Part 2: Reading Assignment

a. Application of NoSQL Database in Web Crawling

1. The document talks about Web crawling, which is a way to collect and filter information in a database and then use it to create search engines. It also discusses how databases play a vital role as well as how relational databases have trouble achieving high availability, scalability, and performance.
2. There are also references to web crawling principles in this paper, such as using multiple spiders simultaneously to crawl pages, extract URLs, and save them in original page libraries.
3. The paper discusses the use case of the Meteorological BBS information collection system, which combines and filters messages from the representative meteorological BBS on the Internet to provide a professional search engine database for meteorological information.
4. Relational databases cannot be used for the Meteorological BBS information collection system due to the limitations of storing nested data structures, affecting the queries' performance.
5. A NoSQL database is an option because it reduces data consistency and integrity constraints in exchange for high availability and partition tolerance, which are needed to store large volumes of data online.
6. The discussion focused on MongoDB, a document-oriented database with a high performance (no joins needed), a low hardware cost, schema-free storage, and easy scalability.
7. Compared to MySQL, MongoDB works great when the amount of data exceeds 50GB. The MongoDB access speed is ten times faster than MySQL, so it's more suitable for data storage in web crawlers.

b. Comparing NoSQL MongoDB to an SQL DB

1. This paper compares NoSQL MongoDB to an SQL DB in terms of performance.
2. Data in the relational model is represented by a database schema, No SQL databases organize data into key-value pairs.
3. NoSQL database is used for unstructured and extremely large data, however, there is still ambiguity regarding the use of NoSQL or SQL for a modest-sized database.
4. Larger the schema in SQL DB, the longer it takes to fetch the data, but in NoSQL processing is simpler, more affordable, and more flexible.
5. Then the paper explains the related published work comparing NoSQL databases with SQL databases.
6. Although MongoDB is highly scalable with no schema definition yet writing queries can be complex for the user.
7. MongoDB can be similar in some respects to SQL join as it requires the user to define their own method for retrieving data on a reference.

8. There are no in-built aggregate functions in MongoDB, and one has to use MapReduce for those operations.
9. It further undertakes 4 experiments comparing the performance of MongoDB and SQL. It was found that MongoDB performs better than SQL in general. It also highlighted how MongoDB's performance degraded for aggregate functions and querying based on non-key values.
10. MongoDB would give better performance when used as a distributed database.

c. Data Aggregation System

1. This article explains how frequent more data lookups can become so expensive with data being scattered in multiple data sources. Data Aggregation System is one such system that provides makes this process simpler by providing a single point of access.
2. Cache server acts as an intermediary between data and queries received from the web front-end. It consists of one or more MongoDB shards.
3. Data in cache need not be stored as backup and can be recreated from original data source at any time. DAS is an entirely read-only system.
4. Two separate collections are used to store data: Raw and Merged caches where Merged cache acts as the central repository for a query with a single primary key.
5. This system reduces latency experienced by users while performing complex, cross-service queries.

Part 3: Programming Assignment

a. Create a database for a Contact Management System in MongoDB.

Commands:

use ContactManagementSystem;

```
27017> show dbs;
admin       40.00 KiB
config      60.00 KiB
local       72.00 KiB
todolistdb  72.00 KiB
27017> use ContactManagementSystem;
switched to db ContactManagementSystem
ContactManagementSystem> show dbs;
admin       40.00 KiB
config      60.00 KiB
local       72.00 KiB
todolistdb  72.00 KiB
ContactManagementSystem>
```

b. Create 5 records with different attributes and values you choose.

