# 1. a) Write a program to insert a record using Spring JDBC

Sure, here is an example program to insert a record using Spring JDBC:

```java
import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;

public class RecordDao {

    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;

    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public void insertRecord(Record record) {
        String sql = "INSERT INTO records (id, name, age) VALUES (?, ?, ?)";
        Object[] params = {record.getId(), record.getName(), record.getAge()};
        jdbcTemplate.update(sql, params);
    }
}
```

```

In this example, the RecordDao class uses Spring JDBC's JdbcTemplate to insert a new record into a database. The dataSource property is set through a setter method, which allows Spring to inject a DataSource object into the class. The insertRecord() method takes a Record object as input, and uses the JdbcTemplate's update() method to execute an INSERT SQL statement with the Record's properties as parameters.

## b) Write short note on DDL query with example

DDL stands for Data Definition Language, which is used to define and manipulate the structure of a database. DDL commands are used to create, alter, and delete database objects such as tables, views, indexes, and constraints.

Here is an example of a DDL query to create a new table:

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    age INT,
    address VARCHAR(200),
    salary DECIMAL(10, 2),
    hire_date DATE
);
```

This SQL query creates a table named "employees" with several columns, including an id column
```

with a primary key constraint, a name column with a maximum length of 50 characters and a NOT NULL constraint, an age column of type INT, an address column of type VARCHAR with a maximum length of 200 characters, a salary column of type DECIMAL with a maximum of 10 digits and 2 decimal places, and a hire_date column of type DATE.

DDL commands can also be used to alter the structure of an existing table. For example, to add a new column to the "employees" table, you could use the following ALTER TABLE statement:

```
ALTER TABLE employees ADD COLUMN email VARCHAR(100);
```

This query adds a new column named "email" to the "employees" table with a maximum length of 100 characters.

Overall, DDL queries are an essential part of managing and defining the structure of a database.

2. a) Write a program to delete a record using Spring JDBC

```
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;
public class RecordDao {

    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;

    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }
```

```
    public void deleteRecord(int id) {
        String sql = "DELETE FROM records WHERE id = ?";
        Object[] params = {id};
        jdbcTemplate.update(sql, params);
    }
}
```

## b) Explain about DML Query with example

DML stands for Data Manipulation Language, which is used to manipulate data within a database. DML commands are used to insert, update, and delete data in database tables.

Here are examples of some common DML queries:

1. INSERT: The INSERT command is used to add new rows to a database table. For example:

```
INSERT INTO employees (id, name, age, salary) VALUES (1, 'John', 30, 50000);
```

This query inserts a new row into the "employees" table with values for the id, name, age, and salary columns.

2. UPDATE: The UPDATE command is used to modify existing rows in a database table. For example:

```

UPDATE employees SET salary = 55000 WHERE name = 'John';

```

This query updates the "salary" column for the row where the "name" column is equal to 'John'. It sets the salary value to 55000.

3. DELETE: The DELETE command is used to remove rows from a database table. For example:

```

DELETE FROM employees WHERE id = 1;

```

This query deletes the row from the "employees" table where the "id" column is equal to 1.

4.SELECT: To display the data in the table

DML commands are essential for managing data within a database. They allow for the insertion, modification, and deletion of data, which is essential for creating and maintaining accurate and up-to-date data in a database.

## 3. a) Write a program to update a record using Spring JDBC

Sure, here's an example program to update a record using Spring JDBC:

```java

import javax.sql.DataSource;
```

```
import org.springframework.jdbc.core.JdbcTemplate;

public class RecordDao {

    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;

    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public void updateRecord(int id, String name, int age, int salary) {
        String sql = "UPDATE records SET name = ?, age = ?, salary = ? WHERE id = ?";
        Object[] params = {name, age, salary, id};
        jdbcTemplate.update(sql, params);
    }
}
```

In this example, the RecordDao class uses Spring JDBC's JdbcTemplate to update a record in a database. The dataSource property is set through a setter method, which allows Spring to inject a DataSource object into the class. The updateRecord() method takes four input values: the id of the record to update, and the new values for the name, age, and salary columns. The method uses the JdbcTemplate's update() method to execute an UPDATE SQL statement with the new values as parameters. The statement updates the row in the "records" table with the matching id value.

## b) Describe DCL and TCL in detail with example

DCL (Data Control Language) and TCL (Transaction Control Language) are two important subsets of SQL that are used to control access to data and manage transactions.

DCL is used to control user access to the database. The following are some of the commonly used DCL commands:

1. GRANT: This command is used to grant specific privileges to a user or group of users on a database object. For example:

```
GRANT SELECT, UPDATE ON employees TO user1;
```

This command grants user1 the SELECT and UPDATE privileges on the "employees" table.

2. REVOKE: This command is used to revoke previously granted privileges from a user or group of users on a database object. For example:

```
REVOKE SELECT ON employees FROM user1;
```

This command revokes user1's SELECT privilege on the "employees" table.

TCL is used to manage transactions in the database. The following are some of the commonly used TCL commands:

1. COMMIT: This command is used to permanently save changes made in a transaction to the database. For example:

```
BEGIN TRANSACTION;

UPDATE employees SET salary = salary + 1000 WHERE department = 'IT';

COMMIT;
```

This code updates the salaries of all employees in the "IT" department and then commits the changes to the database.

2. ROLLBACK: This command is used to undo changes made in a transaction and restore the database to its state before the transaction began. For example:

```
BEGIN TRANSACTION;

UPDATE employees SET salary = salary + 1000 WHERE department = 'IT';

ROLLBACK;
```

This code updates the salaries of all employees in the "IT" department but then rolls back the changes, effectively undoing the update.

In summary, DCL commands are used to grant or revoke user privileges on database objects, while TCL commands are used to manage transactions in the database.

## 4. Write a program to read records using Spring JDBC

```
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
```

```
import org.springframework.jdbc.datasource.DriverManagerDataSource;


public class ReadRecordsExample {
    public static void main(String[] args) {
        // create a data source
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/test");
        dataSource.setUsername("root");
        dataSource.setPassword("password");

        // create a JDBC template
        JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

        // query for all records in the table
        List<Employee> employees = jdbcTemplate.query("SELECT * FROM employees",
            (rs, rowNum) -> new Employee(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("department"),
                rs.getInt("salary")
            ));

        // print the results
        for (Employee employee : employees) {
```

```
            System.out.println(employee);
        }
    }
}

class Employee {
    private int id;
    private String name;
    private String department;
    private int salary;

    public Employee(int id, String name, String department, int salary) {
        this.id = id;
        this.name = name;
        this.department = department;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}

public int getSalary() {
    return salary;
}

public void setSalary(int salary) {
    this.salary = salary;
}
```

```
    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", department=" + department + ",
salary=" + salary + "]";
    }
}
```

## 5. a) Describe Different types of cloud computing deployment models

Cloud computing deployment models refer to the different ways in which cloud computing services are made available to users. The four main types of cloud computing deployment models are:

1. Public cloud: A public cloud is a cloud computing model that offers services to the general public over the internet. Public clouds are owned and operated by third-party cloud service providers, and users can access computing resources on a pay-per-use basis.

2. Private cloud: A private cloud is a cloud computing model that is dedicated to a single organization. Private clouds can be hosted on-premises or by a third-party provider, and offer greater control and customization options than public clouds.

3. Hybrid cloud: A hybrid cloud is a cloud computing model that combines elements of both public and private clouds. In a hybrid cloud, an organization can use public cloud services for some tasks and private cloud services for others. Hybrid clouds can be used to balance performance, security, and cost considerations.

4. Community cloud: A community cloud is a cloud computing model that is shared by multiple organizations that have common interests or requirements. Community clouds can be owned and operated by one of the organizations, or by a third-party provider. Community clouds can provide cost savings, as resources are shared among multiple organizations.

## b) Explain Real World Applications of Cloud Computing.

Cloud computing has many real-world applications in various industries. Some of the most common applications of cloud computing include:

1. Data storage and backup: Cloud computing provides businesses with a cost-effective and reliable way to store and backup their data. This is particularly useful for small and medium-sized businesses that may not have the resources to maintain their own data centers.

2. Software development and testing: Cloud computing provides developers with a scalable and flexible platform for software development and testing. Developers can quickly spin up virtual environments and test their applications in real-world scenarios without having to invest in expensive hardware.

3. Disaster recovery: Cloud computing provides businesses with a reliable and scalable disaster recovery solution. By replicating their data and applications to the cloud, businesses can quickly recover from any disaster and minimize downtime.

4. E-commerce: Cloud computing provides e-commerce businesses with a scalable and reliable platform for their online stores. Cloud-based e-commerce solutions can handle large amounts of traffic and provide a seamless shopping experience for customers.

5. Healthcare: Cloud computing is transforming the healthcare industry by providing healthcare providers with access to patient data from anywhere at any time. Cloud-based healthcare solutions can improve patient outcomes, reduce costs, and increase efficiency.

6. Education: Cloud computing is transforming education by providing students and teachers with access to online learning resources from anywhere at any time. Cloud-based educational solutions can improve student engagement, collaboration, and learning outcomes.

7. Entertainment: Cloud computing is transforming the entertainment industry by providing consumers with access to on-demand streaming services. Cloud-based entertainment solutions can provide a seamless and personalized entertainment experience for consumers.

## 6. Write a program to perform any DML operation using Spring JDBC

Here is an example program to perform an insert operation using Spring JDBC:

```java
import org.springframework.jdbc.core.JdbcTemplate;

public class EmployeeDAO {

    private JdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public int insertEmployee(Employee employee) {
        String sql = "INSERT INTO employees (id, name, salary) VALUES (?, ?, ?)";
        return jdbcTemplate.update(sql, employee.getId(), employee.getName(), employee.getSalary());
    }
```

}

```

In this example, we have created an EmployeeDAO class that has a JdbcTemplate object injected using Spring's dependency injection mechanism. The insertEmployee method takes an Employee object as input and inserts it into the database using the jdbcTemplate object's update method.

Here's an example program to perform a select operation using Spring JDBC:

```java
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;

public class EmployeeDAO {

    private JdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public List<Employee> getAllEmployees() {
        String sql = "SELECT * FROM employees";
        return jdbcTemplate.query(sql, new EmployeeRowMapper());
    }
```
```

}

```

In this example, we have created an EmployeeDAO class that has a JdbcTemplate object injected using Spring's dependency injection mechanism. The getAllEmployees method retrieves all employees from the database using the jdbcTemplate object's query method and returns them as a list of Employee objects. The query method requires a RowMapper object, which is implemented in the EmployeeRowMapper class.


ee saala cup namde.!!!!
```