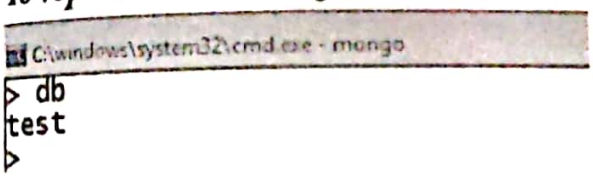
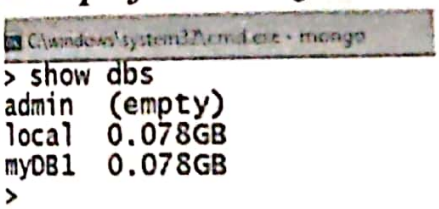
A few commands worth looking at are as follows (try them!!!).

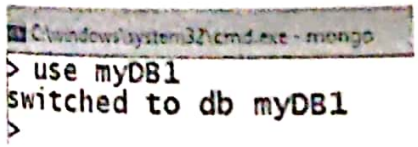
To report the name of the current database:



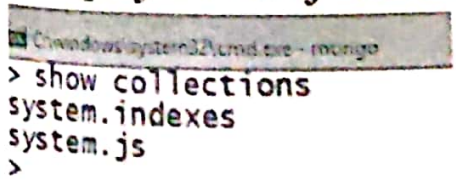
To display the list of databases:



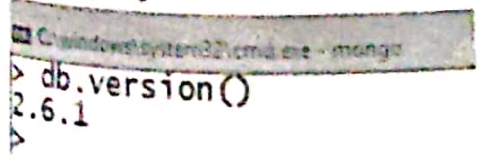
To switch to a new database, for example, myDB1:



To display the list of collections (tables) in the current database:



To display the current version of the MongoDB server:



RDBMS	MongoDB
Select StudRollNo, StudName, Hobbies From Students Where Grade ='VII' and Hobbies ='Ice Hockey'	db.Students.find({Grade: 'VII' , Hobbies: 'Ice Hockey'}, {StudRollNo : 1, StudName: 1, Hobbies : 1, _id:0})
Select StudRollNo, StudName, Hobbies From Students Where Grade ='VII' or Hobbies = 'Ice Hockey'	db.Students.find({ $or: [{Grade: 'VII' , Hobbies: 'Ice Hockey'}], StudRollNo : 1, StudName: 1, Hobbies : 1, _id:0})
Select * From Students Where StudName like 'S%'	db.Students.find({ StudName: /^S/}).pretty()

6.5 MONGODB QUERY LANGUAGE

CRUD (Create, Read Update, and Delete) operations in MongoDB

Create → Creation of data is done using insert() or update() or save() method.
Read → Reading the data is performed using the find() method.
Update → Update to data is accomplished using the update() method with UPSERT set to false.
Delete → a document is Deleted using the remove() method.

We will present the various methods available in MongoDB shell to deal with data in the next few sections. The sections have been designed as follows:

Objective: What is it that we are trying to achieve here?
Input: What is the input that has been given to us to act upon?
Act: The actual statement/command to accomplish the task at hand.
Outcome: The result/output as a consequence of executing the statement.

At few places we have also provided the equivalent in RDBMS such as Oracle.

Objective: To create a collection by the name "Person". Let us take a look at the collection list prior to the creation of the new collection "Person":

```
C:\windows\system32\cmd.exe - mongo
> show collections
Students
food
system.indexes
system.js
>
```

Act: The statement to create the collection is

db.createCollection("Person")

```
> db.createCollection("Person");
{ "ok" : 1 }
>
```

Outcome: Below is the collection list after the creation of the new collection "Person":

```
> show collections;
Person
Students
food
system.indexes
system.js
>
```

Objective: To drop a collection by the name "food".