Commands:

db.CMScollection.insertMany (

[

{firstname: "Mark", lastname: "Jacobs", phone: " +13323039876"},
{firstname: "Anna", lastname:"Hathway", email:"anna@gmail.com", phone: "+19876543210"},
{firstname: "Mike", lastname:"Ross", address:"30 S.Huntington Ave", city:"Boston", zipcode: 02120},
{firstname: "Neal", lastname:"Reardon", address:"200 Pavonia Ave", city:"New York", country: "USA"},
{firstname: "Hanna", lastname:"Garcia", email: "garcia@college.edu", city:"San Francisco", country: "USA"}

]

)

```
ContactManagementSystem> db.CMScollection.insertMany([{firstname: "Mark",lastname: "Jacobs", phone: " +13323039876"},
... {firstname: "Anna", lastname:"Hathway", email:"anna@gmail.com", phone: "+19876543210"},
... {firstname: "Mike", lastname:"Ross", address:"30 S.Huntington Ave", city:"Boston", zipcode: 02120},
... {firstname: "Neal", lastname:"Reardon", address:"200 Pavonia Ave", city:"New York", country: "USA"},
... {firstname: "Hanna", lastname:"Garcia", email: "garcia@college.edu", city:"San Francisco", country: "USA"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6331ca3e70e8dbc65620b45a"),
    '1': ObjectId("6331ca3e70e8dbc65620b45b"),
    '2': ObjectId("6331ca3e70e8dbc65620b45c"),
    '3': ObjectId("6331ca3e70e8dbc65620b45d"),
    '4': ObjectId("6331ca3e70e8dbc65620b45e")
  }
}
```

Show records:

Command: db.CMSCollection.find()

```
ContactManagementSystem> db.CMScollection.find({})
[
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45a"),
    firstname: 'Mark',
    lastname: 'Jacobs',
    phone: ' +13323039876'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45b"),
    firstname: 'Anna',
    lastname: 'Hathway',
    email: 'anna@gmail.com',
    phone: '+19876543210'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45c"),
    firstname: 'Mike',
    lastname: 'Ross',
    address: '30 S.Huntington Ave',
    city: 'Boston',
    zipcode: 1104
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45d"),
    firstname: 'Neal',
    lastname: 'Reardon',
    address: '200 Pavonia Ave',
    city: 'New York',
    country: 'USA'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45e"),
    firstname: 'Hanna',
    lastname: 'Garcia',
    email: 'garcia@college.edu',
    city: 'San Francisco',
    country: 'USA'
  }
]
```

c. Delete a record of your choice

Command: db.CMScollection.deleteOne({firstname:"Mike"})

```
ContactManagementSystem> db.CMScollection.deleteOne({firstname:"Mike"})
{ acknowledged: true, deletedCount: 1 }
ContactManagementSystem> db.CMScollection.find({})
[
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45a"),
    firstname: 'Mark',
    lastname: 'Jacobs',
    phone: ' +13323039876'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45b"),
    firstname: 'Anna',
    lastname: 'Hathway',
    email: 'anna@gmail.com',
    phone: '+19876543210'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45d"),
    firstname: 'Neal',
    lastname: 'Reardon',
    address: '200 Pavonia Ave',
    city: 'New York',
    country: 'USA'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45e"),
    firstname: 'Hanna',
    lastname: 'Garcia',
    email: 'garcia@college.edu',
    city: 'San Francisco',
    country: 'USA'
  }
]
```

d. Update a record in the document.

Command: db.CMScollection.updateOne({firstname:"Anna"},

{$set:{email: "hathway@college.com"},

$currentDate:{lastModified: true}

})

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

ContactManagementSystem> db.CMScollection.updateOne({firstname:"Anna"},
... {
..... $set:{email: "hathway@college.com"},
..... $currentDate:{lastModified: true}
..... }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
ContactManagementSystem> db.CMScollection.find({})
[
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45a"),
    firstname: 'Mark',
    lastname: 'Jacobs',
    phone: ' +13323039876'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45b"),
    firstname: 'Anna',
    lastname: 'Hathway',
    email: 'hathway@college.com',
    phone: '+19876543210',
    lastModified: ISODate("2022-09-26T16:05:48.395Z")
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45d"),
    firstname: 'Neal',
    lastname: 'Reardon',
    address: '200 Pavonia Ave',
    city: 'New York',
    country: 'USA'
  },
  {
    _id: ObjectId("6331ca3e70e8dbc65620b45e"),
    firstname: 'Hanna',
    lastname: 'Garcia',
    email: 'garcia@college.edu',
    city: 'San Francisco',
    country: 'USA'
```

Part 4: Programming Assignment

a. **Create a collection called 'games'. Add 5 games to the database.**
   Commands:

```
use gamesdb;
db.games.insertMany(
 [
        {name: "Call of Duty", genre: "Shooter Video Game", rating: 90},
        {name: "Chess", genre: "Abstract Strategy", rating: 85},
        {name: "Minecraft", genre: "Sandbox", rating: 92},
        {name: "Road Rash", genre: "Racing", rating: 75},
        {name: "Temple Run", genre: "Survival", rating: 80}
])
```

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
gamesdb> db.games.insertMany( [
... {name:"Call of Duty", genre: "Shooter Video Game", rating:90},
... {name:"Chess", genre: "Abstract Strategy", rating:85},
... {name:"Minecraft", genre: "Sandbox", rating:92},
... {name:"Road Rash", genre: "Racing", rating:75},
... {name:"Temple Run", genre: "Survival", rating:80}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63320d7470e8dbc65620b464"),
    '1': ObjectId("63320d7470e8dbc65620b465"),
    '2': ObjectId("63320d7470e8dbc65620b466"),
    '3': ObjectId("63320d7470e8dbc65620b467"),
    '4': ObjectId("63320d7470e8dbc65620b468")
  }
}
gamesdb> db.games.find();
[
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b465"),
    name: 'Chess',
    genre: 'Abstract Strategy',
    rating: 85
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b466"),
    name: 'Minecraft',
    genre: 'Sandbox',
    rating: 92
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b467"),
    name: 'Road Rash',
    genre: 'Racing',
    rating: 75
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b468"),
```

b. **Query to return all games**

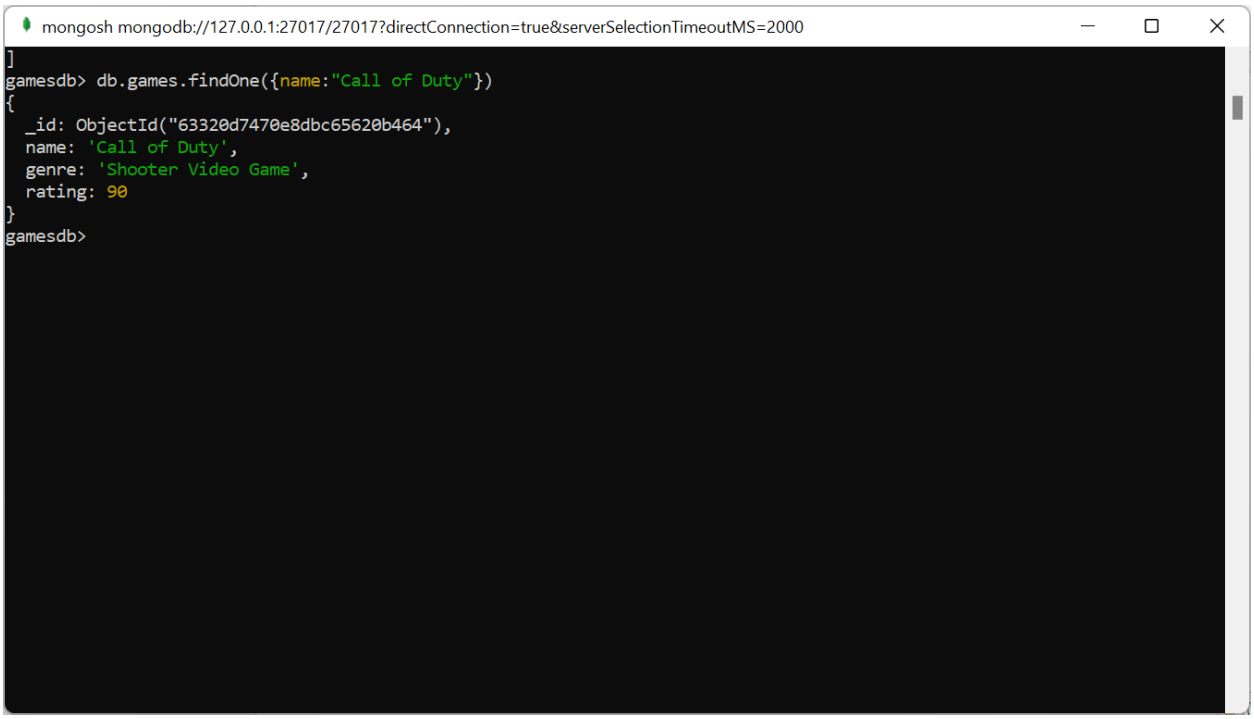
Command: db.games.find();

