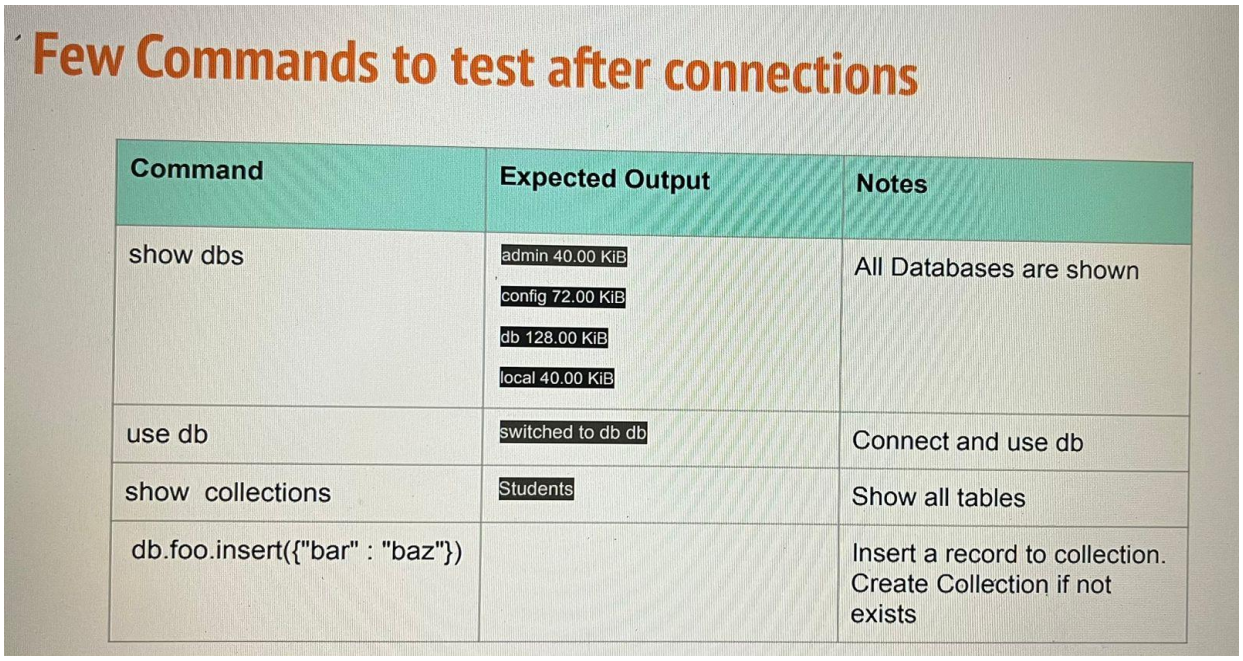
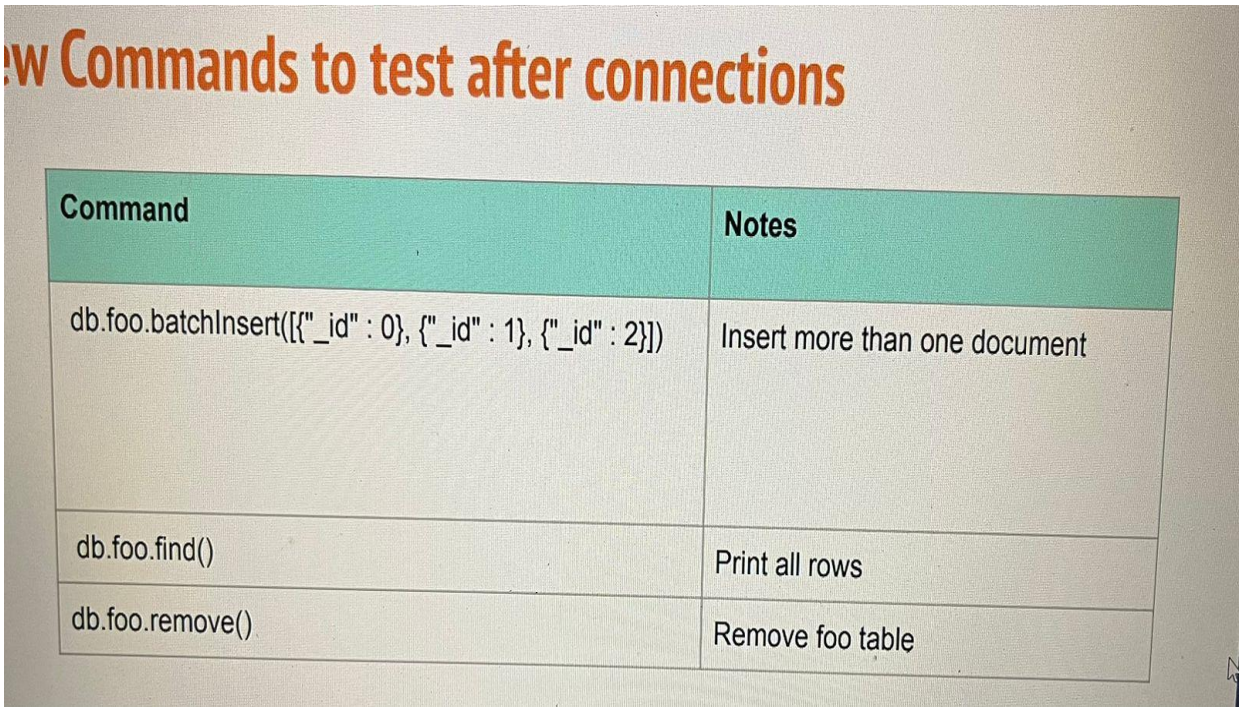
Class 2: ADD, UPDATE & DELETE

