MongoDB Command

Prepared By :
Dr Ezzatul Akmal Kamaru-zaman

Operation	Command	Description	Example Syntax
Insert	`insertOne`	Inserts a single document into a collection.	`db.collection.insertOne({ name: "Alice", age: 25 })`
	`insertMany`	Inserts multiple documents into a collection.	`db.collection.insertMany([{ name: "Bob" }, { name: "Charlie" }])`
Read	`find`	Queries documents that match a specified condition; returns a cursor.	`db.collection.find({ age: { $gt: 18 } })`
	`findOne`	Retrieves a single document that matches the condition.	`db.collection.findOne({ name: "Alice" })`
	`find().limit()`	Limits the number of documents returned in a query.	`db.collection.find({ age: { $gte: 18 } }).limit(5)`
Update	`updateOne`	Updates the first document that matches the specified condition.	`db.collection.updateOne({ name: "Alice" }, { $set: { age: 30 } })`
	`updateMany`	Updates all documents that match the specified condition.	`db.collection.updateMany({ active: true }, { $inc: { loginCount: 1 } })`
	`replaceOne`	Replaces the entire document with a new one that matches the condition.	`db.collection.replaceOne({ name: "Alice" }, { name: "Alice", age: 30, active: true })`

Delete	`deleteOne`	Deletes the first document that matches the specified condition.	`db.collection.deleteOne({ name: "Alice" })`
	`deleteMany`	Deletes all documents that match the specified condition.	`db.collection.deleteMany({ age: { $lt: 18 } })`
Aggregation	`aggregate`	Performs data aggregation operations using a pipeline of stages.	`db.collection.aggregate([{ $match: { status: "active" } }, { $group: { _id: "$age", count: { $sum: 1 } } }])`

Sorting	`find().sort()`	Sorts query results by specified fields in ascending or descending order.	`db.collection.find().sort({ age: -1 })`
Indexing	`createIndex`	Creates an index on a field or fields to optimize query performance.	`db.collection.createIndex({ name: 1 })`
Transaction	`startSession` and `startTransaction`	Used for multi-document ACID-compliant transactions.	`const session = client.startSession(); session.startTransaction(); ...session.commitTransaction();`
Counting	`countDocuments`	Counts the documents that match a specified condition.	`db.collection.countDocuments({ active: true })`
Distinct	`distinct`	Finds distinct values for a specified field across documents.	`db.collection.distinct("age")`

Question	TRUE/FALSE
db.users.find({ age: { $gte: 18 } }); This query returns all documents from the users collection where the age field is 18 or older.	
db.users.updateMany({}, { $set: { status: "active" } }); This command sets the status field to "active" for all documents in the users collection, even if they already have a status field.	
db.inventory.find({ "tags.0": "fruit" }) This query finds all documents in the inventory collection where the first element in the tags array is "fruit".	
db.students.deleteOne({ name: "John" }); This command deletes all documents from the students collection where the name field is "John".	

Question	TRUE/FALSE
db.orders.find({ date: { $gte: new Date("2023-01-01") } }); This query returns all documents in the orders collection where the date field is on or after January 1, 2023.	
db.collection.dropIndex("name_1"); This command deletes the index named "name_1" from the collection.	
db.products.aggregate([{ $match: { price: { $gt: 100 } } }]); This aggregation pipeline filters the products collection, returning only documents where the price field is greater than or equal to 100.	
db.users.insertOne({ name: "Alice", age: 25, name: "Bob" } This command will successfully insert a document with both name fields (“Alice” and “Bob”) in the users collection.	

Question	TRUE/FALSE
db.sales.count({ total: { $gte: 500 } }); This command counts all documents in the sales collection where the total field is higher than 500	
db.employees.find({ $or: [{ department: "HR" }, { role: "Manager" }] }); This query finds all documents in the employees collection where either the department is "HR" or the role is "Manager".	

Question	TRUE/FALSE
db.users.find({ age: { $gte: 18 } }); This query returns all documents from the users collection where the age field is 18 or older.	TRUE
db.users.updateMany({}, { $set: { status: "active" } }); This command sets the status field to "active" for all documents in the users collection, even if they already have a status field.	TRUE
db.inventory.find({ "tags.0": "fruit" }) This query finds all documents in the inventory collection where the first element in the tags array is "fruit".	TRUE
db.students.deleteOne({ name: "John" }); This command deletes all documents from the students collection where the name field is "John".	FALSE

Question	TRUE/FALSE
db.orders.find({ date: { $gte: new Date("2023-01-01") } }); This query returns all documents in the orders collection where the date field is on or after January 1, 2023.	TRUE
db.collection.dropIndex("name_1"); This command deletes the index named "name_1" from the collection.	TRUE
db.products.aggregate([{ $match: { price: { $gt: 100 } } }]); This aggregation pipeline filters the products collection, returning only documents where the price field is greater than or equal to 100.	FALSE
db.users.insertOne({ name: "Alice", age: 25, name: "Bob" } This command will successfully insert a document with both name fields (“Alice” and “Bob”) in the users collection.	FALSE (Document fields must have unique keys; duplicate field names are not allowed)

Question	TRUE/FALSE
db.sales.count({ total: { $gte: 500 } }); This command counts all documents in the sales collection where the total field is higher than 500	FALSE
db.employees.find({ $or: [{ department: "HR" }, { role: "Manager" }] }); This query finds all documents in the employees collection where either the department is "HR" or the role is "Manager".	TRUE