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
  }
}
gamesdb> db.games.find();
[
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b465"),
    name: 'Chess',
    genre: 'Abstract Strategy',
    rating: 85
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b466"),
    name: 'Minecraft',
    genre: 'Sandbox',
    rating: 92
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b467"),
    name: 'Road Rash',
    genre: 'Racing',
    rating: 75
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b468"),
    name: 'Temple Run',
    genre: 'Survival',
    rating: 80
  }
]
gamesdb>
```

c. **Query to find one of your games by name without using limit().**

Command:

db.games.findOne({name: “Call of Duty”});



c. **Query that returns the 3 highest rated games.**

Command:

db.games.find({}).sort({rating:-1}).limit(3);

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
}
gamesdb> db.games.find({}).sort({rating: -1}).limit(3)
[
  {
    _id: ObjectId("63320d7470e8dbc65620b466"),
    name: 'Minecraft',
    genre: 'Sandbox',
    rating: 92
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b465"),
    name: 'Chess',
    genre: 'Abstract Strategy',
    rating: 85
  }
]
gamesdb>
```

d. Update your two favorite games to have two achievements called 'Game Master' and 'Speed Demon', each under a single key.

Using update()

Command: db.games.updateOne(
{name:"Call of Duty"},
{$set:{achievements:["Game Master", "Speed Demon"]}});

```
gamesdb> db.games.updateOne(
... {name:"Call of Duty"},
... {$set:{achievements:["Game Master", "Speed Demon"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Using replaceOne()

Command: db.games.replaceOne(
{name:”Road Rash”},
{name:”Road Rash”, genre:”Racing”, rating:75,

achievements:[“Game Master”, “Speed Demon”});

```
gamesdb> db.games.replaceOne(
... {name:"Road Rash"},
... {name:"Road Rash", genre:"Racing", rating:75, achievements:["Game Master", "Speed Demon"]});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
gamesdb> db.games.find();
[
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b465"),
    name: 'Chess',
    genre: 'Abstract Strategy',
    rating: 85
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b466"),
    name: 'Minecraft',
    genre: 'Sandbox',
    rating: 92
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b467"),
    name: 'Road Rash',
    genre: 'Racing',
    rating: 75,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b468"),
    name: 'Temple Run',
    genre: 'Survival',
    rating: 80
  }
]
```

e. Query that returns all the games that have both the 'Game Master' and the 'Speed Demon' achievements.

Command: db.games.find({$and:[{achievements:"Game Master"},
{achievements:"Speed Demon"}]})

```
gamesdb> db.games.find({$and:[{achievements:"Game Master"},{achievements:"Speed Demon"}]})
[
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b467"),
    name: 'Road Rash',
    genre: 'Racing',
    rating: 75,
    achievements: [ 'Game Master', 'Speed Demon' ]
  }
]
```

f. Query that returns only games that have achievements.

Command: db.games.find({achievements:{$exists:true}});