Few commands to test after connection



Few Commands to test after connections

Command	Expected Output	Notes
show dbs	admin 40.00 KiB config 72.00 KiB db 128.00 KiB local 40.00 KiB	All Databases are shown
use db	switched to db db	Connect and use db
show collections	Students	Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists



w Commands to test after connections

Command	Notes
db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])	Insert more than one document
db.foo.find()	Print all rows
db.foo.remove()	Remove foo table

1.Show dbs:

The show dbs command in MongoDB is used to list all the databases present on the MongoDB server. Here's a detailed explanation:

Command:

```
show dbs
```

What It Does:

- Lists Databases: It returns a list of all databases on the server along with their respective sizes on disk.

- Size Information: Each database in the list is accompanied by its size, showing how much space it occupies.

Output:

The output is a simple list, typically formatted like this:

```
admin       0.000GB
config      0.001GB
local       0.000GB
mydatabase  0.002GB
```

Usage Context:

- Initial Setup: After setting up MongoDB, you might use show dbs to confirm the creation of your databases.

- Monitoring: To monitor and ensure that all expected databases are present and check their sizes.

- Administrative Tasks: Helpful in administration tasks where you need to know which databases exist before performing operations like backups or migrations.

2.Use bds:

The use command in MongoDB is used to switch the current database context to the specified database. Here's a detailed explanation:

Command:

use <databaseName>

What It Does:

- Switches Context: Changes the context to the specified database. Any subsequent commands will be executed against this database.

- Creates Database on First Insert: If the specified database does not exist, MongoDB will not create it immediately when you run use. Instead, the database is created when you first insert data into it.

Example Usage:

1. Switch to an Existing Databas:

```
use mydatabase
```

This switches the context to mydatabase. If mydatabase exists, any subsequent operations (e.g., creating collections, inserting documents) will be executed within this database.

2. Create and Switch to a New Database:

```
use newdatabase
```

At this point, newdatabase does not exist. It will be created when you first insert a document into a collection within it:

```
db.mycollection.insertOne({ name: "test" })
```

After this insert, newdatabase is created and will appear in the output of show dbs.

Output:

The use command provides a simple acknowledgment indicating the switch:

switched to db mydatabase

Usage Context:

- Database Management: Switching to the correct database context is essential for performing operations on the intended database.

- Development and Testing: Developers often switch between databases to test different datasets or configurations.

3.Show collection:

The show collections command in MongoDB is used to list all the collections within the currently selected database. Here's a detailed explanation:

Command:

show collections

What It Does:

- Lists Collections: This command outputs the names of all collections present in the current database context.

Example Usage:

1. Switch to a Database:

Before using show collections, you need to switch to the desired database using the use command:

use mydatabase

2. List Collections:

After switching to the database, run:

show collections

This command will list all the collections in mydatabase.