Take a look at the current collection list:

```
> show collections;
Person
Students
food
system.indexes
system.js
>
```

Act: The statement to drop the collection is

```
db.food.drop();
```

```
> db.food.drop();
true
>
```

Outcome: The collection list after the execution of the statement is as follows:

```
> show collections
Person
Students
system.indexes
system.js
>
```

6.5.1 Insert Method

We now explain the syntax of insert method.

```
db.students.insert(          <-- Collection
{
RollNo: 101,                 <-- Field: value
Age: 19,                     <-- Field: value
ContactNo:0123456789,        <-- Field: value
EmailID:Sample@abc.com       <-- Field: value
}
)
```

Objective: To create a collection by the name "Students" and insert documents.

Input: Check the list of existing collections.

```
> show collections
system.indexes
system.js
>
```

Act: Create a collection by the name "Students" and store the following data in it.

db.Students.insert({_id:1, StudName:"Michelle Jacintha", Grade: "VII", Hobbies: "Internet Surfing"});

```
> db.Students.insert({_id:1, StudName:"Michelle Jacintha", Grade: "VII", Hobbies: "Internet Surfing"});
WriteResult({ "nInserted" : 1 })
>
```

Outcome: Check if the collection has been successfully created.

```
> show collections
Students
system.indexes
system.js
>
```

Check if the document for Student "Michelle Jacintha" has been successfully inserted into the "Students" collection.

```
> db.Students.find();
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : "Internet Surfing" }
>
```

To format the result, one can add the pretty() method to the operation.

```
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
>
```

Objective: Insert another document into the collection.

Input: Check the documents in the "Students" collection before proceeding.

```
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
>
```

Act:

db.Students.insert({_id:2, StudName:"Mabel Mathews", Grade: "VII", Hobbies: "Baseball"});

```
> db.Students.insert({_id:2, StudName:"Mabel Mathews", Grade: "VII", Hobbies: "Baseball"});
WriteResult({ "nInserted" : 1 })
>
```

Outcome:

```
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
```

Objective: Insert the document for "Aryan David" into the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from "Skating" to "Chess".) Use "**Update else insert**" (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

Input: Check the documents in the "Students" collection before proceeding.

```
C:\windows\system32\cmd.exe - mongo
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
```

Act:

db.Students.update({_id:3, StudName:"Aryan David", Grade: "VII"},{$set:{Hobbies: "Skating"}},{upsert:true});

```
> db.Students.update({_id:3, StudName:"Aryan David", Grade: "VII"},{$set:{Hobbies: "Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
>
```

Outcome: Confirm the presence of the document of "Aryan David" in the "Students" collection.

```
> db.Students.find({_id:3});
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Skating" }
>
```

Objective: Insert the document for "Aryan David" into the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from "Skating" to "Chess"). Use "**Update else insert**" (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it). Try the UPSERT operator by setting it first to "true" and then to "false" and observe the output.

Input: Check the documents in the "Students" collection before proceeding.

```
C:\windows\system32\cmd.exe - mongo
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Skating"
}
>
```

Act:

db.Students.update({_id:3, StudName:"Aryan David", Grade: "VII"},{$set:{Hobbies: "Chess"}},{upsert:true});

```
C:\windows\system32\cmd.exe - mongo
> db.Students.update({_id:3, StudName:"Aryan David", Grade: "VII"},{$set:{Hobbies: "Chess"}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Outcome: Confirm that the required changes have been made to the document of "Aryan David" in the "Students" collection.

```
C:\windows\system32\cmd.exe - mongo
> db.Students.find({_id:3});
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess" }
>
```

Objective: To demonstrate Save method to insert a document for student "Vamsi Bapat" in the "Students" collection. Omit providing value for the _id key.

Input: Check the documents in the "Students" collection before proceeding.

```
C:\windows\system32\cmd.exe - mongo
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
>
```

Act:
db.Students.save({StudName:"Vamsi Bapat",Grade:"VI"})

```
> db.Students.save({StudName:"Vamsi Bapat",Grade:"VI"})
WriteResult({ "nInserted" : 1 })
>
```

Outcome: Confirm the presence of the document of "Vamsi Bapat" in the "Students" collection.

```
> db.Students.find().pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : ObjectId("546dd0e0a7fba710799bb94d"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
>
```

6.5.2 Save() Method

We now explain the save() method. The save() method will insert a new document if the document with the specified _id does not exist. However, if a document with the specified id exists, it replaces the existing document with the new one.

Objective: Insert the document of "Hersch Gibbs" into the Students collection using the Update method. First try with upsert set to false and then with upsert set to true.

Input: Check the documents in the "Students" collection before proceeding.

