Group: - 1 from T4 Batch
0077 – Aakash Joshi
2039 – Akanksha Lokhande

- **Title:**

  MongoDB Queries

- **Date of completion:**

- **Problem statement:**
  Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators etc.)
- **Objectives:**
  1. To develop Database programming skills.
  2. To develop basic Database administration skills.

- **Software and Hardware Requirements:**
  Software – 64-bit Open-Source Linux/Windows, Oracle.
  Hardware – Computer ,Mouse ,Keyboard , CPU

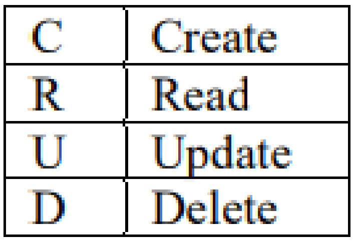
- **Theory :**

➢ MongoDB Queries:

MongoDB Query is a way to get the data from the MongoDB database. MongoDB queries provide the simplicity in process of fetching data from the database, it is like SQL queries in SQL Database language. While performing a query operation, one can also use criteria or conditions which can be used to retrieve specific data from the database.

➢ CRUD Operations:

As we know that we can use MongoDB for various things like building an application (including web and mobile), or analysis of data, or an administrator of a MongoDB database, in all these cases we need to interact with the MongoDB server to perform certain operations like entering new data into the application, updating data into the application, deleting data from the application, and reading the data of the application. MongoDB provides a set of some basic but most essential operations that will help you to easily interact with the MongoDB server and these operations are known as CRUD operations.



C	Create
R	Read
U	Update
D	Delete

<u>Create Operations :</u> Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.
MongoDB provides the following methods to insert documents into a collection:
• db.collection.insertOne()
• db.collection.insertMany()

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

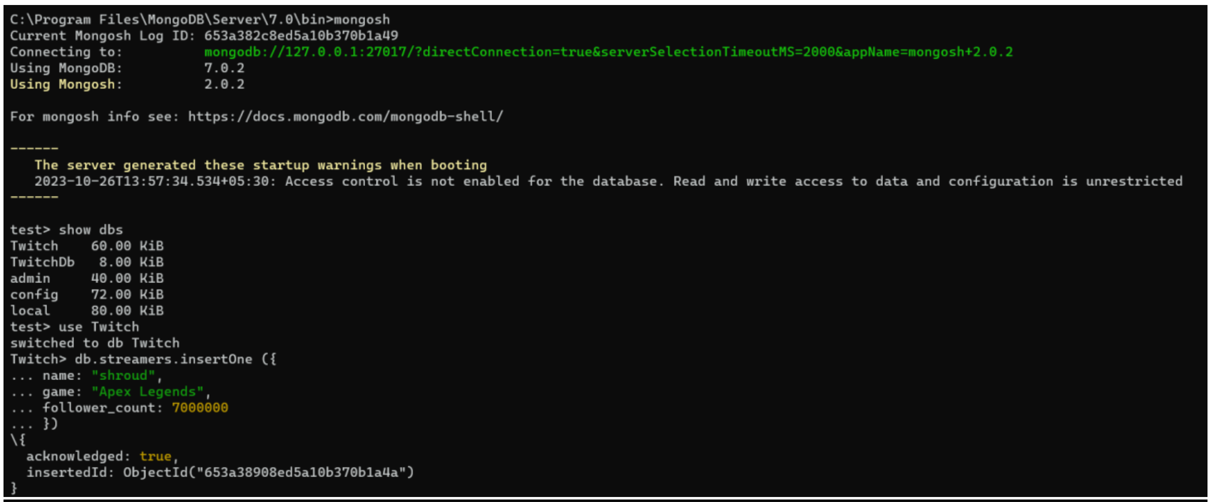
<u>Read Operations:</u>
Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:
• db.collection.find()
You can specify query filters or criteria that identify the documents to return.

