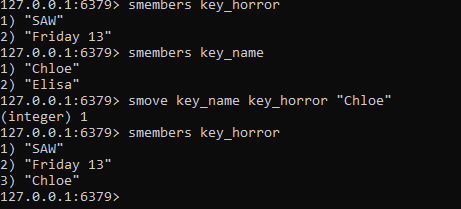
1. Removing Members from sets “spop”



1. Removing a specific member “srem”



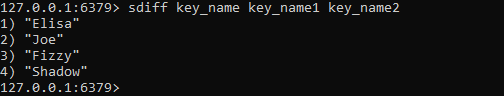
1. Move a member from one list to another “smove”



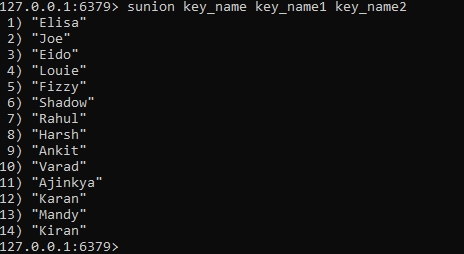
1. Comparing sets “sinter”



1. Set differences “sdiff”



1. Overwrite the destination key “sunion”



1. Creating sorted set and adding members “zadd”



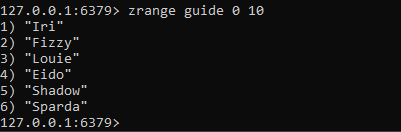
1. Add multiple members



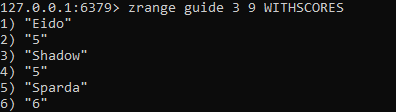
1. Increment Score of member “zincrby”



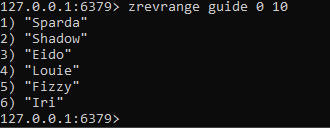
1. Retrieve Members from Sorted set “ zrange”



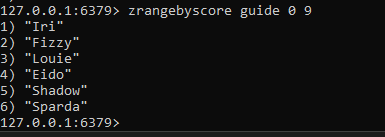
1. Return member and score “zrange key (Int) (Int) WITHSCORE”



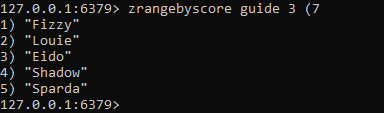
1. Arrange in descending order “zrevrange”



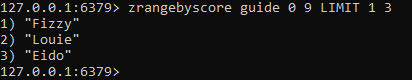
1. Return members based on Scores “zrangebyscore”



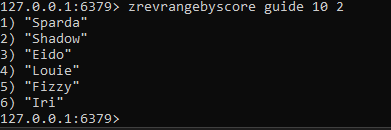
1. Return member with greater or less than score “zrangebyscore key (Int) )/( (Int)”



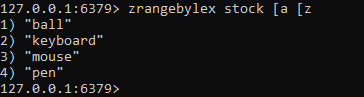
1. Using Limit Options “zrangebyscore key (Int) (Int) LIMIT (Int) (Int)”



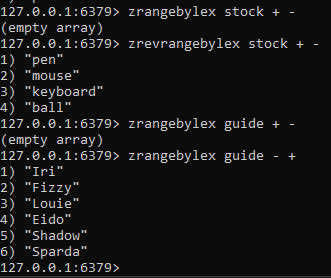
1. Return the Member in reverse “zrevrangebyscore”



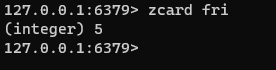
1. Sorting by alphabetical order “zrangebylex key [a [z”



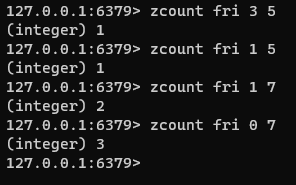
1. Using special characters + and - “zrangebylex key + -”



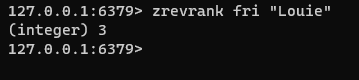
1. Retrieving information about Sorted set “zcard”



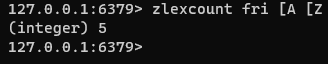
1. Retrieving how many elements are held “zcount”



1. Reverse the rank of the members “zrevrank”



1. Retrieve how many members are held in a sorted set “zlexcount”



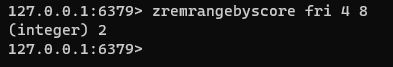
1. Removing Members from sorted sets “zrem”



