SKILL–3: Working with HQL (Sorting, Pagination & Aggregates)

Project Name: hibernate-hql-skill3

Folder Structure

```
C:.
├───src
│   └───main
│       ├───java
│       │   └───com
│       │       └───example
│       │           ├───entity
│       │           ├───main
│       │           └───util
│       └───resources
└───target
    ├───classes
    │   └───com
    │       └───example
    │           ├───entity
    │           ├───main
    │           └───util
    ├───generated-sources
    │   └───annotations
    └───maven-status
        └───maven-compiler-plugin
            └───compile
                └───default-compile
```

Product.java

```
package com.example.entity;
```

```
import javax.persistence.*;
@Entity
@Table(name = "product")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int productId;
    private String name;
    private String description;
    private double price;
    private int quantity;
    public Product() {}
    public Product(String name, String description, double price, int quantity) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.quantity = quantity;
    }
    // Getters & Setters
    public int getProductId() { return productId; }
    public void setProductId(int productId) { this.productId = productId; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }
    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }
    @Override
    public String toString() {
        return productId + " | " + name + " | " + description + " | " + price + " | " + quantity;
```

```
    }
}
```

HQLDEMO.java

```
package com.example.main;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;
import com.example.entity.Product;
import com.example.util.HibernateUtil;
public class HQLDemo {
	public static void main(String[] args) {
		Transaction tx = null;
		// Use try-with-resources for session
		try (Session session = HibernateUtil.getSessionFactory().openSession()) {
			tx = session.beginTransaction();
			// --- STEP 6.1: Insert Products ---
			session.save(new Product("Laptop", "Electronics", 55000, 10));
			session.save(new Product("Mouse", "Electronics", 500, 50));
			session.save(new Product("Keyboard", "Electronics", 1200, 30));
			session.save(new Product("Chair", "Furniture", 3500, 15));
			session.save(new Product("Table", "Furniture", 8000, 5));
			session.save(new Product("Pen", "Stationery", 20, 100));
			tx.commit();
			// --- STEP 6.2: Sorting by Price ASC ---
			System.out.println("\n--- Price Ascending ---");
			session.createQuery("FROM Product p ORDER BY p.price ASC",
Product.class)
					.list()
					.forEach(System.out::println);
			// --- STEP 6.3: Sorting by Quantity DESC ---
```

```
                System.out.println("\n--- Quantity Descending ---");
                session.createQuery("FROM Product p ORDER BY p.quantity DESC",
Product.class)
                                .list()
                                .forEach(System.out::println);
                // --- STEP 6.4 & 6.5: Pagination ---
                Query<Product> query = session.createQuery("FROM Product",
Product.class);
                System.out.println("\n--- First 3 Products ---");
                query.setFirstResult(0);
                query.setMaxResults(3);
                query.list().forEach(System.out::println);
                System.out.println("\n--- Next 3 Products ---");
                query.setFirstResult(3);
                query.setMaxResults(3);
                query.list().forEach(System.out::println);
                // --- STEP 6.6: Count Products ---
                Long count = session.createQuery("SELECT COUNT(p) FROM Product p",
Long.class)
                                .uniqueResult();
                System.out.println("\nTotal Products: " + count);
                // --- STEP 6.7: Count Quantity > 0 ---
                Long countAvailable = session
                                .createQuery("SELECT COUNT(p) FROM Product p WHERE
p.quantity > 0", Long.class)
                                .uniqueResult();
                System.out.println("Available Products: " + countAvailable);
                // --- STEP 6.8: Group By Description ---
                System.out.println("\n--- Group By Description ---");
                session.createQuery("SELECT p.description, COUNT(p) FROM Product p
GROUP BY p.description")
                                .list()
                                .forEach(obj -> {
                                        Object[] arr = (Object[]) obj;
```

```
                                System.out.println(arr[0] + " -> " + arr[1]);
                        });
                // --- STEP 6.9: Min & Max Price ---
                Object[] minMax = (Object[]) session
                        .createQuery("SELECT MIN(p.price), MAX(p.price) FROM Product
p")
                        .uniqueResult();
                System.out.println("\nMin Price: " + minMax[0]);
                System.out.println("Max Price: " + minMax[1]);
                // --- STEP 6.10: Filter by Price Range ---
                System.out.println("\n--- Price Between 1000 and 10000 ---");
                session.createQuery("FROM Product p WHERE p.price BETWEEN :min
AND :max", Product.class)
                        .setParameter("min", 1000.0)
                        .setParameter("max", 10000.0)
                        .list()
                        .forEach(System.out::println);

                // --- STEP 6.11: LIKE Example ---
                System.out.println("\n--- Names Starting with L ---");
                session.createQuery("FROM Product p WHERE p.name LIKE 'L%'",
Product.class)
                        .list()
                        .forEach(System.out::println);

        } catch (Exception e) {
                if (tx != null)
                        tx.rollback(); // rollback if exception occurs
                e.printStackTrace();
        } finally {
                HibernateUtil.shutdown(); // close SessionFactory properly
        }
    }
```

```
}
```

