**Mongo DB**

|  |  |  |
| --- | --- | --- |
| Description | Command |  |
| Show All Databases | show dbs |  |
| Show Current Database | db |  |
| Create Or Switch Database | Use <db\_name> |  |
| Drop | db.dropDatabase() |  |
| Create Collection | db.createCollection('posts') |  |
| Create User | db.createUser({  user : “user1”,  pwd : “123123”,  roles : [“readWrite”,”dbAdmin”]  }) |  |
| Create Collection | db.createCollection(‘posts’) |  |
| Show Collections | show collections |  |
| Insert Row | db.posts.insert({  title: 'Post One',  body: 'Body of post one',  category: 'News',  tags: ['news', 'events'],  user: {  name: 'John Doe',  status: 'author'  },  date: Date()  }) |  |
| Insert Multiple Rows | db.posts.insertMany([  {  title: 'Post Two',  body: 'Body of post two',  category: 'Technology',  date: Date()  },  {  title: 'Post Three',  body: 'Body of post three',  category: 'News',  date: Date()  },  {  title: 'Post Four',  body: 'Body of post three',  category: 'Entertainment',  date: Date()  }  ]) |  |
| Get All Rows | db.posts.find() |  |
| Get All Rows Formatted | db.find().pretty() |  |
| Find Rows | db.posts.find({ category: 'News' }) |  |
| Sort Rows | db.posts.find().sort({ title: 1 }).pretty() | Default sort is ascending |
| desc sort | db.posts.find().sort({ title: -1 }).pretty() |  |
| Count Rows | db.posts.find().count() |  |
| db.posts.find({ category: 'news' }).count() | Here we are counting by category news. |
| Limit Rows | db.posts.find().limit(2).pretty() |  |
| Limit and sort with formatted output(Chaining) | db.posts.find().limit(2).sort({ title: 1 }).pretty() |  |
| Foreach loop | db.posts.find().forEach(function(doc) {  print("Blog Post: " + doc.title)  }) | It will print doc title for all the row |
| Find one row | db.posts.findOne({ category: 'News' }) |  |
| Find specific fields | db.posts.find({ title: 'Post One' }, {  title: 1,  author: 1  }) |  |
|  | db.posts.find({  “user.name” : “joe smith”  }) |  |
|  | db.posts.find({  $or : [{title : 'Post One'},{title : 'Post 1'}]  }) |  |
| Update row | db.posts.update({ title: 'Post Two' },  {  title: 'Post Two',  body: 'New body for post 2',  date: Date()  },  {  upsert: true  }) |  |
| Update Specific Field | db.posts.update({ title: 'Post Two' },  {  $set: {  body: 'Body for post 2',  category: 'Technology'  }  }) | Here we are updating body and category for title “Post Two” |
| Increment Field ($inc) | db.posts.update({ title: 'Post Two' },  {  $inc: {  likes: 5  }  }) | This will increase the likes by 5. Let say likes are 25 then it will become 30 after executing this command. |
| Rename Field | db.posts.update({ title: 'Post Two' },  {  $rename: {  likes: 'views'  }  }) | Likes column will rename as views. |
| Delete Row | db.posts.remove({ title: 'Post Four' }) |  |
|  | db.posts.remove({"title":"Post Four"},{justOne:true}) | Delete only first row if there are multiple |
| Sub-Documents | db.posts.update({ title: 'Post One' },  {  $set: {  comments: [  {  body: 'Comment One',  user: 'Mary Williams',  date: Date()  },  {  body: 'Comment Two',  user: 'Harry White',  date: Date()  }  ]  }  }) |  |
| Find By Element in Array ($elemMatch) | db.posts.find({  comments: {  $elemMatch: {  user: 'Mary Williams'  }  }  }) |  |
| Add Index | db.posts.createIndex({ title: 'text' }) |  |
| Text Search | db.posts.find({  $text: {  $search: "\"Post O\""  }  }) |  |
| Comparisons ( Greater/Greater equal & Less than/ Less than equal) | db.posts.find({ views: { $gt: 2 } })  db.posts.find({ views: { $gte: 7 } })  db.posts.find({ views: { $lt: 7 } })  db.posts.find({ views: { $lte: 7 } }) |  |
| set | db.posts.update({  “Title” : “Post Two”,  {$set : {  “Additional\_info” : “Text info”  }  }  }) |  |
| unset | db.posts.update({  {$unset : {tilte}}  }) | Remove field from document |