1. Remove multiple Members “zremrangebylex key [A [Z”



1. Remove members according to range of score “zremrangebyscore key 4 6”



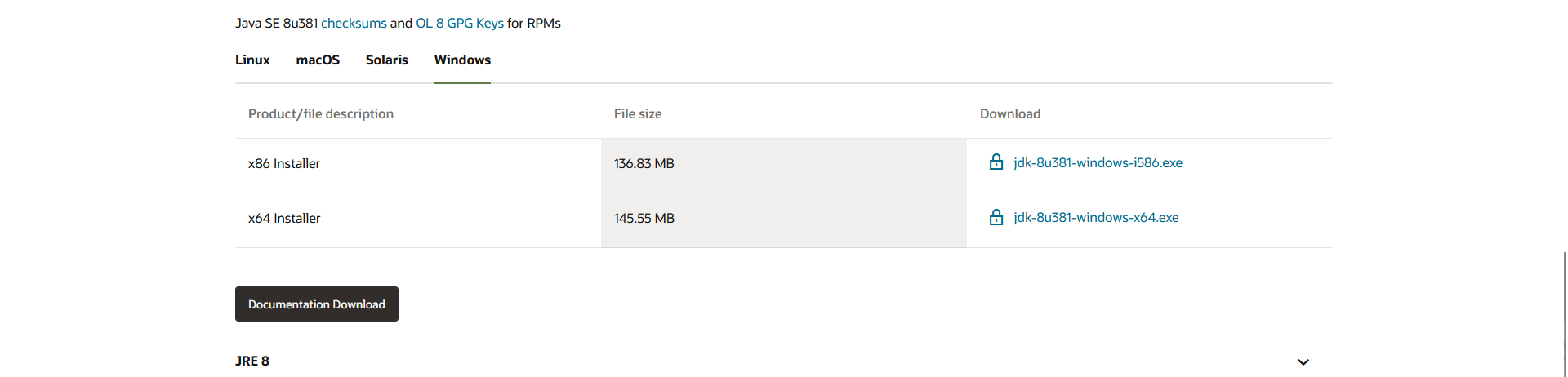
1. Remove members based on range “zremrangebyrank key 0 4”\



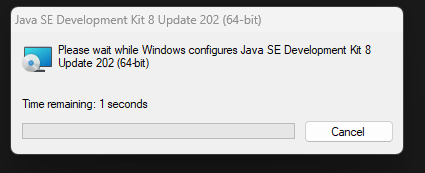
|  |  |
| --- | --- |
| **Practical No. 10 A** | |
| **Aim :** To demonstrate cassandra installation and execute queries to create column family data to perform CURD operations. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : | Sign : |

Requirements : java 8 and Python2.7

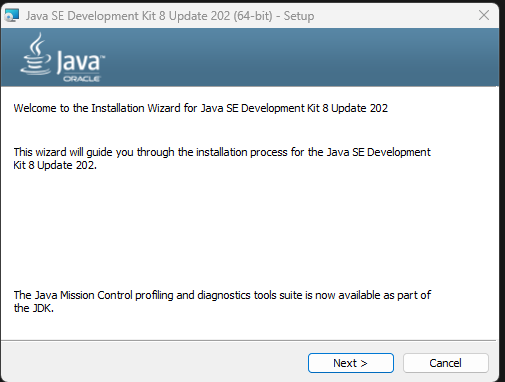
1. Installation of java 8 :
   1. Go to official oracle website <https://www.oracle.com/java/technologies/downloads/#java8-windows>



