MongoDB Database Cheatsheet

Author: Placements Simplified Editorial Team

Introduction: The NoSQL Document Database

MongoDB is a document-oriented, NoSQL database that stores data in flexible, JSON-like BSON documents. Unlike relational databases, MongoDB does not enforce fixed schemas, making it perfect for rapid scaling and hierarchical data models. Below are high-frequency MongoDB commands for your reference.

1. Database Level Commands

View all databases:

```
show dbs
```

Create or switch to a database:

```
use dbName
```

View current active database:

```
db
```

Delete current database:

```
db.dropDatabase()
```

2. Collection Commands

Show collections in active database:

```
show collections
```

Create a new collection:

```
db.createCollection('comments')
```

Drop an existing collection:

```
db.comments.drop()
```

3. Document (Row) Operations

Show all documents in a collection:

```
db.comments.find()
```

Show documents formatted cleanly (pretty):

```
db.comments.find().pretty()
```

Find the first document matching a filter:

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

Insert a single document:

```
db.comments.insertOne({
  name: 'Harry',
  lang: 'JavaScript',
  member_since: 5
})
```

Insert multiple documents:

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

4. Document Query Modifiers

Limit result size:

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

Skip results (useful for pagination):

```
db.comments.find().skip(2).limit(2)
```

Count matching documents:

```
db.comments.countDocuments({lang: 'Python'})
```

5. Update Commands

Update a single document (or upsert if not exists):

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

Update multiple documents concurrently:

```
db.comments.updateMany(
  {lang: 'JavaScript'},
  {$set: {verified: true}}
)
```

Increment numeric fields:

```
db.comments.updateOne(
  {name: 'Rohan'},
  {$inc: {member_since: 2}}
)
```

Rename an document attribute:

```
db.comments.updateOne(
  {name: 'Rohan'},
  {$rename: {member_since: 'member'}}
)
```

6. Query Operators

Comparison Operators:

```
db.comments.find({member_since: {$lt: 90}}) // Less Than
db.comments.find({member_since: {$lte: 90}}) // Less Than or Equal
db.comments.find({member_since: {$gt: 90}}) // Greater Than
db.comments.find({member_since: {$gte: 90}}) // Greater Than or Equal
db.comments.find({member_since: {$ne: 5}}) // Not Equal
```

Logical Operators:

```
db.comments.find({$and: [{lang: 'Python'}, {member_since: {$gt: 2}}]})
db.comments.find({$or: [{lang: 'Python'}, {lang: 'Java'}]})
db.comments.find({lang: {$in: ['Python', 'Java']}})
db.comments.find({lang: {$nin: ['C++', 'Rust']}})
```

7. Indexes & Aggregations

Create database index:

```
db.comments.createIndex({name: 1})
```

List all collection indexes:

```
db.comments.getIndexes()
```

Drop an index:

```
db.comments.dropIndex({name: 1})
```

Aggregation - Group and Count:

```
db.comments.aggregate([
  {$group: {_id: "$lang", total: {$sum: 1}}}
])
```

Aggregation - Average by field value:

```
db.comments.aggregate([
  {$group: {_id: "$lang", avgExperience: {$avg: "$member_since"}}}
])
```