PART 3:

❖ Add - Update – Delete:

CRUD:

- C-create/insert
- R-remove
- U-update
- D-delete

This is applicable for a Collection (Table) or a Document (Row)

- **INSERT:**

```
db> const stuData={
... "name":"Chandana",
... "age":20,
... "courses":["Mathematics","Computer Science","English"],
... "gpa":3.8,
... "home_city":"New York",
... "blood_group":"B+",
... "is_hostel_resident":true
... };

db> db.stu.insertOne(stuData);
{
  acknowledged: true,
  insertedId: ObjectId('6661d7bd48f9e36487cdcdf6')
}
db> db.stu.find().count()
501
```

Here, we are inserting a new student data into a collection.

And after mentioning the details of student we are using the command called insertOne to insert one student details.

After it has successfully uploaded we can check the count for confirmation.

To insert the data into the collection we use the following command:

Const stuData= {

“name”: ”Chandana”,

“age”:”20”,

“course”: [“Mathematics” “Computer Science” “English”],

“gpa”:3.8,

‘home_city’: ”New York”,

“blood_group” : ”B+”,

“is_hotel_resident”: true

}

And after mentioning the details of the student, we use the

‘insertOne’ command to insert a single student's details.

After it has successfully uploaded, we can check the count for confirmation.

- **Update:**

Here, we can update any data that are present in the collections.

To update we use ‘$set’ command.

```
db> db.stu.updateOne({name:"Chandana"},{$set:{gpa:2.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- **Delete:**

  The delete operation is used to delete the data present in the given collection.

```
db> db.stu.deleteOne({name:"Chandana"})
{ acknowledged: true, deletedCount: 1 }
db>
```

- **Projection:**

  - This is used when we don't need all columns or attributes.
  - The projection document is used as the second argument to the find method.
  - Include field names with a value of 1 to specify fields to be returned.

Get Selected Attributes

```
db> db.stu.find({},{name:1,gpa:1})
[
  {
    _id: ObjectId('665759022abe60278b889fe7'),
    name: 'Student 948',
    gpa: 3.44
  },
  {
    _id: ObjectId('665759022abe60278b889fe8'),
    name: 'Student 157',
    gpa: 2.27
  },
  {
    _id: ObjectId('665759022abe60278b889fe9'),
    name: 'Student 316',
    gpa: 2.32
  },
```

In the above example it shows only the name and gpa , because the command is given as 'name:1' and 'gpa:1'.

Ignore Attributes:

Here, we have ignored the Id and home city of a student.

```
db> db.stu.find({},{_id:0,home_city:0});
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    name: 'Student 157',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 2.27,
    blood_group: 'O-',
    is_hotel_resident: true
  },
```

- **Benefits of Projection:**

  - Reduced data transferred between the database and your application.
  - Simplifies your code by focusing on the specific information you need.
  - Improve query performance by retrieving only necessary data.

❖ Limit And Selectors:

➢ Limit:

The limit operator is used with the find method. It's chained after the filter criteria or any sorting operations.

- Syntax:

```
db.collection.find({filter},
{projection}).limit(number);
```

```
db> db.stu.find({},{_id:0,home_city:0}).limit(1)
[
  {
    name: 'Student 948',
    age: 19,
    courses: "['English', 'Computer Science', 'Physics', 'Mathematics']",
    gpa: 3.44,
    blood_group: 'O+',
    is_hotel_resident: true
  }
]
db>
```

Here, to get only first document we have used the limit(1);

➢ Selectors:

- **Comparison greater than(gt) and less than(lt):**

To find all the students with age greater then 20

```
db> db.stu.find({age:{$gt:20}});
[
  {
    _id: ObjectId('665759022abe60278b889fea'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665759022abe60278b889feb'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
```

```
db> db.stu.find({age:{$lt:20}}).count()
124
db> db.stu.find({age:{$gt:20}}).count()
310
db>
```

Here, we have 310 students who are older than 20 and 124 students who are elder than 20.