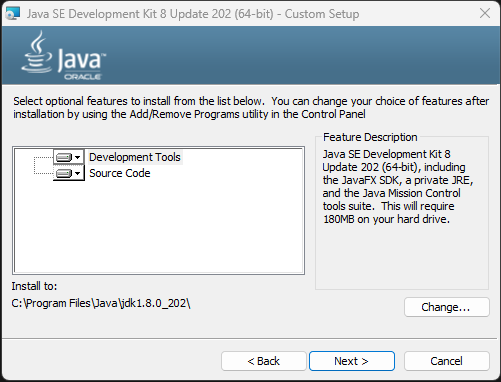
* 1. Run the launcher



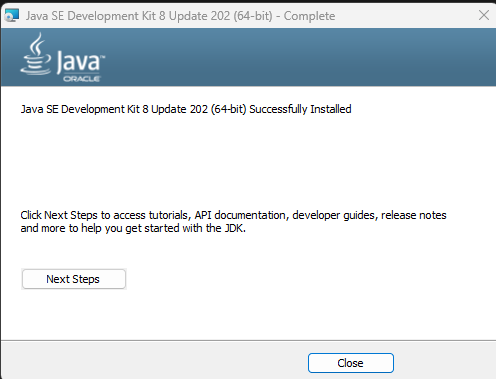
* 1. Select Next



* 1. Select the designated Drive

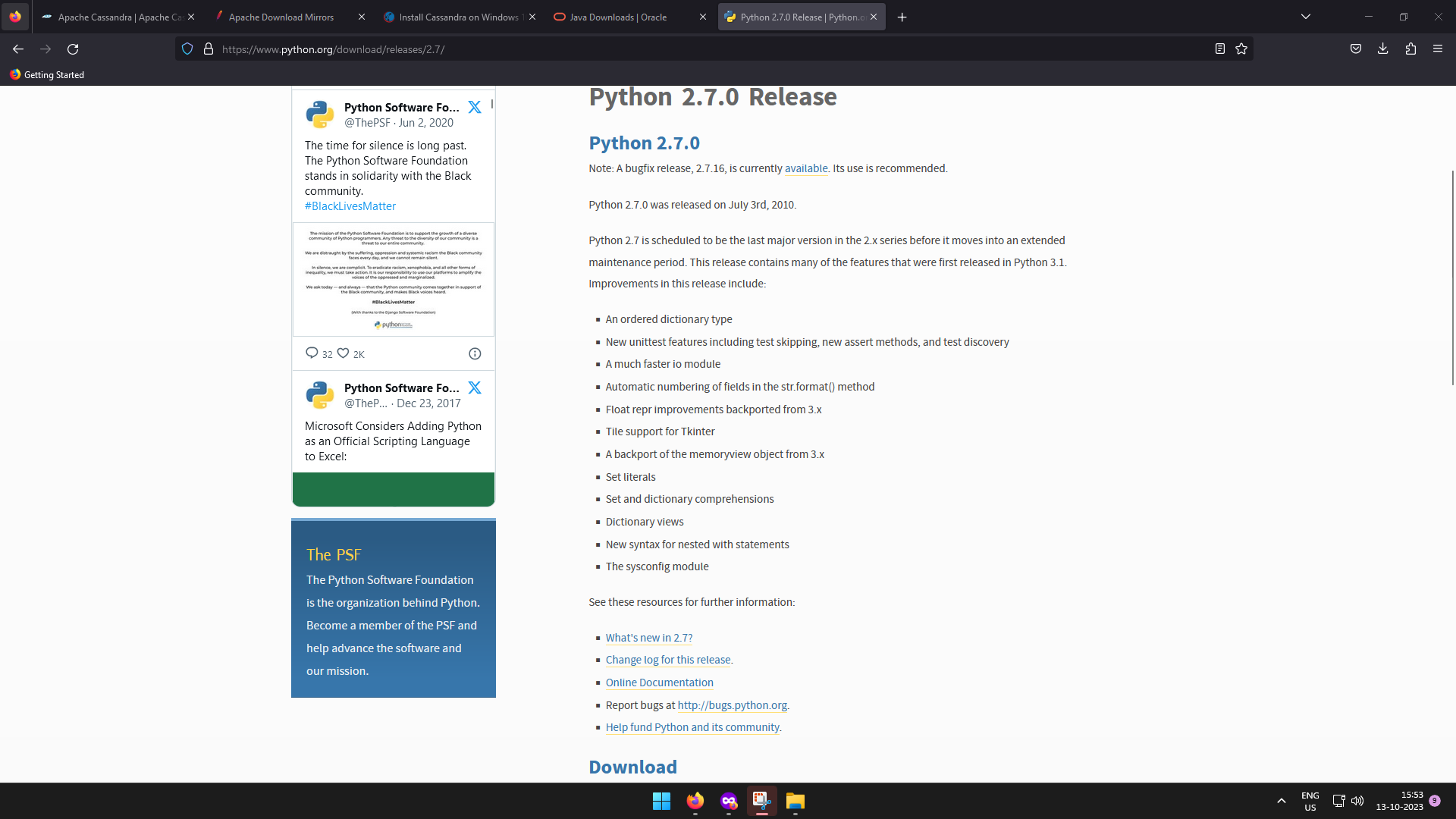


* 1. Once the installation is finished click on Close

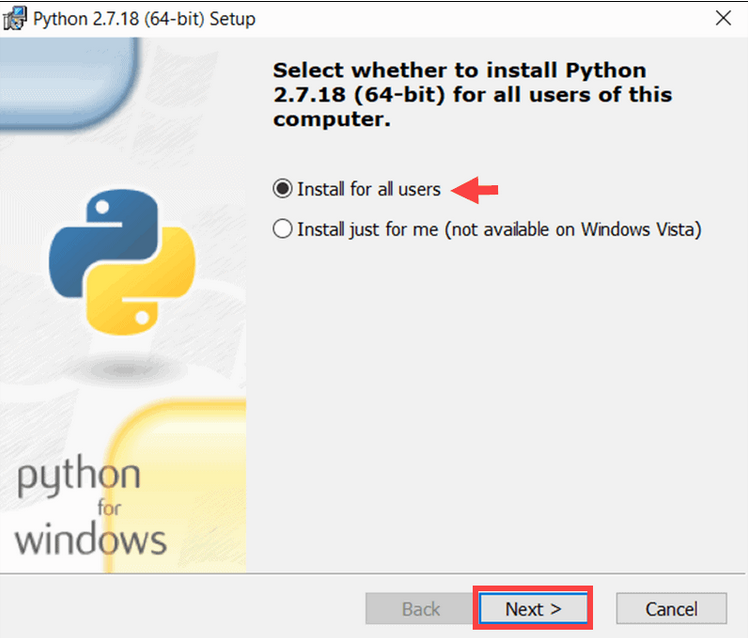


