Querying with HQL:

Explain Hibernate Query Language (HQL) and its role in Hibernate. Describe how to write and execute HQL queries to retrieve data from the database. Include examples of basic queries (SELECT, UPDATE, DELETE) and parameterized queries (setParameter()).

Hibernate Query Language (HQL) is a powerful query language similar to SQL (Structured Query Language), but it operates on Hibernate's object model rather than directly on database tables. HQL allows developers to write database-independent queries, which are then translated by Hibernate into native SQL queries suitable for the underlying database.

**Role of HQL in Hibernate:**

1. **Object-Oriented Queries**: HQL treats persistent objects as entities and their properties as object attributes rather than dealing with database tables and columns directly.
2. **Database Independence**: HQL queries are independent of the underlying database and are translated into appropriate SQL dialects by Hibernate, which makes Hibernate applications highly portable across different databases.
3. **Powerful and Flexible**: HQL supports various features such as joins, aggregates, and subqueries, making it suitable for complex querying requirements.

**Writing and Executing HQL Queries:**

To write and execute HQL queries in Hibernate, follow these steps:

1. **Create an HQL Query Object**: Use the createQuery method on the Hibernate Session to create an HQL query object.
2. **Write HQL Query**: Write the query string using HQL syntax. This is similar to writing SQL queries but uses entity names and their properties.
3. **Execute the Query**: Execute the query using methods like list() for SELECT queries or executeUpdate() for UPDATE and DELETE queries.

**Examples of HQL Queries:**

**1. SELECT Query:**

String hql = "FROM Employee";

Query query = session.createQuery(hql);

List<Employee> employees = query.list();

**2. UPDATE Query:**

String hqlUpdate = "UPDATE Employee e SET e.salary = :newSalary WHERE e.department = :dept";

Query query = session.createQuery(hqlUpdate);

query.setParameter("newSalary", 5000.00);

query.setParameter("dept", "IT");

int rowCount = query.executeUpdate();

**3. DELETE Query:**

String hqlDelete = "DELETE FROM Employee e WHERE e.department = :dept";

Query query = session.createQuery(hqlDelete);

query.setParameter("dept", "HR");

int rowCount = query.executeUpdate();

4. Parameterized Query:

String hql = "FROM Employee e WHERE e.salary > :minSalary";

Query query = session.createQuery(hql);

query.setParameter("minSalary", 3000.00);

List<Employee> employees = query.list();

**Using setParameter() for Parameters:**

The setParameter() method is used to set named parameters in HQL queries. It binds values to named parameters in the query string, ensuring type safety and protecting against SQL injection attacks.

query.setParameter("parameterName", value);

**Executing the Query:**

* **For SELECT Queries**: Use list() to retrieve a list of results.
* **For UPDATE and DELETE Queries**: Use executeUpdate() to perform the update or delete operation. It returns the number of affected rows.