```
> db.Students.find();
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : "Internet Surfing" }
{ "_id" : 2, "StudName" : "Mabel Mathews", "Grade" : "VII", "Hobbies" : "Baseball" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess" }
{ "_id" : ObjectId("546dd0e0a7fba710799bb94d"), "StudName" : "Vamsi Bapat", "Grade" : "VI" }
>
```

Act: *Update method with upsert set to false.*
db.Students.update({_id:4, StudName:"Hersch Gibbs", Grade: "VII"},{$set:{Hobbies: "Graffiti"}}, {upsert:false});

```
> db.Students.update({_id:4, StudName:"Hersch Gibbs", Grade: "VII"},{$set:{Hobbies: "Graffiti"}}, {upsert:false});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

As evident from the above display (nUpserted : 0), no document has been inserted because upsert is set to false.

Update method with upsert set to true.

db.Students.update({_id:4, StudName:"Hersch Gibbs", Grade: "VII"},{$set:{Hobbies: "Graffiti"}}, {upsert:true});

```
> db.Students.update({_id:4, StudName:"Hersch Gibbs", Grade: "VII"},{$set:{Hobbies: "Graffiti"}}, {upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 4 })
>
```

nUpserted: 1 implies that a document with _id:4 has been inserted.

Outcome: Confirm the presence of the document of "Hersch Gibbs" in the "Students" collection.

```
> db.Students.find();
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : "Internet Surfing" }
{ "_id" : 2, "StudName" : "Mabel Mathews", "Grade" : "VII", "Hobbies" : "Baseball" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess" }
{ "_id" : ObjectId("546dd0e0a7fba710799bb94d"), "StudName" : "Vamsi Bapat", "Grade" : "VI" }
{ "_id" : 4, "Grade" : "VII", "StudName" : "Hersch Gibbs", "Hobbies" : "Graffiti" }
>
```

6.5.3 Adding a New Field to an Existing Document – Update Method

We now discuss the syntax of update method.

```
db.students.update(          <-------  Collection
{Age: {$gt 18}},             <-------  Update Criteria
{$set: {Status: "A" }}       <-------  Update Action
{multi:true}                 <-------  Update Option
)
```

Objective: To add a new field "Location" with value "Newark" to the document (_id:4) of "Students" collection.

Input: Check the document (_id:4) in the "Students" collection before proceeding.

```
> db.Students.find({_id:4}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
>
```

Act:

db.Students.update({_id:4},{$set:{Location:"Newark"}});

```
> db.Students.update({_id:4},{$set:{Location:"Newark"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Outcome: Confirm that the new field "Location" with value "Newark" has been added to document (_id:4) in the "Students" collection.

```
> db.Students.find({_id:4}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti",
        "Location" : "Newark"
}
>
```

6.5.4 Removing an Existing Field from an Existing Document – Remove Method

In this section we will explain the syntax of remove method.

```
db.students.remove(          <-------- Collection
{Age: {$gt 18}},             <-------- Remove Criteria
)
```

Objective: To remove the field "Location" with value "Newark" in the document (_id:4) of "Students" collection.

Input: Check the document (_id:4) in the "Students" collection before proceeding.

```
> db.students.find({_id:4}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti",
        "Location" : "Newark"
}
>
```

Act:

```
db.Students.update({_id:4},{$unset:{Location:"Newark"}});
```

```
> db.students.update({_id:4},{$unset:{Location:"Newark"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Outcome: Confirm if the stated field ("Location") has been dropped from the document (_id:4) of the "Students" collection.

```
> db.students.find({_id:4}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
>
```

6.5.5 Finding Documents based on Search Criteria – Find Method

The syntax of find method is as follows:

```
db.students.find(            <-------- Collection
{Age: {$gt 18}},             <-------- Selection Criteria
{RollNo:1,Age:1,_id:1}       <-------- Projection
).limit(10)                  <-------- Cursor Modifier
```

Objective: To search for documents from the "Students" collection based on certain search criteria.