* 1. Java has been successfully installed

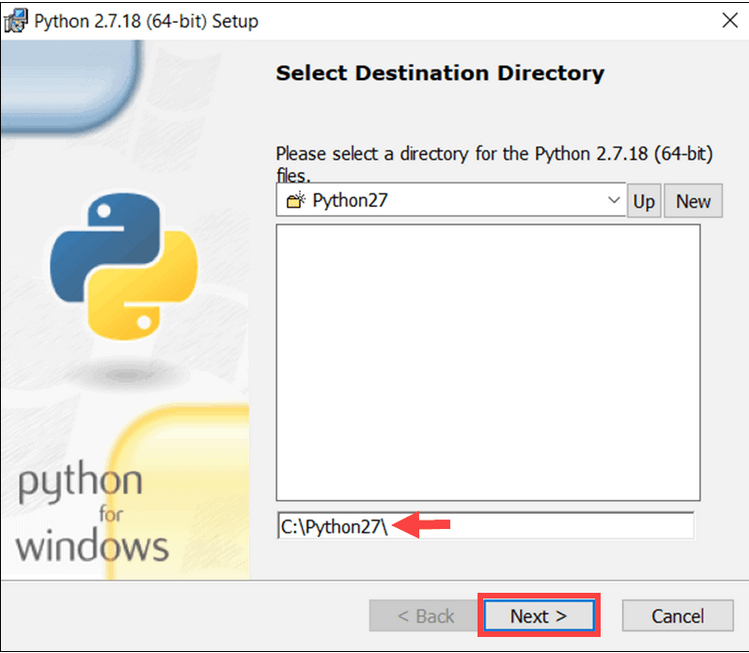
1. Installing Python 2.7
   1. Visit the official python website <https://www.python.org/download/releases/2.7/>



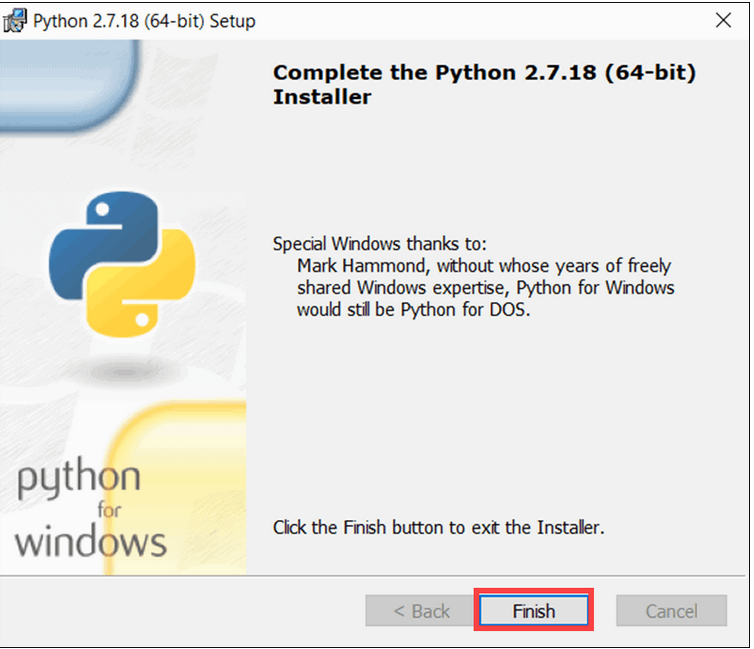
* 1. Proceed with the installation of python 2.7



