Assignment-2

Create, update, delete commands in mysql

Assuming we have a table named "users" with the following columns: "id" (primary key), "name" (varchar), "age" (int), and "email" (varchar).

1. Create Table:

To create a table "users," use the following query:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    email VARCHAR(100)
);
```

2. Insert Data:

To insert data into the "users" table, you can use the INSERT INTO command:

```
INSERT INTO users (name, age, email)
VALUES ('John Doe', 30, 'john.doe@example.com');

INSERT INTO users (name, age, email)
VALUES ('Jane Smith', 25, 'jane.smith@example.com');
```

3. Update Data:

To update existing data in the "users" table, you can use the UPDATE command:

```
UPDATE users
SET age = 31, email = 'john.doe.updated@example.com'
WHERE id = 1;
```

This query will update the age and email for the user with id = 1.

4. Delete Data:

To delete data from the "users" table, you can use the DELETE command:

```
DELETE FROM users
WHERE id = 2;
```

This query will delete the user with id = 2 from the table.

Please be cautious when using DELETE as it permanently removes data from the table, and there's no way to recover it once deleted. Make sure to have proper backups and confirm the DELETE condition before executing the query.

create tables and perform joins in mysql

1. Create Tables:

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    emp_age INT,
    department_id INT
);

CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);
```

2. Insert Data:

Let's insert some data into the tables:

```
INSERT INTO employees (emp_id, emp_name, emp_age, department_id)
VALUES (1, 'John Doe', 30, 1);

INSERT INTO employees (emp_id, emp_name, emp_age, department_id)
VALUES (2, 'Jane Smith', 25, 2);

INSERT INTO departments (department_id, department_name)
VALUES (1, 'HR');

INSERT INTO departments (department_id, department_name)
VALUES (2, 'IT');
```

3. Perform Joins:

Now, let's perform different types of joins between the "employees" and "departments" tables.

a. INNER JOIN:

```
SELECT employees.emp_id, emp_name, emp_age, department_name
FROM employees
INNER JOIN departments ON employees.department_id =
departments.department_id;
```

This query will return only the rows where there is a match between the "department_id" column in the "employees" table and the "department_id" column in the "departments" table.

b. LEFT JOIN (or LEFT OUTER JOIN):

```
SELECT employees.emp_id, emp_name, emp_age, department_name
FROM employees
LEFT JOIN departments ON employees.department_id =
departments.department_id;
```

This query will return all rows from the "employees" table and the matching rows from the "departments" table. If there is no match in the "departments" table, the result will contain NULL values for the "department_name" column.

c. RIGHT JOIN (or RIGHT OUTER JOIN):

```
SELECT employees.emp_id, emp_name, emp_age, department_name
FROM employees
RIGHT JOIN departments ON employees.department_id =
departments.department_id;
```

This query will return all rows from the "departments" table and the matching rows from the "employees" table. If there is no match in the "employees" table, the result will contain NULL values for the "emp_id," "emp_name," and "emp_age" columns.

d. FULL OUTER JOIN:

```
SELECT employees.emp_id, emp_name, emp_age, department_name
FROM employees
FULL OUTER JOIN departments ON employees.department_id =
departments.department_id;
```

Note: MySQL doesn't directly support FULL OUTER JOIN, but you can achieve it using UNION:

```
SELECT employees.emp_id, emp_name, emp_age, department_name
```

```
FROM employees
LEFT JOIN departments ON employees.department_id =
departments.department_id
UNION
SELECT employees.emp_id, emp_name, emp_age, department_name
FROM employees
RIGHT JOIN departments ON employees.department_id =
departments.department_id;
```

These are some of the common join types in MySQL. Depending on your data and requirements, you can choose the appropriate join to retrieve the desired results.

Create, update, delete commands in mongo

Sure! Below are examples of how to create, update, and delete data in MongoDB using the `db.collection` methods.

Assuming we have a collection named "users" with documents having the following structure:

```
{
  "_id": ObjectId("615723356b03a0a2f5b2d2e6"),
  "name": "John Doe",
  "age": 30,
  "email": "john.doe@example.com"
}
```

1. Create Data:

To insert data into the "users" collection, you can use the `insertOne()` or `insertMany()` method:

```
// Insert a single document
db.users.insertOne({
  "name": "Jane Smith",
  "age": 25,
  "email": "jane.smith@example.com"
});

// Insert multiple documents at once
db.users.insertMany([
  {
    "name": "Alice",
    "age": 28,
    "email": "alice@example.com"
  },
  {
    "name": "Bob",
```

```
    "age": 32,
    "email": "bob@example.com"
  }
]);
```

2. Update Data:

To update existing data in the "users" collection, you can use the `updateOne()` or `updateMany()` method:

```
// Update a single document
db.users.updateOne(
  { "_id": ObjectId("615723356b03a0a2f5b2d2e6") },
  { $set: { "age": 31, "email": "john.doe.updated@example.com" } }
);

// Update multiple documents that match a certain condition
db.users.updateMany(
  { "age": { $gte: 30 } },
  { $set: { "status": "adult" } }
);
```

3. Delete Data:

To delete data from the "users" collection, you can use the `deleteOne()` or `deleteMany()` method:

```
// Delete a single document
db.users.deleteOne({ "_id": ObjectId("615723356b03a0a2f5b2d2e6") });

// Delete multiple documents that match a certain condition
db.users.deleteMany({ "age": { $lt: 30 } });
```

These are basic examples of CRUD operations (Create, Read, Update, Delete) in MongoDB. Remember to exercise caution when performing delete operations as they permanently remove data from the collection. Always make sure to have proper backups and confirm the delete condition before executing the query.