

 write mongo db quries with example, write all possible diffrent queries

\$

ChatGPT ~

MongoDB is a NoSQL database that uses JSON-like documents with optional schemas. Here's a comprehensive guide to various MongoDB queries, along with examples for each type.

1. Insert Queries

Insert a Single Document

```
javascript

db.collection.insertOne({
    name: "Alice",
    age: 30,
    city: "New York"
});
```

Insert Multiple Documents

2. Find Queries

Find All Documents

```
javascript ☐ Copy code db.collection.find({});
```

Find Documents with a Condition

```
javascript

db.collection.find({ age: { $gt: 30 } });
```

Find with Specific Fields

```
javascript

db.collection.find({ age: { $gt: 30 } }, { name: 1, city: 1, _id: 0 });
```

Find One Document

3. Update Queries

Update a Single Document

```
javascript

db.collection.updateOne(
    { name: "Alice" },
    { $set: { age: 31 } }
);
```

Update Multiple Documents

```
javascript

db.collection.updateMany(
    { city: "New York" },
    { $set: { state: "NY" } }
);
```

Danings - Daguman

iavascrint fil Conv.cc

```
javascript

db.collection.replaceOne(
    { name: "Alice" },
    { name: "Alice", age: 32, city: "New York", profession: "Engineer" }
);
```

4. Delete Queries

Delete a Single Document

```
javascript ☐ Copy code

db.collection.deleteOne({ name: "Alice" });
```

Delete Multiple Documents

```
javascript

db.collection.deleteMany({ city: "Chicago" });
```

5. Aggregation Queries

Aggregation Pipeline

```
javascript

db.collection.aggregate{[
    { $match: { age: { $gt: 25 } } },
    { $group: { _id: "$city", averageAge: { $avg: "$age" } } }
]);
```

6. Indexing Queries

Create an Index

Find with Index Hint

```
javascript

db.collection.find({ name: "Bob" }).hint({ name: 1 });
```

 \downarrow

7. Distinct Query

Find Distinct Values

```
javascript ☐ Copy code db.collection.distinct("city");
```

8. Text Search Queries

Create Text Index

```
javascript ☐ Copy code

db.collection.createIndex({ description: "text" });
```

Text Search

```
javascript

db.collection.find({ $text: { $search. "engineer" } });
```

9. Geospatial Queries

Insert Geospatial Data

```
javascript

db.collection.insertOne({
   name: "Central Park",
   location: { type: "Point", coordinates: [-73.965355, 40.782865] }
}):
```

Find Near a Location

10. Array Queries

Find Documents with Array Elements

Find Documents with All Elements

```
javascript

db.collection.find({ tags: { $all: ["mongodb", "database"] } });
```

Update Array Elements

11. Advanced Queries

Find with Regular Expression

```
javascript

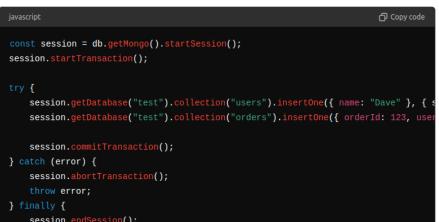
db.collection.find({ name: { $regex: /^A/ } });
```

Conditional Queries

 \downarrow

12. Transactions

Start a Session and Transaction



}

These examples cover a wide range of MongoDB operations. They can be adapted and extended to fit various application needs. If you have specific scenarios in mind, feel free to let me know!



compare mondo db quries with sql queries

\$

Comparing MongoDB queries to SQL queries involves understanding the differences between a NoSQL database like MongoDB and a relational database that uses SQL. Below, I'll illustrate various operations in both MongoDB and SQL to highlight these differences.

1. Database and Collection/Table Setup

MongoDB

- Database: A logical grouping of collections (like tables).
- Collection: A group of documents (like rows).

SQL

- Database: A logical grouping of tables.
- Table: A group of rows.

2. Insert Data

MongoDB

SQL

```
INSERT INTO users (name, age, city) VALUES ('Alice', 30, 'New York');

INSERT INTO users (name, age, city) VALUES
('Bob', 25, 'Los Angeles'),
('Charlie', 35, 'Chicago');
```

3. Select Data

MongoDB

```
javascript

// Find all documents
db.collection.find({});

// Find with a condition
db.collection.find({ age: { $gt: 30 } });

// Find specific fields
db.collection.find({ age: { $gt: 30 } }, { name: 1, city: 1, _id: 0 });
```

SQL

```
sql

-- Select all rows

SELECT * FROM users;

-- Select with a condition

SELECT * FROM users WHERE age > 30;

-- Select specific columns

SELECT name, city FROM users WHERE age > 30;
```

4. Update Data



MongoDB

```
javascript

// Update a single document
db.collection.updateOne({ name: "Alice" }, { $set: { age: 31 } });

// Update multiple documents
db.collection.updateMany({ city: "New York" }, { $set: { state: "NY" } });
```

 $(\mathbf{\Psi}$

SQL

```
sql

-- Update a single row

UPDATE users SET age = 31 WHERE name = 'Alice';

-- Update multiple rows

UPDATE users SET state = 'NY' WHERE city = 'New York';
```

5. Delete Data

MongoDB

```
javascript

// Delete a single document
db.collection.deleteOne({ name: "Alice" });

// Delete multiple documents
db.collection.deleteMany({ city: "Chicago" });
```

SQL

6. Aggregation

MongoDB

SQL

```
sql Copy code

SELECT city, AVG(age) as averageAge
FROM users
WHERE age > 25
GROUP BY city;
```

7. Indexes

MongoDB

```
javascript

// Create an index
db.collection.createIndex({ name: 1 });

// Use index hint
db.collection.find({ name: "Bob" }) \ t({ name: 1 });
```

```
sql

-- Create an index

CREATE INDEX idx_name ON users (name);
```

8. Joins

MongoDB

MongoDB doesn't support joins in the traditional sense but can perform similar operations using aggregation and `\$lookup`.

SQL

9. Transactions

MongoDB

```
javascript

const session = db.getMongo().startSession();
session.startTransaction();
try {
    session.getDatabase("test").collection("users").insertOne({ name: "Dave" }, { session.getDatabase("test").collection("orders").insertOne({ orderId: 123, user session.commitTransaction();
} catch (error) {
    session.abortTransaction();
    throw error;
} finally {
    session.endSession();
}
```

SQL

```
BEGIN;

INSERT INTO users (name) VALUES ('Dave');
INSERT INTO orders (orderId, user) VALUES (123, 'Dave');

COMMIT;
```

Key Differences

- 1. Schema:
 - MongoDB: Schema-less, allowing flexible document structures.
 - SQL: Fixed schema, requiring a predefined table structure.
- 2. Data Model:
 - MongoDB: Document-oriented, storing data in JSON-like BSON documents.
 - SQL: Relational, storing data in tables with rows and columns.
- 3. Joins:

- MongoDB: Limited support via `\$lookup` in aggregation.
- **SQL**: Full join support with `**JOIN**` operations.

4. Transactions:

- MongoDB: Supports ACID transactions in multi-document operations.
- **SQL**: Traditional ACID transaction support.

5. Indexing

• Both support indexing, but methods and optimizations differ.

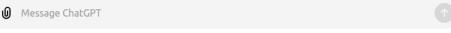
6. Scalability:

- MongoDB: Designed for horizontal scaling with sharding.
- SQL: Typically scaled vertically, though some databases support horizontal scaling.

These examples and comparisons illustrate \checkmark MongoDB and SQL handle various database operations differently due to their underlying architectures.







ChatGPT can make mistakes. Check important info.