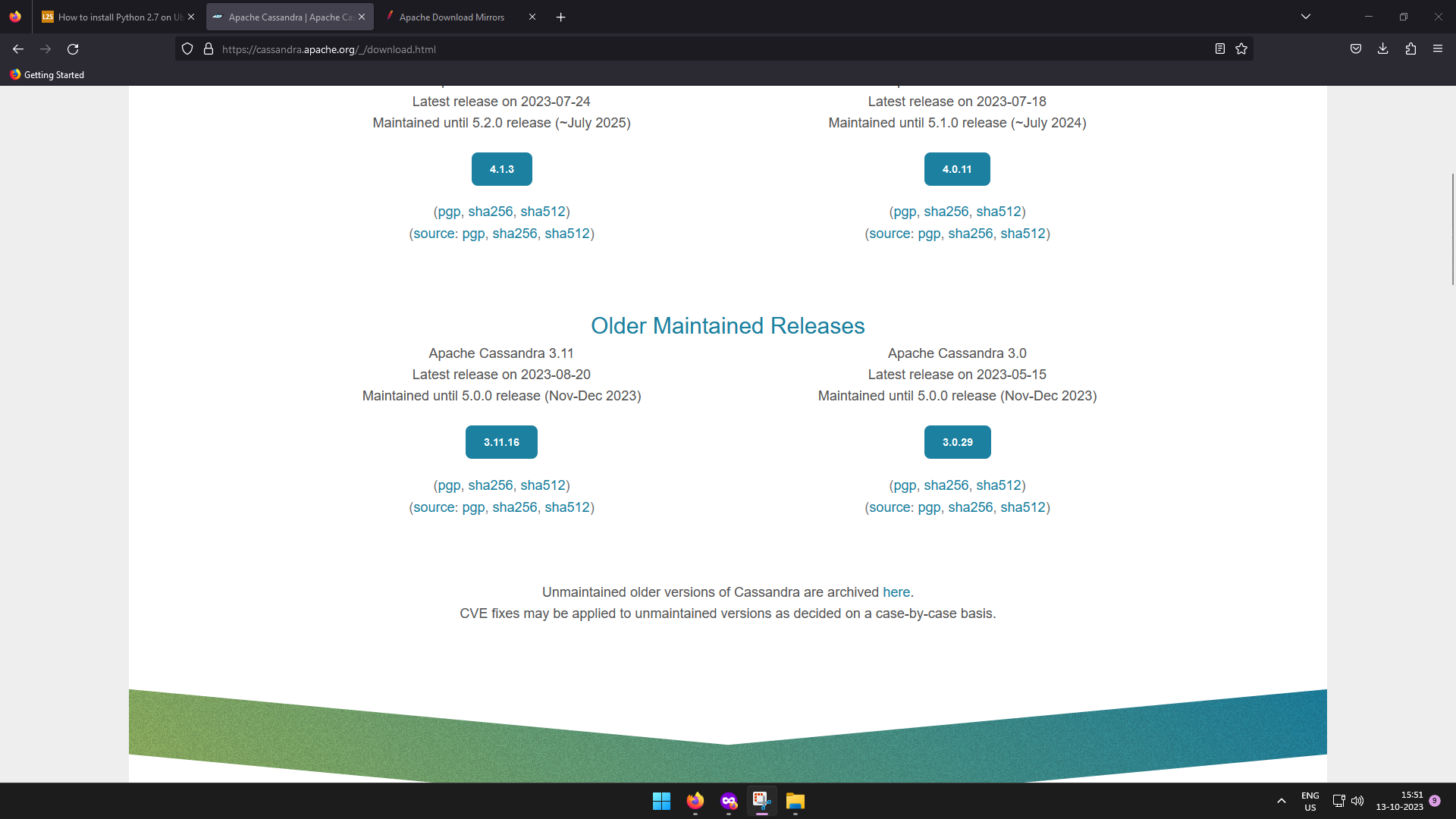
* 1. Select the directory



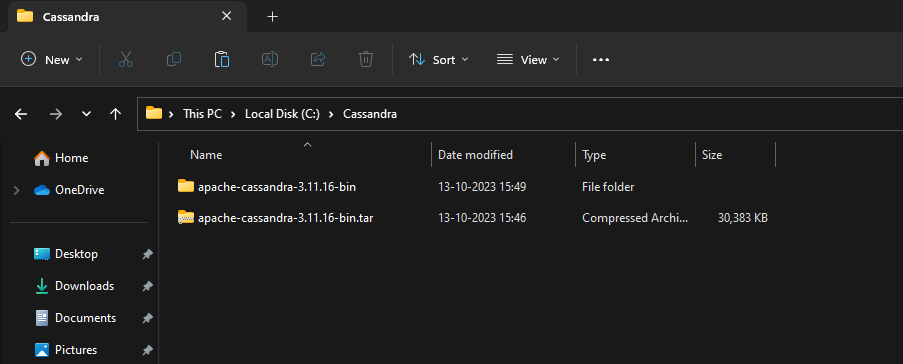
* 1. Once the installation is complete click on Finish



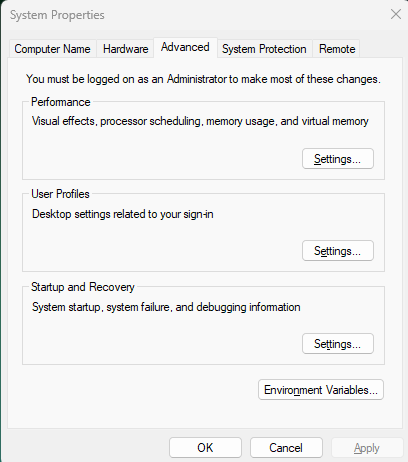
1. Downloading Cassandara 3.11.6
   1. Cassandra is available for download on <https://cassandra.apache.org/_/download.html>



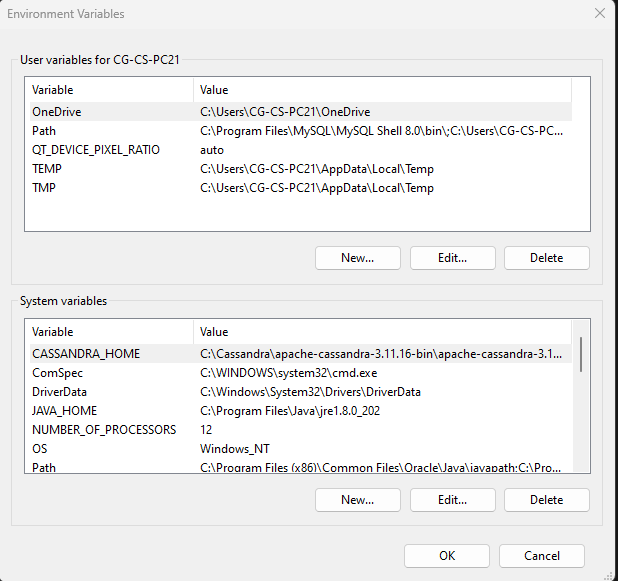
* 1. Extract Cassandra in the same drive



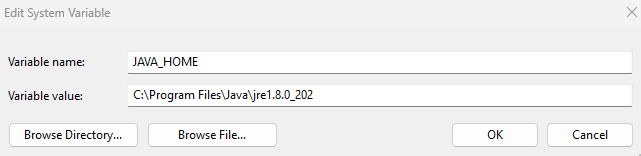
1. Setting up the environment of Java, Python and Cassandra
   1. Go to system properties > Advanced > Environment Variables