```
gamesdb> db.games.find({achievements:{$exists:true}});
[
  {
    _id: ObjectId("63320d7470e8dbc65620b464"),
    name: 'Call of Duty',
    genre: 'Shooter Video Game',
    rating: 90,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("63320d7470e8dbc65620b467"),
    name: 'Road Rash',
    genre: 'Racing',
    rating: 75,
    achievements: [ 'Game Master', 'Speed Demon' ]
  }
]
```

Part 5: Programming Assignment

a. help

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
  }
]
ContactManagementSystem> help

  Shell Help:

    use                                     Set current database
    show                                    'show databases'/'show dbs': Print a list of all available databases.
                                            'show collections'/'show tables': Print a list of all collections for current database.
                                            'show profile': Prints system.profile information.
                                            'show users': Print a list of all users for current database.
                                            'show roles': Print a list of all roles for current database.
                                            'show log <type>': log for current connection, if type is not set uses 'global'
                                            'show logs': Print all logs.

    exit                                    Quit the MongoDB shell with exit/exit()/.exit
    quit                                    Quit the MongoDB shell with quit/quit()
    Mongo                                   Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optional])
    connect                                 Create a new connection and return the Database object. Usage: connect(URI, username [optional], password [optional])
    it                                      result of the last line evaluated; use to further iterate
    version                                 Shell version
    load                                    Loads and runs a JavaScript file into the current shell environment
    enableTelemetry                         Enables collection of anonymous usage data to improve the mongosh CLI
    disableTelemetry                        Disables collection of anonymous usage data to improve the mongosh CLI
    passwordPrompt                          Prompts the user for a password
    sleep                                   Sleep for the specified number of milliseconds
    print                                   Prints the contents of an object to the output
    printjson                               Alias for print()
    cls                                     Clears the screen like console.clear()
    isInteractive                           Returns whether the shell will enter or has entered interactive mode

  For more information on usage: https://docs.mongodb.com/manual/reference/method
ContactManagementSystem>
```

i. show dbs;

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

  For more information on usage: https://docs.mongodb.com/manual/reference/method
ContactManagementSystem> show dbs;
ContactManagementSystem  72.00 KiB
admin                    40.00 KiB
config                   72.00 KiB
local                    72.00 KiB
todolistdb               72.00 KiB
ContactManagementSystem>
```

ii. print(“Hello World!”);

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
                                          'show profile': Prints system.profile information.
                                          'show users': Print a list of all users for current database.
                                          'show roles': Print a list of all roles for current database.
                                          'show log <type>': log for current connection, if type is not set uses 'g
lobal'
                                          'show logs': Print all logs.

    exit                                  Quit the MongoDB shell with exit/exit()/.exit
    quit                                  Quit the MongoDB shell with quit/quit()
    Mongo                                 Create a new connection and return the Mongo object. Usage: new Mongo(URI
, options [optional])
    connect                               Create a new connection and return the Database object. Usage: connect(UR
I, username [optional], password [optional])
    it                                    result of the last line evaluated; use to further iterate
    version                               Shell version
    load                                  Loads and runs a JavaScript file into the current shell environment
    enableTelemetry                       Enables collection of anonymous usage data to improve the mongosh CLI
    disableTelemetry                      Disables collection of anonymous usage data to improve the mongosh CLI
    passwordPrompt                        Prompts the user for a password
    sleep                                 Sleep for the specified number of milliseconds
    print                                 Prints the contents of an object to the output
    printjson                             Alias for print()
    cls                                   Clears the screen like console.clear()
    isInteractive                         Returns whether the shell will enter or has entered interactive mode

  For more information on usage: https://docs.mongodb.com/manual/reference/method
27017> print("Hello World!");
Hello World!

27017>
```

iii. cls;

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

27017>
```

iv. use gamesdb;

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

  For more information on usage: https://docs.mongodb.com/manual/reference/method
27017> use gamesdb;
switched to db gamesdb
gamesdb>
```

v. passwordPrompt()

```
gamesdb> db.createUser({
... user:"user1",
... pwd: passwordPrompt(),
... roles:["readWrite"]
... })
Enter password
******{ ok: 1 }
```

b. db.help();

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000

gamesdb> db.help();