Input: Check the documents in the "Students" collection before proceeding.

```
> db.Students.find({}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

Act: Find the document wherein the "StudName" has value "Aryan David".

db.Students.find({StudName:"Aryan David"});

Outcome:

```
> db.Students.find({StudName:"Aryan David"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess" }
>
```

To format the above output, use the pretty() method:

db.Students.find({StudName:"Aryan David"}).pretty();

```
> db.Students.find({StudName:"Aryan David"}).pretty();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
>
```

RDBMS equivalent:

```
Select *
        From Students
                Where StudName like 'Aryan David';
```

```
SQL> select * from Students where StudName like 'Aryan David';
STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
3     Aryan David                 VII   Chess
SQL>
```

Objective: To display only the StudName from all the documents of the Student's collection. The identifier "__id" should be suppressed and NOT displayed.

Act:

db.Students.find({},{StudName:1,_id:0});

Outcome:

```
> db.Students.find({},{studName:1,_id:0});
{ "studName" : "Michelle Jacintha" }
{ "studName" : "Aryan David" }
{ "studName" : "Hersch Gibbs" }
{ "studName" : "Vamsi Bapat" }
{ "studName" : "Mabel Mathews" }
{ "studName" : "Mabel Mathews" }
>
```

RDBMS equivalent:

Select StudName
From Students;

```
SQL> select StudName from Students;

STUDNAME
--------------------
Michelle Jacintha
Aryan David
Mabel Mathews
Hersch Gibbs
Vamsi Bapat

SQL>
```

Objective: To display only the StudName and Grade from all the documents of the Students collection. The identifier _id should be suppressed and NOT displayed.

Act:

db.Students.find({},{StudName:1,Grade:1,_id:0});

Outcome:

```
> db.Students.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Michelle Jacintha", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "Aryan David" }
{ "Grade" : "VII", "StudName" : "Hersch Gibbs" }
{ "StudName" : "Vamsi Bapat", "Grade" : "VI" }
{ "StudName" : "Mabel Mathews", "Grade" : "VII" }
>
```

RDBMS equivalent:

Select StudName, Grade
From Students;

```
SQL> select StudName, Grade from Students;

STUDNAME             GRADE
-------------------- -----
Michelle Jacintha    VII
Aryan David          VII
Mabel Mathews        VII
Hersch Gibbs         VII
Vamsi Bapat          VI

SQL>
```

Objective: To display the StudName, Grade as well the identifier, _id from the document of the Students collection where the _id column is 1.

Act:

db.Students.find({_id:1},{StudName:1,Grade:1});

Outcome:

```
> db.students.find({_id:1},{StudName:1,Grade:1});
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII" }
>
```

RDBMS equivalent:

Select StudRollNo, StudName, Grade
From Students
Where StudRollNo = '1';

```
SQL> select StudRollNo, StudName, Grade from Students where StudRollNo = '1';
STUDR STUDNAME                 GRADE
----- ------------------------ -----
1     Michelle Jacintha        VII
SQL>
```

Objective: To display the StudName and Grade from the document of the Students collection where the _id column is 1. The _id field should NOT be displayed.

Act:

db.Students.find({_id:1},{StudName:1,Grade:1,_id:0});

Outcome:

```
> db.Students.find({_id:1},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Michelle Jacintha", "Grade" : "VII" }
>
```

RDBMS equivalent:

Select StudName, Grade
From Students
Where StudRollNo like '1';

```
SQL> select StudName, Grade from Students where StudRollNo like '1';
STUDNAME                 GRADE
------------------------ -----
Michelle Jacintha        VII
SQL>
```

Relational operators available to use in the search criteria:

$eq	→ equal to
$ne	→ not equal to
$gte	→ greater than or equal to
$lte	→ less than or equal to
$gt	→ greater than
$lt	→ less than

Objective: To find those documents where the Grade is set to 'VII'.

Act:

db.Students.find({Grade:{$eq:'VII'}}).pretty();

Outcome:

```
> db.students.find({Grade:{$eq:'VII'}}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where Grade like 'VII';
```

```
SQL> select * from Students where Grade like 'VII';

STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
    1 Michelle Jacintha           VII   Internet surfing
    3 Aryan David                 VII   Chess
    2 Mabel Mathews               VII   Baseball
    4 Hersch Gibbs                VII   Graffiti

SQL>
```

Objective: To find those documents where the Grade is NOT set to 'VII'.

Act:

db.Students.find({Grade:{$ne:'VII'}}).pretty();

Outcome:

```
> db.Students.find({Grade:{$ne:'VI'}}).pretty();
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
>
```

There is just one document that meets the above criteria of Grade NOT EQUAL to 'VII'.

RDBMS Equivalent:

```
Select *
        From Students
                Where Grade <> 'VII';
```

```
SQL> select * from Students where Grade <> 'VII';

STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
    5 Vamsi Bapat                 VI

SQL>
```

```
SQL> select * from Students where Grade != 'VII';
STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
      Vamsi Bapat                 VI

SQL>
```

Objective: To find those documents from the Students collection where the Hobbies is set to either 'Chess' or is set to 'Skating'.

Act:

```
db.Students.find ({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();
```

Outcome:

```
> db.Students.find ({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where Hobbies in ('Chess', 'Skating');
```

```
SQL> select * from Students where Hobbies in ('Chess', 'Skating');
STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
3     Aryan David                 VII   Chess
SQL>
```

Objective: To find those documents from the Students collection where the Hobbies is set neither to 'Chess' nor is set to 'Skating'.

Act:

```
db.Students.find ({Hobbies :{ $nin: ['Chess','Skating']}}).pretty ();
```

Outcome:

```
> db.Students.find ({Hobbies :{ $nin: ['Chess','Skating']}}).pretty ();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where Hobbies not in ('Chess', 'Skating');
```

```
SQL> select * from Students where Hobbies not in ('Chess','Skating');
STUDR STUDNAME                 GRADE HOBBIES
----- ------------------------ ----- ------------------------------
1     Michelle Jacintha        VII   Internet surfing
2     Mabel Mathews            VII   Baseball
4     Hersch Gibbs             VII   Graffiti

SQL>
```

Objective: To find those documents from the Students collection where the Hobbies is set to 'Graffiti' and the StudName is set to 'Hersch Gibbs' (AND condition).

Act:

```
db.Students.find({Hobbies:'Graffiti', StudName: 'Hersch Gibbs'}).pretty();
```

Outcome:

```
> db.Students.find({Hobbies:'Graffiti', StudName: 'Hersch Gibbs'}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where Hobbies like 'Graffiti' and StudName like 'Hersch Gibbs';
```

```
SQL> select * from Students where Hobbies like 'Graffiti' and StudName like 'Her
sch Gibbs';
STUDR STUDNAME                 GRADE HOBBIES
----- ------------------------ ----- ------------------------------
4     Hersch Gibbs             VII   Graffiti

SQL>
```

Objective: To find documents from the Students collection where the StudName begins with "M".

Act:

```
db.Students.find({StudName:/^M/}).pretty();
```

Outcome:

```
> db.Students.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
```

RDBMS Equivalent:

```
Select *
        From Students
                Where StudName like 'M%';
```

```
SQL> select * from Students where StudName like 'M%';
STUDR STUDNAME                    GRADE HOBBIES
----- ------------------------- ----- ------------------------------
1     Michelle Jacintha           VII   Internet surfing
2     Mabel Mathews               VII   Baseball
SQL>
```

Objective: To find documents from the Students collection where the StudName ends in "s".

Act:

```
db.Students.find({StudName:/s$/}).pretty();
```

Outcome:

```
> db.Students.find({StudName:/s$/}).pretty();
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where StudName like '%s';
```

```
SQL> select * from Students where StudName like '%s';
STUDR STUDNAME                    GRADE HOBBIES
----- ------------------------- ----- ------------------------------
2     Mabel Mathews               VII   Baseball
4     Hersch Gibbs                VII   Graffiti
SQL>
```

Objective: To find documents from the Students collection where the StudName has an "e" in any position.

Act:

```
db.Students.find({StudName:/e/}).pretty();
```

OR

```
db.Students.find({StudName:/.*e.*/}).pretty();
```

OR

```
db.Students.find({StudName:{$regex:"e"}}).pretty();
```

Outcome:

```
> db.Students.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

RDBMS Equivalent:

Select *

From Students

Where StudName like '%e%';

```
SQL> select * from Students where StudName like '%e%';

STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
1     Michelle Jacintha           VII   Internet surfing
2     Mabel Mathews               VII   Baseball
4     Hersch Gibbs                VII   Graffiti

SQL>
```

Objective: To find documents from the Students collection where the StudName ends in "a".

Act:

db.Students.find({StudName:{$regex:"a$"}}).pretty();

Outcome:

```
> db.Students.find({StudName:{$regex:"a$"}}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
>
```

RDBMS Equivalent:

Select *

From Students

Where StudName like '%a';

```
SQL> select * from Students where StudName like '%a';

STUDR STUDNAME                    GRADE HOBBIES
----- --------------------------- ----- ------------------------------
1     Michelle Jacintha           VII   Internet surfing

SQL>
```

Objective: To find documents from the Students collection where the StudName begins with "M".

Act:

```
db.Students.find({StudName:{$regex:"^M"}}).pretty();
```

Outcome:

```
> db.Students.find({StudName:{$regex:"^M"}}).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Where StudName like 'M%';
```

```
SQL> select * from Students where StudName like 'M%';

STUDR STUDNAME                 GRADE HOBBIES
----- ------------------------ ----- ------------------------------
1     Michelle Jacintha        VII   Internet surfing
2     Mabel Mathews            VII   Baseball

SQL>
```

6.5.6 Dealing with NULL Values

Objective: To add a new field with null value in existing documents (_id:3 and _id:4) of Students collection. A NULL is a missing or unknown value. When we place NULL as a value for a field, it implies that currently we do not know the value or the value is missing. We can always update the value of the field once we know it.

Input: Before we execute the commands to update documents with a null value in a column, let us first view the two documents.

```
db.Students.find({$or:[{_id:3},{_id:4}]})
```

```
> db.Students.find({$or:[{_id:3},{_id:4}]});
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess" }
{ "_id" : 4, "Grade" : "VII", "StudName" : "Hersch Gibbs", "Hobbies" : "Graffiti" }
>
```

Act: Update the documents with NULL values in the "Location" column.

```
db.Students.update({_id:3},{$set:{Location:null}});
db.Students.update({_id:4},{$set:{Location:null}});
```

```
> db.Students.update({_id:3},{$set:{Location:null}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Students.update({_id:4},{$set:{Location:null}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

RDBMS Equivalent:

```
Update Students
        Set Location = null
                Where StudRollNo in ('3','4');
```

```
SQL> update Students set Location = null where StudRollNo in ('3','4');
2 rows updated.
SQL>
```

Outcome: To search for NULL values in Location column.

db.Students.find({Location:{$eq:null}});

```
> db.Students.find({Location:{$eq:null}});
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : "Internet Surfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess", "Location" : null }
{ "_id" : 4, "Grade" : "VII", "StudName" : "Hersch Gibbs", "Hobbies" : "Graffiti", "Location" : null }
{ "_id" : ObjectId("5464849889ad1ab07d489b7f"), "StudName" : "Vamsi Bapat", "Grade" : "VI" }
{ "_id" : 2, "StudName" : "Mabel Mathews", "Grade" : "VII", "Hobbies" : "Baseball" }
>
```

The above statement displays documents which either have NULL values in the location column or do not have the location column at all.

RDBMS Equivalent:

```
Select *
        From Students
                Where Location is Null;
```

```
SQL> select * from Students where Location is NULL;

STUDR STUDNAME                    GRADE HOBBIES                        LOCATION
----- --------------------------- ----- ------------------------------ ----------
---------
    1 Michelle Jacintha           VII   Internet surfing
    3 Aryan David                 VII   Chess
    4 Mabel Mathews               VII   Baseball
    4 Hersch Gibbs                VII   Graffiti
      Vamsi Bapat                 VI

SQL>
```

Objective: To remove "Location" field having "NULL" values from the documents (_id:3 and _id:4) from the Students collection.

Input: Document from the "Students" collection having "NULL" values in the "Location" column.