* 1. Set up your Environment path



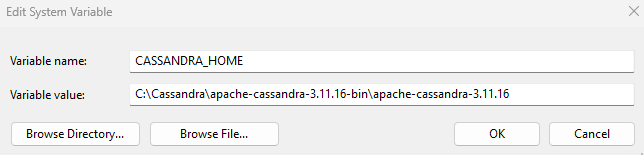
* 1. For java 8



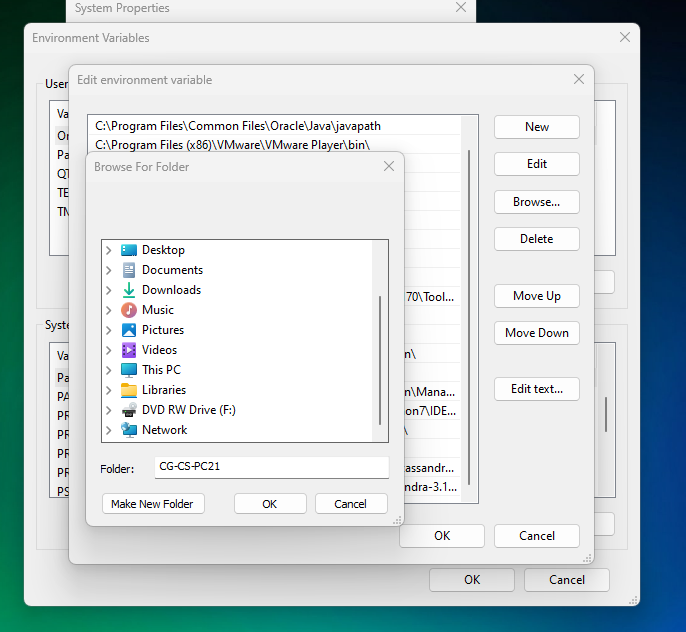
* 1. For Python 2.7



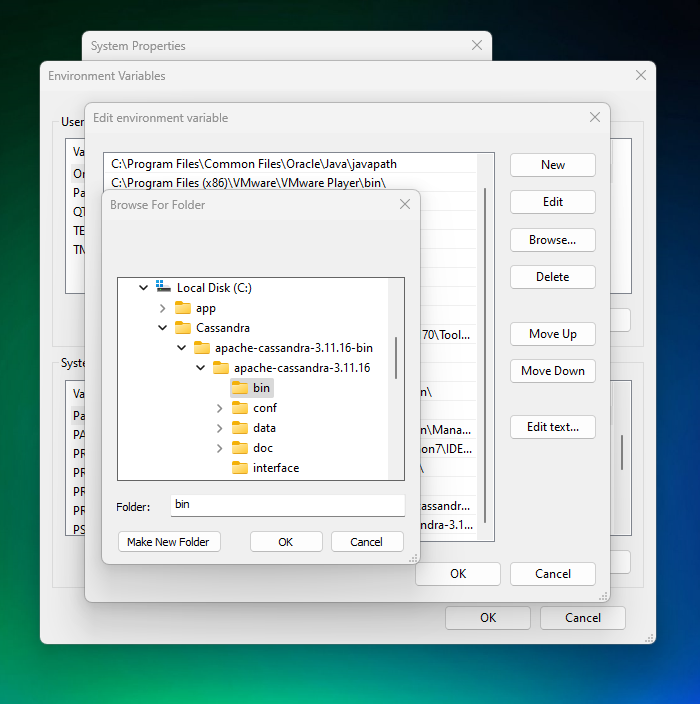
* 1. For cassandra



Go to %Path%

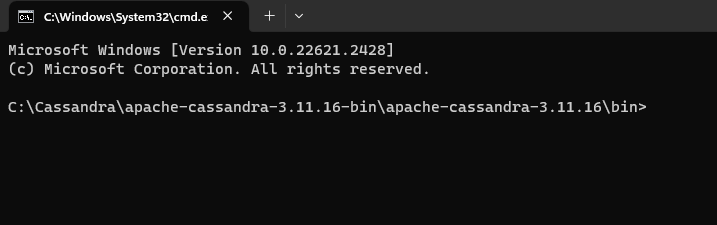


* 1. Browse to your Cassandra folder

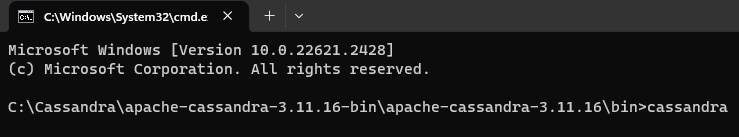


* 1. Select the bin folder and click on ok
  2. Your are done setting up the PATH for all the Softwares required

