MongoDB Cheatsheet

Basic Concepts

- **Database**: Holds collections.
- **Collection**: A set of documents (similar to a table in relational databases).
- **Document**: A JSON-like object with fields and values (equivalent to rows in relational databases).
- **Field**: A key-value pair within a document (equivalent to columns in relational databases).

1. Database and Collection Management

- **List Databases**:

```
show dbs
```

- **Create or Switch to Database**:

```
use myDatabase
```

- **Show Collections in a Database**:

```
show collections
```

- **Create or Drop Collection**:

```
db.createCollection('myCollection')
db.myCollection.drop()
```

2. Basic CRUD Operations

Create Documents

- **Insert a Single Document**:

```
db.myCollection.insertOne({ name: 'Alice', age: 25, city: 'Pretoria' })
```

- **Insert Multiple Documents**:

```
db.myCollection.insertMany([
  { name: 'Bob', age: 30, city: 'Johannesburg' },
  { name: 'Carol', age: 27, city: 'Cape Town' }
])
```

Read Documents

- **Find All Documents**:

```
db.myCollection.find()
```

- **Find Documents with Filters**:

```
db.myCollection.find({ age: { $gt: 25 } })
```

- **Find One Document**:

```
db.myCollection.findOne({ name: 'Alice' })
```

- **Project Specific Fields**:

```
db.myCollection.find({ city: 'Pretoria' }, { name: 1, age: 1, _id: 0 })
```

Update Documents

- **Update a Single Document**:

```
db.myCollection.updateOne(
  { name: 'Alice' },
  { $set: { age: 26 } }
)
```

- **Update Multiple Documents**:

```
db.myCollection.updateMany(
  { city: 'Pretoria' },
  { $inc: { age: 1 } }
)
```

- **Replace a Document**:

```
db.myCollection.replaceOne(
  { name: 'Alice' },
  { name: 'Alice', age: 26, city: 'Pretoria', occupation: 'Engineer' }
)
```

Delete Documents

- **Delete a Single Document**:

```
db.myCollection.deleteOne({ name: 'Alice' })
```

- **Delete Multiple Documents**:

```
db.myCollection.deleteMany({ age: { $lt: 25 } })
```

3. Advanced Querying Operators

- **Comparison Operators**:
  - Greater Than (`$gt`): `{ age: { $gt: 25 } }`
  - Less Than (`$lt`): `{ age: { $lt: 25 } }`
  - Greater Than or Equal (`$gte`): `{ age: { $gte: 25 } }`
  - Less Than or Equal (`$lte`): `{ age: { $lte: 25 } }`
  - Not Equal (`$ne`): `{ city: { $ne: 'Pretoria' } }`
- **Logical Operators**:
  - **$and**:

```
db.myCollection.find({ $and: [{ age: { $gt: 25 } }, { city: 'Pretoria' }]
})
```

  - **$or**:

```
db.myCollection.find({ $or: [{ city: 'Pretoria' }, { city: 'Cape Town' }]
})
```

  - **$not**:

```
db.myCollection.find({ age: { $not: { $gt: 25 } } })
```

- **Existence Check**:

```
db.myCollection.find({ occupation: { $exists: true } })
```

- **Array Querying**:
  - Match specific value:

```
db.myCollection.find({ tags: 'mongoDB' })
```

  - Match all values:

```
db.myCollection.find({ tags: { $all: ['mongoDB', 'database'] } })
```

- Element match in array:

```
db.myCollection.find({ scores: { $elemMatch: { score: { $gt: 80 } } } })
```

4. Aggregation Pipeline

The aggregation pipeline in MongoDB is a powerful way to process and transform documents in collections.

Pipeline Stages

1. **$match**: Filters documents.

```
db.myCollection.aggregate([
  { $match: { city: 'Pretoria' } }
])
```

2. **$group**: Groups documents by a specified field and applies aggregations.

```
db.myCollection.aggregate([
  { $group: { _id: '$city', total: { $sum: 1 } } }
])
```

3. **$sort**: Sorts documents.

```
db.myCollection.aggregate([
  { $sort: { age: -1 } }
])
```

4. **$project**: Reshapes documents to include or exclude fields.

```
db.myCollection.aggregate([
  { $project: { name: 1, age: 1, city: 1 } }
])
```

5. **$limit** and **$skip**: Limits the number of documents or skips a specified number of documents.

```
db.myCollection.aggregate([
  { $limit: 5 }
])
```

6. **$unwind**: Deconstructs an array field to output a document for each element.

```
db.myCollection.aggregate([
  { $unwind: '$tags' }
```

```
])
```

Common Aggregation Examples

- **Count Documents by a Field**:

```
db.myCollection.aggregate([
  { $group: { _id: '$city', count: { $sum: 1 } } }
])
```

- **Calculate Average Age**:

```
db.myCollection.aggregate([
  { $group: { _id: null, averageAge: { $avg: '$age' } } }
])
```

- **Nested Aggregation with $lookup (Join)**:

```
db.orders.aggregate([
  {
    $lookup: {
      from: 'products',
      localField: 'productId',
      foreignField: '_id',
      as: 'productDetails'
    }
  }
])
```

- **Date Aggregation ($year, $month, etc.)**:

```
db.myCollection.aggregate([
  { $project: { year: { $year: '$dateField' }, month: { $month: '$dateField' }
} }
])
```

5. Indexing and Performance

- **Create an Index**:

```
db.myCollection.createIndex({ age: 1 })
```

- **Compound Index**:

```
db.myCollection.createIndex({ city: 1, age: -1 })
```

- **Text Index for Full-Text Search**:

```
db.myCollection.createIndex({ description: 'text' })
```

- **Remove an Index**:

```
db.myCollection.dropIndex('age_1')
```

- **Analyze Query Performance**:

```
db.myCollection.find({ age: { $gt: 30 } }).explain('executionStats')
```