```
> db.Students.find({Location:{$eq:null}});
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : "Internet Surfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess", "Location" : null }
{ "_id" : 4, "Grade" : "VII", "StudName" : "Hersch Gibbs", "Hobbies" : "Graffiti", "Location" : null }
{ "_id" : ObjectId("5464849889ad1ab07d489b7f"), "StudName" : "Vamsi Bapat", "Grade" : "VI" }
{ "_id" : 2, "StudName" : "Mabel Mathews", "Grade" : "VII", "Hobbies" : "Baseball" }
>
```

Act:

db.Students.update({_id:3},{$unset:{Location:null}});
db.Students.update({_id:4},{$unset:{Location:null}});

```
> db.Students.update({_id:3},{$unset:{Location:null}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.Students.update({_id:4},{$unset:{Location:null}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
>
```

Outcome: Let us confirm if the changes have been made by running find method on the Students collection.

```
> db.Students.find({})
{ "_id" : 1, "StudName" : "Michelle Jacintha", "Grade" : "VII", "Hobbies" : '
{ "_id" : 3, "Grade" : "VII", "StudName" : "Aryan David", "Hobbies" : "Chess'
{ "_id" : 4, "Grade" : "VII", "StudName" : "Hersch Gibbs", "Hobbies" : "Graf'
{ "_id" : ObjectId("5464849869ad1ab07d489b7f"), "StudName" : "Vamsi Bapat",
```

6.5.7 Count, Limit, Sort, and Skip

Objective: To find the number of documents in the Students collection.

Act:

```
db.Students.count()
```

Outcome:

```
> db.Students.count()
5
>
```

Objective: To find the number of documents in the Students collection wherein the Grade is VII.

Act:

```
db.Students.count({Grade:"VII"});
```

Outcome:

```
> db.Students.count({Grade:"VII"});
4
>
```

Objective: To retrieve the first 3 documents from the Students collection wherein the Grade is VII.

Act:

```
db.Students.find({Grade:"VII"}).limit(3).pretty();
```

Outcome:

```
> db.Students.find({Grade:"VII"}).limit(3).pretty();
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
>
```

RDBMS Equivalent:

Select *

From Students
Where Grade like 'VII' and rownum < 4;

```
SQL> select * from Students where Grade like 'VII' and rownum < 4;

STUDR STUDNAME                 GRADE HOBBIES                                  LOCATION
----- ------------------------ ----- ---------------------------------------- ----------
--------
1     Michelle Jacintha        VII   Internet surfing
3     Aryan David              VII   Chess
2     Mabel Mathews            VII   Baseball

SQL>
```

Objective: To sort the documents from the Students collection in the ascending order of StudName.

Act:

```
db.Students.find().sort({StudName:1}).pretty();
```

Outcome:

```
> db.Students.find().sort({StudName:1}).pretty();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
>
```

RDBMS Equivalent:

Select *
From Students
Order by StudName asc;

```
SQL> select * from Students order by StudName asc;

STUDR STUDNAME                 GRADE HOBBIES                                  LOCATION
----- ------------------------ ----- ---------------------------------------- ----------
--------
3     Aryan David              VII   Chess
4     Hersch Gibbs             VII   Graffiti
2     Mabel Mathews            VII   Baseball
1     Michelle Jacintha        VII   Internet surfing
      Vamsi Bapat              VI

SQL>
```

Objective: To sort the documents from the Students collection in the descending order of StudName.

Act:

```
db.Students.find().sort({StudName:-1}).pretty();
```

Outcome:

```
> db.Students.find().sort({StudName:-1}).pretty();
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
```

RDBMS Equivalent:

```
Select *
        From Students
                Order by StudName desc;
```

```
SQL> select * from Students order by StudName desc;
STUDR STUDNAME                 GRADE HOBBIES                        LOCATION
----- ------------------------ ----- ------------------------------ ----------
      Vamsi Bapat              VI
1     Michelle Jacintha        VII   Internet surfing
2     Mabel Mathews            VII   Baseball
4     Hersch Gibbs             VII   Graffiti
3     Aryan David              VII   Chess
SQL>
```

Objective: To sort the documents from the Students collection first on Grade in ascending order and then on Hobbies in descending order.

Act:

db.Students.find().sort({Grade:1, Hobbies:-1}).pretty();

Outcome:

```
> db.Students.find().sort({Grade:1, Hobbies:-1}).pretty();
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
```

RDBMS Equivalent:

```
Select *
        From Students
                Order by Grade asc, hobbies desc;
```

```
SQL> select * from Students order by Grade asc, Hobbies desc;

STUDR STUDNAME                 GRADE HOBBIES                        LOCATION
----- ------------------------ ----- ------------------------------ ----------
--------
      Vamsi Bapat              VI
      Michelle Jacintha        VII   Internet surfing
1     Hersch Gibbs             VII   Graffiti
4     Aryan David              VII   Chess
3     Mabel Mathews            VII   Baseball
2

SQL>
```

Objective: To sort the documents from the Students collection first on Grade in ascending order and then on Hobbies in ascending order.

Act:

db.Students.find().sort({Grade:1, Hobbies:1}).pretty();

Outcome:

```
> db.Students.find().sort({Grade:1, Hobbies:1}).pretty();
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Order by Grade asc, Hobbies asc;
```

```
SQL> select * from Students order by Grade asc, Hobbies asc;

STUDR STUDNAME                 GRADE HOBBIES                        LOCATION
----- ------------------------ ----- ------------------------------ ----------
---------
      Vamsi Bapat              VI
2     Mabel Mathews            VII   Baseball
4     Aryan David              VII   Chess
3     Hersch Gibbs             VII   Graffiti
1     Michelle Jacintha        VII   Internet surfing

SQL>
```

RDBMS Equivalent:

```
Select *
        From Students
                Order by Grade asc, hobbies desc;
```

```
SQL> select * from Students order by Grade asc, Hobbies desc;

STUDR STUDNAME                    GRADE HOBBIES                        LOCATION
----- --------------------------- ----- ------------------------------ ----------
      Vamsi Bapat                 VI
      Michelle Jacintha           VII   Internet surfing
1     Hersch Gibbs                VII   Graffiti
4     Aryan David                 VII   Chess
3     Mabel Mathews               VII   Baseball

SQL>
```

Objective: To sort the documents from the Students collection first on Grade in ascending order and then on Hobbies in ascending order.

Act:

db.Students.find().sort({Grade:1, Hobbies:1}).pretty();

Outcome:

```
> db.Students.find().sort({Grade:1, Hobbies:1}).pretty();
{
        "_id" : ObjectId("5464849889ad1ab07d489b7f"),
        "StudName" : "Vamsi Bapat",
        "Grade" : "VI"
}
{
        "_id" : 2,
        "StudName" : "Mabel Mathews",
        "Grade" : "VII",
        "Hobbies" : "Baseball"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "Aryan David",
        "Hobbies" : "Chess"
}
{
        "_id" : 4,
        "Grade" : "VII",
        "StudName" : "Hersch Gibbs",
        "Hobbies" : "Graffiti"
}
{
        "_id" : 1,
        "StudName" : "Michelle Jacintha",
        "Grade" : "VII",
        "Hobbies" : "Internet Surfing"
}
>
```

RDBMS Equivalent:

```
Select *
        From Students
                Order by Grade asc, Hobbies asc;
```

```
SQL> select * from Students order by Grade asc, Hobbies asc;

STUDR STUDNAME                    GRADE HOBBIES                        LOCATION
----- --------------------------- ----- ------------------------------ ----------
      Vamsi Bapat                 VI
2     Mabel Mathews               VII   Baseball
4     Aryan David                 VII   Chess
1     Hersch Gibbs                VII   Graffiti
      Michelle Jacintha           VII   Internet surfing

SQL>
```