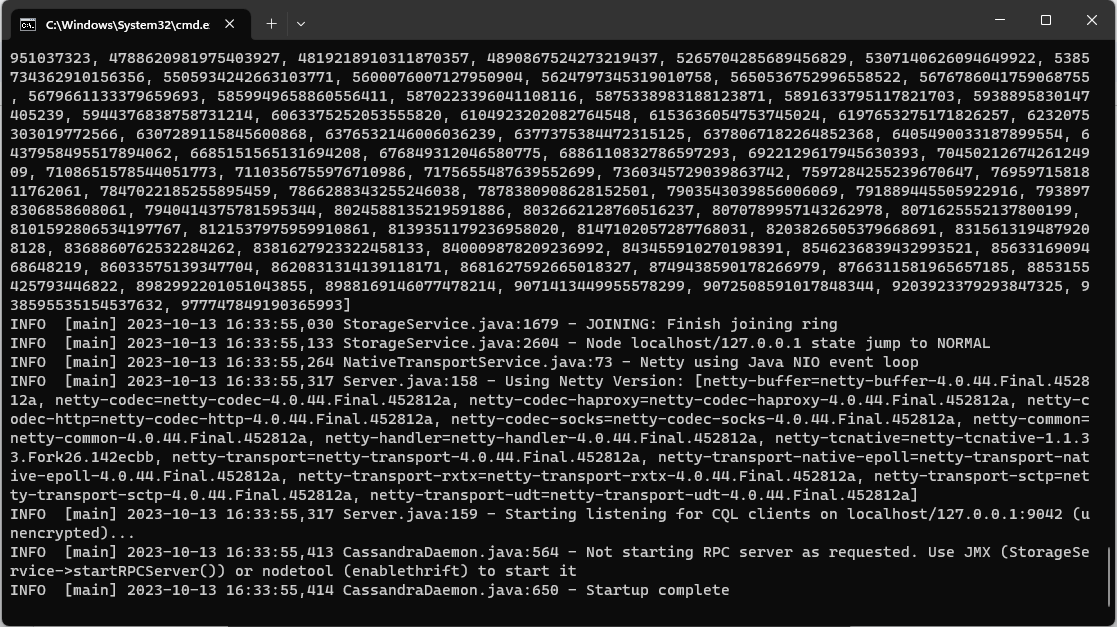
1. Open CMD
   1. Go to your Cassandra folder



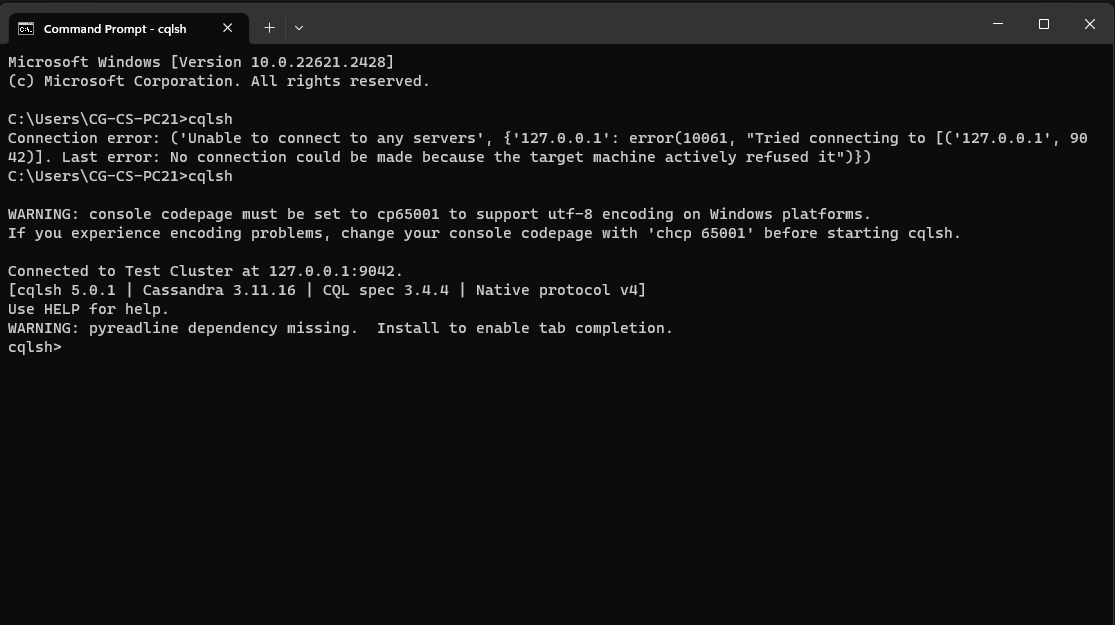
* 1. Type cassandra



* 1. Wait for the Command to say Startup complete



* 1. Type cqlsh in new Command Prompt



|  |  |
| --- | --- |
| **Practical No. 10 B** | |
| **Aim :** To demonstrate cassandra installation and execute queries to create column family data to perform CURD operations. | |
| Name : Calvin Koshy | Roll No. : KFPMSCCS005 |
| Date : | Sign : |

Create a keyspace;

cqlsh> CREATE KEYSPACE msccs WITH replication = {'class':'SimpleStrategy', 'replication\_factor' : 1};