  Database Class:

    getMongo                                    Returns the current database connection
    getName                                     Returns the name of the DB
    getCollectionNames                          Returns an array containing the names of all collections in the current d
atabase.
    getCollectionInfos                          Returns an array of documents with collection information, i.e. collectio
n name and options, for the current database.
    runCommand                                  Runs an arbitrary command on the database.
    adminCommand                                Runs an arbitrary command against the admin database.
    aggregate                                   Runs a specified admin/diagnostic pipeline which does not require an unde
rlying collection.
    getSiblingDB                                Returns another database without modifying the db variable in the shell e
nvironment.
    getCollection                               Returns a collection or a view object that is functionally equivalent to
using the db.<collectionName>.
    dropDatabase                                Removes the current database, deleting the associated data files.
    createUser                                  Creates a new user for the database on which the method is run. db.create
User() returns a duplicate user error if the user already exists on the database.
    updateUser                                  Updates the user's profile on the database on which you run the method. A
n update to a field completely replaces the previous field's values. This includes updates to the user's roles array.
    changeUserPassword                          Updates a user's password. Run the method in the database where the user
is defined, i.e. the database you created the user.
    logout                                      Ends the current authentication session. This function has no effect if t
he current session is not authenticated.
    dropUser                                    Removes the user from the current database.
    dropAllUsers                                Removes all users from the current database.
```

i. db.getCollectionNames();

```
gamesdb> db.getCollectionNames();
[ 'games' ]
gamesdb> show dbs;
ContactManagementSystem   72.00 KiB
admin                     40.00 KiB
config                   108.00 KiB
gamesdb                   60.00 KiB
local                     72.00 KiB
todolistdb                72.00 KiB
gamesdb>
```

ii. db.getCollectionInfos();

```
gamesdb> db.getCollectionInfos();
[
  {
    name: 'games',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: UUID("e73703c4-0128-4ce9-a525-73acce7c3fdc")
    },
    idIndex: { v: 2, key: { _id: 1 }, name: '_id_' }
  }
]
```

iii. db.dropDatabase();

```
gamesdb> use todolistdb;
switched to db todolistdb
todolistdb> db.dropDatabase();
{ ok: 1, dropped: 'todolistdb' }
todolistdb> show dbs;
ContactManagementSystem   72.00 KiB
admin                     40.00 KiB
config                   108.00 KiB
gamesdb                   60.00 KiB
local                     72.00 KiB
todolistdb>
```

iv. db.version();

```
todolistdb> use gamesdb;
switched to db gamesdb
gamesdb> db.version();
6.0.1
gamesdb>
```

v. db.getProfilingStatus();

```
gamesdb> db.getProfilingStatus();
{ was: 0, slowms: 100, sampleRate: 1, ok: 1 }
gamesdb>
```

c. db.mycoll.help()

```
mongosh mongodb://127.0.0.1:27017/27017?directConnection=true&serverSelectionTimeoutMS=2000
Shell Help:

  use                                       Set current database
  show                                      'show databases'/'show dbs': Print a list of all available databases.
                                            'show collections'/'show tables': Print a list of all collections for current database.
                                            'show profile': Prints system.profile information.
                                            'show users': Print a list of all users for current database.
                                            'show roles': Print a list of all roles for current database.
                                            'show log <type>': log for current connection, if type is not set uses 'global'
                                            'show logs': Print all logs.

  exit                                      Quit the MongoDB shell with exit/exit()/.exit
  quit                                      Quit the MongoDB shell with quit/quit()
  Mongo                                     Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optional])
  connect                                   Create a new connection and return the Database object. Usage: connect(URI, username [optional], password [optional])
  it                                        result of the last line evaluated; use to further iterate
  version                                   Shell version
  load                                      Loads and runs a JavaScript file into the current shell environment
  enableTelemetry                           Enables collection of anonymous usage data to improve the mongosh CLI
  disableTelemetry                          Disables collection of anonymous usage data to improve the mongosh CLI
  passwordPrompt                            Prompts the user for a password
  sleep                                     Sleep for the specified number of milliseconds
  print                                     Prints the contents of an object to the output
  printjson                                 Alias for print()
  cls                                       Clears the screen like console.clear()
  isInteractive                             Returns whether the shell will enter or has entered interactive mode

 For more information on usage: https://docs.mongodb.com/manual/reference/method
27017> db.mycoll.help()