<u>Update Operations :</u>
Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:
• db.collection.updateOne()
• db.collection.updateMany()

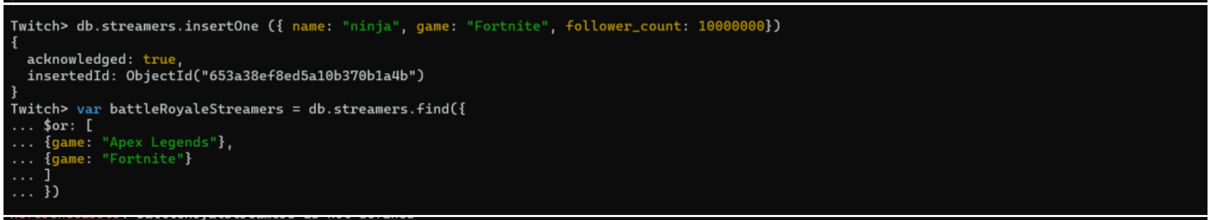
• db.collection.replaceOne()
In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document. You can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

Delete Operations :
Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:
• db.collection.deleteOne()
• db.collection.deleteMany()
In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document. You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

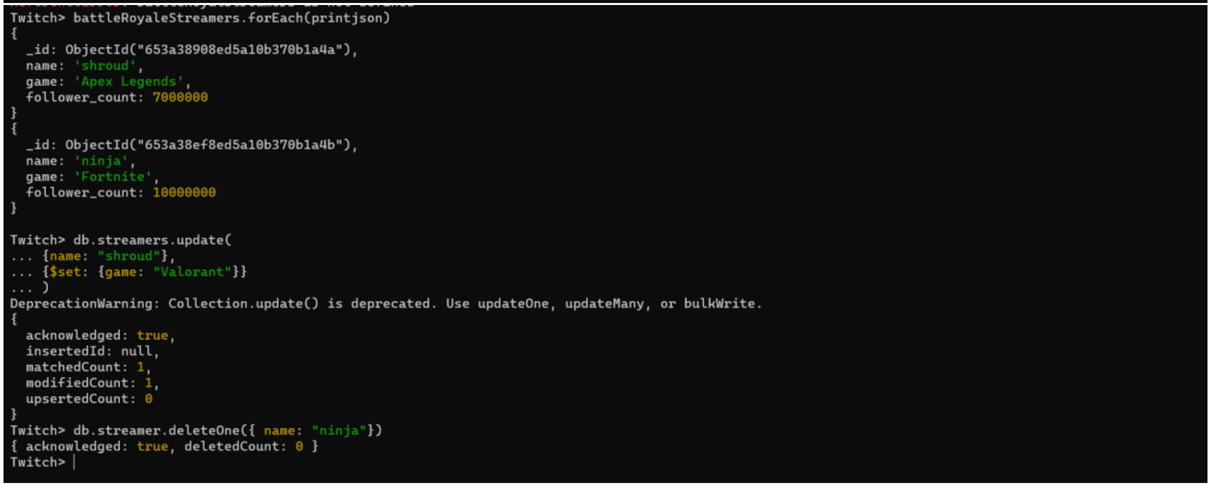
Code + Output:



```
C:\Program Files\MongoDB\Server\7.0\bin>mongosh
Current Mongosh Log ID: 653a382c8ed5a10b370b1a49
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.2
Using MongoDB:          7.0.2
Using Mongosh:          2.0.2

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

------
   The server generated these startup warnings when booting
   2023-10-26T13:57:34.534+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> show dbs
Twitch    60.00 KiB
TwitchDb   8.00 KiB
admin     40.00 KiB
config    72.00 KiB
local     80.00 KiB
test> use Twitch
switched to db Twitch
Twitch> db.streamers.insertOne ({
... name: "shroud",
... game: "Apex Legends",
... follower_count: 7000000
... })
\{
  acknowledged: true,
  insertedId: ObjectId("653a38908ed5a10b370b1a4a")
}
```



```
Twitch> db.streamers.insertOne ({ name: "ninja", game: "Fortnite", follower_count: 10000000})
{
  acknowledged: true,
  insertedId: ObjectId("653a38ef8ed5a10b370b1a4b")
}
Twitch> var battleRoyaleStreamers = db.streamers.find({
... $or: [
... {game: "Apex Legends"},
... {game: "Fortnite"}
... ]
... })
```



```
Twitch> battleRoyaleStreamers.forEach(printjson)
{
  _id: ObjectId("653a38908ed5a10b370b1a4a"),
  name: 'shroud',
  game: 'Apex Legends',
  follower_count: 7000000
}
{
  _id: ObjectId("653a38ef8ed5a10b370b1a4b"),
  name: 'ninja',
  game: 'Fortnite',
  follower_count: 10000000
}

Twitch> db.streamers.update(
... {name: "shroud"},
... {$set: {game: "Valorant"}}
... )
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Twitch> db.streamer.deleteOne({ name: "ninja"})
{ acknowledged: true, deletedCount: 0 }
Twitch> |
```