SMARTBRIDGE – SUMMER EXTERNSHIP

MODERN APPLICATION DEVELOPMENT (JAVA SPRING BOOT)

WEEK 2 ASSIGNMENT

1. Create, Delete and Update commands in MySQL

CODE:

```
-- Create table
CREATE TABLE employees (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50) NOT NULL,
  age INT,
  department VARCHAR(50)
);

-- Insert data
INSERT INTO employees (name, age, department)
VALUES ('John Doe', 30, 'IT'),
       ('Jane Smith', 28, 'HR'),
       ('Mark Johnson', 35, 'Sales');

-- Select all records
SELECT * FROM employees;

-- Update record
UPDATE employees
SET age = 32
WHERE id = 1;

-- Delete record
DELETE FROM employees
WHERE id = 3;
```

EXPLANATION:

First, a table called "employees" is created with columns for id, name, age, and department. Then, sample data is inserted into the table using the INSERT INTO statement.

To demonstrate the update command, the age of an employee with id 1 is modified using the UPDATE statement. The SET clause specifies the column to update and its new value, and the WHERE clause identifies the specific record to update.

For the delete command, the employee with id 3 is removed using the DELETE FROM statement. The WHERE clause is used to specify the condition for deleting the record.

OUTPUT:

```
 ----- ---------------- ------- ------------------
| id | name        | age | department |
----- ---------------- ------- ------------------
| 1  | John Doe        | 32  | IT          |
| 2  | Jane Smith      | 28  | HR        |
| 3  | Mark Johnson | 35  | Sales     |
----- ---------------- ------- ------------------
```

2. Create Tables and Perform Joins in MySQL

CODE:

```
-- Create tables
CREATE TABLE employees (
  id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  department_id INT
);

CREATE TABLE departments (
  id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL
);

-- Insert data
INSERT INTO employees (id, name, department_id)
VALUES (1, 'John Doe', 1),
       (2, 'Jane Smith', 2),
       (3, 'Mark Johnson', 1);

INSERT INTO departments (id, name)
VALUES (1, 'IT'),
       (2, 'HR');

-- Perform JOIN operation
SELECT employees.name, departments.name AS department
FROM employees
JOIN departments ON employees.department_id = departments.id;
```

EXPLANATION:

The first step is creating two tables: "employees" and "departments". The "employees" table has columns for id, name, and department_id, while the "departments" table has columns for id and name. The primary key is defined for both tables using the PRIMARY KEY constraint.

Insert some sample data into both tables using the INSERT INTO statement.

To illustrate the JOIN operation, a query is performed that retrieves the names of employees along with their corresponding department names. The JOIN is performed by specifying the related columns in the ON clause: employees.department_id and departments.id.

OUTPUT:

```
 --------------------- ------------------
| name              | department |
--------------------- ------------------
| John Doe          | IT                 |
| Jane Smith        | HR                 |
| Mark Johnson | IT                 |
--------------------- ------------------
```

3. Create, Delete and Update commands in MongoDB

CODE:

```
// Connect to the MongoDB server
mongo

// Switch to the desired database
use mydatabase

// Create a collection (similar to a table in relational databases)
db.createCollection("employees")

// Insert documents (similar to rows in relational databases)
db.employees.insertMany([
  { name: "John Doe", age: 30, department: "IT" },
  { name: "Jane Smith", age: 28, department: "HR" },
  { name: "Mark Johnson", age: 35, department: "Sales" }
])

// Find all documents in the collection
db.employees.find()
```

```
// Update a document
db.employees.updateOne(
  { name: "John Doe" },
  { $set: { age: 32 } }
)

// Find the updated document
db.employees.findOne({ name: "John Doe" })

// Delete a document
db.employees.deleteOne({ name: "Mark Johnson" })

// Find all remaining documents
db.employees.find()
```

EXPLANATION:

Here, the MongoDB server is connected using the mongo command. Then, the desired database is switched to using the use command (mydatabase in this case).

A collection named "employees" is created using the createCollection method. Following that, multiple documents are inserted into the collection using the insertMany method.

To illustrate the update command, the updateOne method is used to update the age of an employee named "John Doe". The filter condition { name: "John Doe" } is specified, and the age field is updated to 32 using the $set operator.

To demonstrate the delete command, the deleteOne method is used to remove the document for the employee named "Mark Johnson". The filter condition { name: "Mark Johnson" } is specified.

The documents in the collection before and after the update and deletion are retrieved using the find method. The findOne method is used to retrieve the specific document for "John Doe" after the update.

OUTPUT:

Before Updation and Deletion

```
{ "_id" : ObjectId("615f981e9a12b789a4ac0d2e"), "name" : "John Doe", "age" : 30, "department" :
"IT" }

{ "_id" : ObjectId("615f981e9a12b789a4ac0d2f"), "name" : "Jane Smith", "age" : 28, "department" :
"HR" }

{ "_id" : ObjectId("615f981e9a12b789a4ac0d30"), "name" : "Mark Johnson", "age" : 35,
"department" : "Sales" }
```

After update:

```
{ "_id" : ObjectId("615f981e9a12b789a4ac0d2e"), "name" : "John Doe", "age" : 32, "department" :
"IT" }
```

After deletion:

```
{ "_id" : ObjectId("615f981e9a12b789a4ac0d2e"), "name" : "John Doe", "age" : 32, "department" :
"IT" }

{ "_id" : ObjectId("615f981e9a12b789a4ac0d2f"), "name" : "Jane Smith", "age" : 28, "department" :
"HR" }
```