
# MongoDB Commands #

# Comparison Between Excel And MySQL

1.Excel:

- It is software
- It is MS software
- It manage the workbooks
- Workbook data is store in our hard disk

# Chain:

- Workbook -> Sheet -> Row -> column

# Flow:

Excel - Multiple Workbooks
Workbook - Multiple Sheets
Sheet - Multiple Rows
Row - Multiple columns

1.MySQL:

- It is software
- It is Database software - RDBMS
- It manages the databases
- Database data is store in our hard disk

# Chain:

- Database -> Table -> Row -> column

# Flow:

MySQL - Multiple Database
Database - Multiple Tables
Table - Multiple Rows
Row - Multiple columns

# Link :

https://www.codewithharry.com/blogpost/mongodb-cheatsheet/

# Database Commands #

1.View all databases

- show dbs OR show databases

# Note: ';' is optional

2.Create a new or switch databases

- use dbName

# Ex: use Suraj-Blog

3.View current Database

- db

4.Delete Database

- db.dropDatabase()

# Collection Commands #

```
    # Consider one collection -> comments
    # collection_name = comments
  ---------------------------------------

1.Show Collections
    - show collections

2.Create a new collection
  - db.createCollection('collection_name')
  # Ex: db.createCollection('comments')

3.Delete any collection
     - db.collection_name.drop()
     # Ex: db.comments.drop()



----------------------
  # Row Commands #
----------------------


1.Show all Rows in a Collection
   - db.comments.find()

2.Show all Rows in a Collection (Prettified)
   - db.comments.find().pretty()
   # Note:
     - It shows all rows in beautiful manner
     - In the form of JavaScript object
```

3.Find the first row matching the object

```
 - db.comments.findOne({name: 'Suraj'})
```

# Note:

- After finding the matching row it will stop searching

4.Insert One Row : (single JS object)

```
   - db.comments.insert({
         'name': 'Suraj',
         'lang': 'JavaScript',
         'year': 5
        })
```

5.Insert many Rows : (Array of objects)

```
  - db.comments.insertMany([
       {
       'name': 'Suraj',
       'lang': 'JavaScript',
       'year': 5
       },
       {'name': 'Rohan',
       'lang': 'Python',
       'year': 3
       },
       {'name': 'Lovish',
       'lang': 'Java',
       'year': 4
       }
    ])
```

6.Insert a Row with one extra field

```
   - db.comments.insert({
        'name': 'Suraj',
        'lang': 'JavaScript',
        'year': 5,
        'date': new Date()
       })
```

# Note:

- Extra field is 'date'
- Insert a row without following a schema
- We can insert an object which don't follow the schema

7.Search any row in collection

- db.comments.find({lang:'Python'})

OR

- db.comments.find({lang:'Python' , name: 'Rohan'})

# Note:

- By using JavaScript object
- Just by using one of the field of that object

8.Limit the number of rows in output

- db.comments.find().limit(2)

# Note:

- It will show first two rows

9.Count the number of rows in the output

- db.comments.find().count() => 5

OR

- db.comments.find().limit(2).count() => 5 and not 2

OR

- db.comments.find({name: 'Suraj'}).count() => 3

10.Sort the Rows in ascending or descending order on the basis of any field

- `db.comments.find().sort({year:1}).pretty()` => 1 means ascending

OR

- `db.comments.find().sort({year:-1}).pretty()` => -1 means descending

11.Update a Row

1.Update a row if match

```
  - db.comments.update({name: 'Suraj'},
     {
        'name': 'Suraj',
        'lang': 'JavaScript',
        'year': 50
     })
```

# Note: If match then modified but not upserted

2.Update a row if not match

```
  - db.comments.update({name: 'Suraj2'},
     {
        'name': 'Suraj',
        'lang': 'JavaScript',
        'year': 50
     })
```

# Note: If not match then not modified and not upserted

3.Update a row if not match

```
  - db.comments.update({name: 'Shubham'},
        {
          'name': 'Suraj',
          'lang': 'JavaScript',
          'year': 50
        }, {upsert: true})
```

# Note: If not match then not modified but upserted

4.Update a row if not match with update Operators

```
  - db.comments.updateOne({name: 'Shubham'},
      {$set: {'name': 'Harry',
          'lang': 'JavaScript',
          'year': 51
        }}, {upsert: true})
```

# Note: It same as above

# Note:

- Upsert means (create + Insert)
- If object is not present then no match is there so it will be created and inserted
- By default no insertion only updation

12.Delete a Row

- `db.comments.remove({name: 'Suraj'})`

# Note:

- It will remove all the rows with matching name field

13.Mongodb Update Operators

1.Mongodb Increment Operator

```
        - db.comments.update({name: 'Rohan'},
           {$inc:{
             year: 2
            }})
```

# Note:

- Increment the value of field
- year: 3 -> year: 5
- field with data type integer values

2.Mongodb Rename Operator

```
        - db.comments.update({name: 'Rohan'},
           {$rename:{
              year: 'member'
            }})
```

# Note:

- It updates the actual field
- It rename the field
- year -> member

14.Inequality:

# Less than/Greater than/ Less than or Equal/Greater than or Equal

```
- db.comments.find({year: {$lt: 5}})
```

```
- db.comments.find({year: {$lte: 4}})
```

```
- db.comments.find({year: {$gt: 3}})
```

-
  ```
  db.comments.find({year: {$gte: 5}})
  ```

Done…..!!