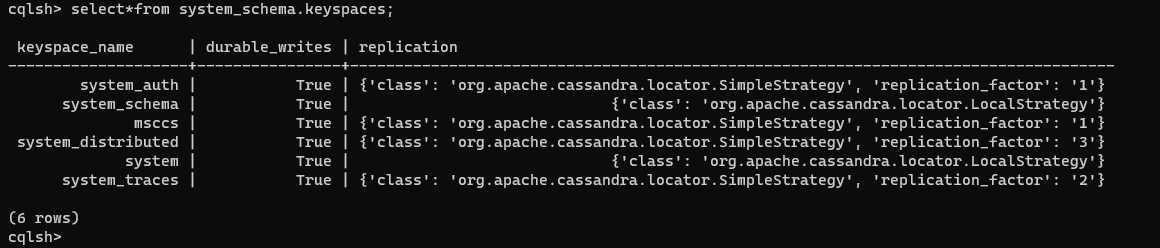
cqlsh> DESCRIBE KEYSPACES;

msccs system\_auth system\_distributed mallika

system\_schema system system\_traces

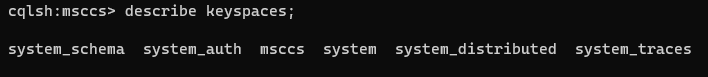
Show the keyspace

select\*from system\_schema.keyspaces;



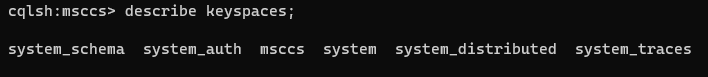
Select keyspace

use msccs;

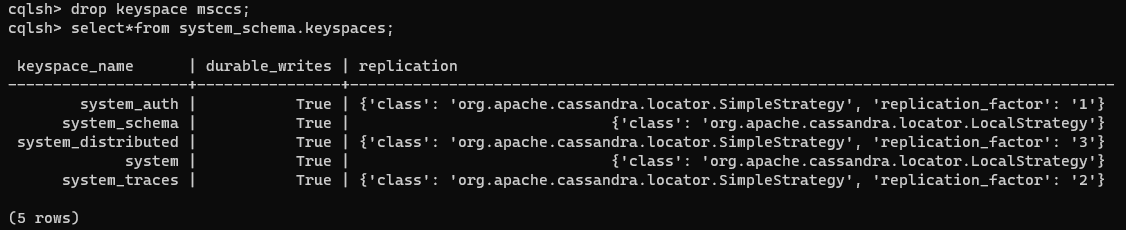


Describe keyspace

describe keyspaces;



Drop keyspace



Create Table

CREATE TABLE report(

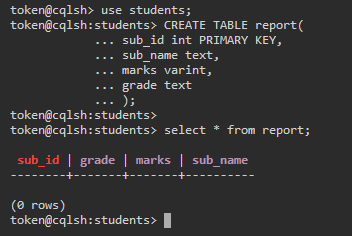
... sub\_id int PRIMARY KEY,

... sub\_name text,

... marks varint,

... grade text

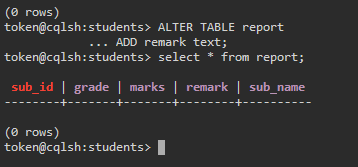
... );



Altering Table

ALTER TABLE report

... ADD remark text;



Dropping column from a table

ALTER TABLE report

... DROP remark;



**CRUD Operations:**

**Creating Data in Table:**

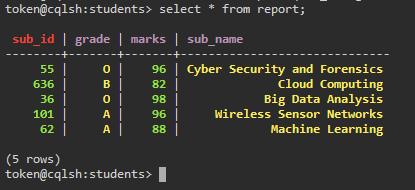
cqlsh:mallika> INSERT INTO report (sub\_id, grade, marks, sub\_name) VALUES (36, 'O', 98, 'Big Data Analysis');

cqlsh:mallika> INSERT INTO report (sub\_id, grade, marks, sub\_name) VALUES (55, 'O', 96, 'Cyber Security and Forensics');

cqlsh:mallika> INSERT INTO report (sub\_id, grade, marks, sub\_name) VALUES (602, ‘A’, 88, 'Machine Learning');

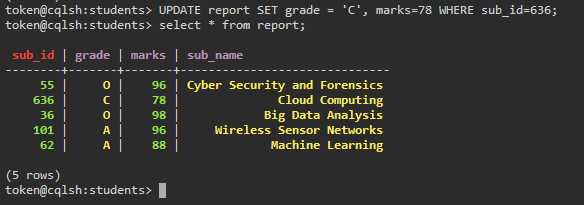
cqlsh:mallika> INSERT INTO report (sub\_id, grade, marks, sub\_name) VALUES (636, 'B', 82, 'Cloud Computing');

cqlsh:mallika> INSERT INTO report (sub\_id, grade, marks, sub\_name) VALUES (101, 'A', 96, 'Wireless Sensor Networks');



**Updating Data in a Table:**

UPDATE report SET grade = 'C', marks=78 WHERE sub\_id=636;



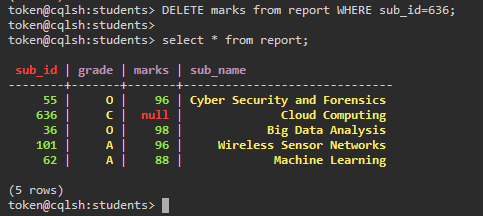
**Reading Required Columns:**

SELECT sub\_id, sub\_name from report;



**Deleting Datafrom a Table:**

DELETE marks from report WHERE sub\_id=636;



**Deleting an Entire Row:**

cqlsh:mallika> DELETE from report WHERE sub\_id=636;