Output:

The output will be a list of collection names. For example:

users

orders

products

inventory

Usage Context:

- Database Exploration: To understand the structure of the database by viewing all the collections it contains.

- Development and Debugging: Allows developers to quickly check which collections exist within the current database.

- Administrative Tasks: Helps database administrators manage and verify the collections within a database.

4.Insert Document:

In MongoDB, inserting a document into a collection is done using the insertOne, insertMany, or other similar methods. Here's a detailed explanation of how to insert a document using insertOne:

Command:

db.collection.insertOne(document)

What It Does:

- Inserts a Single Document: Adds one document to the specified collection. If the collection does not exist, MongoDB will create it upon the first insertion.

Syntax:

- db: Refers to the current database context.

- collection: The name of the collection where you want to insert the document.

- insertOne: The method used to insert a single document.

- document: The JSON-like object you want to insert.

Output:

The output will confirm the insertion and provide details about the inserted document, including the generated _id field:

```
json
{
  "acknowledged" : true,
  "insertedId" : ObjectId("60c72b2f9f1b8b6d5f3e7e72")
}
```

5.Find Document:

Finding documents in MongoDB is a common operation used to retrieve data from a collection based on certain criteria. The primary method for this is find(). Here's a detailed explanation:

Command:

```
db.collection.find(query, projection)
```

What It Does:

- Retrieves Documents: Returns documents from the specified collection that match the given query criteria.

- Projection: Optionally specifies which fields to include or exclude in the returned documents.

Syntax:

- db: Refers to the current database context.

- collection: The name of the collection from which you want to retrieve documents.

- find: The method used to perform the query.

- query: A document specifying the criteria for selecting documents. If empty, it returns all documents in the collection.

- projection: An optional document that specifies the fields to include or exclude in the returned documents.

Output:

The output will be a cursor to the documents that match the query criteria. For example:

json

{ "_id" : ObjectId("60c72b2f9f1b8b6d5f3e7e72"), "name" : "Alice", "age" : 30, "city" : "New York" }

{ "_id" : ObjectId("60c72b2f9f1b8b6d5f3e7e73"), "name" : "Bob", "age" : 25, "city" : "New York" }

6.Remove:

Removing documents in MongoDB can be done using methods like deleteOne, deleteMany, and the deprecated remove. Here's a detailed explanation focusing on deleteOne and deleteMany:

Command: deleteOne

```
db.collection.deleteOne(query)
```

Command: deleteMany

```
db.collection.deleteMany(query)
```

What They Do:

- deleteOne: Removes a single document that matches the specified query criteria. If multiple documents match, only the first one encountered will be deleted.

- deleteMan: Removes all documents that match the specified query criteria.

Syntax:

- db: Refers to the current database context.

- collection: The name of the collection from which you want to remove documents.

- deleteOne: The method used to remove a single document.

- deleteMany: The method used to remove multiple documents.

- query: A document specifying the criteria for selecting documents to be removed. If empty, no documents will be removed.

Output:

Both deleteOne and deleteMany return a result object that provides information about the operation:

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

- acknowledged: Indicates if the operation was acknowledged by the server.

- deletedCount: Shows the number of documents that were deleted.

DOCUMENTS COLLECTIONS DATABASE

DOCUMENT:

At the heart of MongoDB is the document:

an ordered set of keys with associated values.

The representation of a document varies by programming language, but most languages have a data structure that is a natural fit, such as a map, hash, or dictionary.

1. Insert:

- Adding a new document to a collection.
- Example:

```
db.users.insertOne({
  "name": "Eve",
  "age": 28,
  "city": "Los Angeles"
})
```

2. Find:

- Querying documents in a collection.
- Example:

```
db.users.find({ "city": "New York" })
```

3. Update:

- Modifying existing documents.
- Example:

```
db.users.updateOne(
  { "name": "Alice" },
  { $set: { "age": 31 } }
)
```

4. Delete:

- Removing documents from a collection.

- Example:

```
db.users.deleteOne({ "name": "Bob" })
```

COLLECTION:

Collections A collection is a group of documents. If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.

DATABASE:

MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

A database has its own permissions, and each database is stored in separate files on disk.

A good rule of thumb is to store all data for a single application in the same database.

DATATYPE:

Basically each document will be in JSON format which will be as follows. Where each attributes inside can be of multiple data types.