 Collection Class:

  aggregate                                 Calculates aggregate values for the data in a collection or a view.
  bulkWrite                                 Performs multiple write operations with controls for order of execution.
  count                                     Returns the count of documents that would match a find() query for the collection or view.
  countDocuments                            Returns the count of documents that match the query for a collection or view.
  deleteMany                                Removes all documents that match the filter from a collection.
  deleteOne                                 Removes a single document from a collection.
  distinct                                  Finds the distinct values for a specified field across a single collection or view and returns the results in an array.
  estimatedDocumentCount                    Returns the count of all documents in a collection or view.
  find                                      Selects documents in a collection or view.
  findAndModify                             Modifies and returns a single document.
  findOne                                   Selects documents in a collection or view.
  renameCollection                          Renames a collection.
27017>
```

i. insertOne()

```
gamesdb> use sampleDb;
switched to db sampleDb
sampleDb> db.users.insertOne({name:"ABC", age:23, gender: "M"});
{
  acknowledged: true,
  insertedId: ObjectId("6333052d1526d755d78a7a03")
}
```

ii. insertMany()

```
sampleDb> db.users.insertMany([{name:"DEF", age:20, gender:"F"},
... {name:"GHI", age: 15, gender: "M"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("633305eb1526d755d78a7a04"),
    '1': ObjectId("633305eb1526d755d78a7a05")
  }
}
```

iii. find()

```
sampleDb> db.users.find();
[
  {
    _id: ObjectId("6333052d1526d755d78a7a03"),
    name: 'ABC',
    age: 23,
    gender: 'M'
  },
  {
    _id: ObjectId("633305eb1526d755d78a7a04"),
    name: 'DEF',
    age: 20,
    gender: 'F'
  },
  {
    _id: ObjectId("633305eb1526d755d78a7a05"),
    name: 'GHI',
    age: 15,
    gender: 'M'
  }
]
```

iv. findOne()

```
sampleDb> db.users.findOne({name:"ABC"});
{
  _id: ObjectId("6333052d1526d755d78a7a03"),
  name: 'ABC',
  age: 23,
  gender: 'M'
}
```

v. deleteOne()

```
sampleDb> db.users.deleteOne({name:"ABC"})
{ acknowledged: true, deletedCount: 1 }
sampleDb> db.users.find();
[
  {
    _id: ObjectId("633305eb1526d755d78a7a04"),
    name: 'DEF',
    age: 20,
    gender: 'F'
  },
  {
    _id: ObjectId("633305eb1526d755d78a7a05"),
    name: 'GHI',
    age: 15,
    gender: 'M'
  }
]
```

vi. replaceOne()

```
sampleDb> db.users.replaceOne({name:"GHI"},
... {name:"Colleen", age: 17, gender:"F"});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sampleDb> db.users.find();
[
  {
    _id: ObjectId("633305eb1526d755d78a7a05"),
    name: 'Colleen',
    age: 17,
    gender: 'F'
  },
  {
    _id: ObjectId("633308041526d755d78a7a06"),
    name: 'Ann',
    age: 20,
    gender: 'F'
  },
  {
    _id: ObjectId("633308041526d755d78a7a07"),
    name: 'Bob',
    age: 26,
    gender: 'M'
  }
]
```

vii. estimatedDocumentCount()

```
sampleDb> db.users.estimatedDocumentCount({});
3
sampleDb>
```

viii. renameCollection()

```
sampleDb> db.users.renameCollection("customers");
{ ok: 1 }
sampleDb> show collections;
customers
sampleDb>
```

ix. updateOne()

```
sampleDb> db.customers.updateOne({name:"Ann"},
... {$set:{age:22}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
sampleDb> db.customers.find();
[
  {
    _id: ObjectId("633305eb1526d755d78a7a05"),
    name: 'Colleen',
    age: 17,
    gender: 'F'
  },
  {
    _id: ObjectId("633308041526d755d78a7a06"),
    name: 'Ann',
    age: 22,
    gender: 'F'
  },
  {
    _id: ObjectId("633308041526d755d78a7a07"),
    name: 'Bob',
    age: 26,
    gender: 'M'
  }
]
```

x. drop()

```
sampleDb> db.customers.drop();
true
sampleDb> show collections;
```