HibernateUtil.java

```
package com.example.util;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import com.example.entity.Product;
public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    static {
        try {
            Configuration configuration = new Configuration();
            configuration.configure("hibernate.cfg.xml"); // loads from resources
            configuration.addAnnotatedClass(Product.class); // register your entity explicitly

            sessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
    public static void shutdown() {
        sessionFactory.close();
    }
}
```

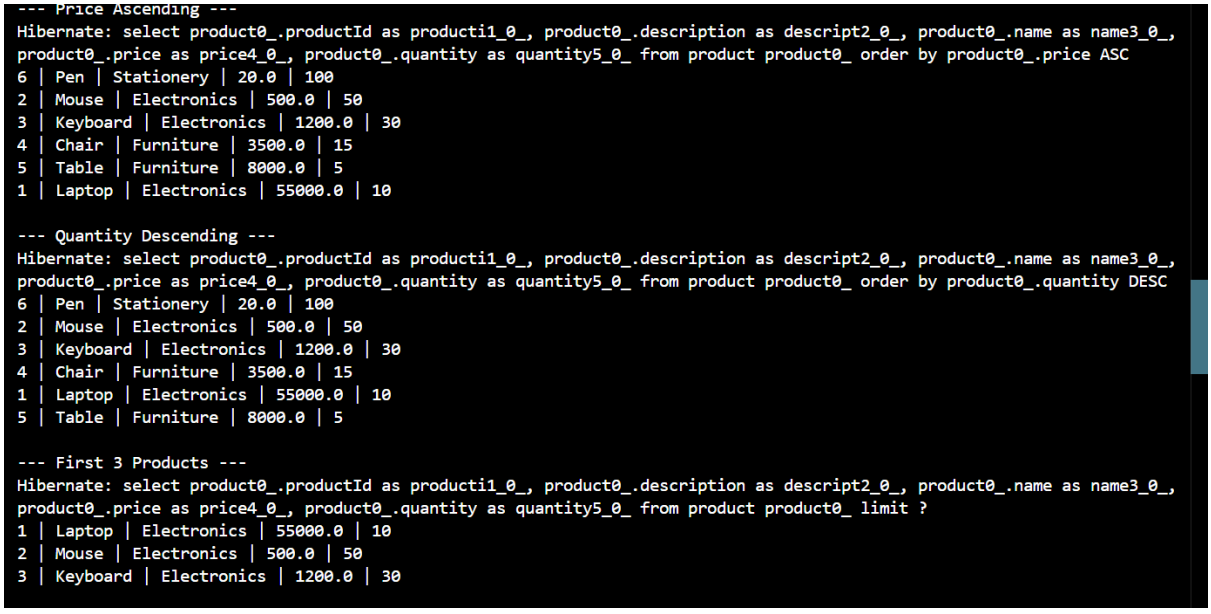
POM.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>hibernate-hql-skill3</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
</properties>
<dependencies>
    <!-- Hibernate Core -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.6.15.Final</version>
    </dependency>
    <!-- MySQL Connector (stable, downloadable version) -->
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.1.0</version>
    </dependency>
    <!-- JPA -->
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
</dependencies>
<build>
    <plugins>
```

```
        <!-- Plugin to run your main class -->
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>3.1.0</version>
            <configuration>
                <mainClass>com.example.main.HQLDemo</mainClass>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

OUTPUT



```
--- Next 3 Products ---
Hibernate: select product0_.productId as producti1_0_, product0_.description as descript2_0_, product0_.name as name3_0_,
product0_.price as price4_0_, product0_.quantity as quantity5_0_ from product product0_ limit ?, ?
4 | Chair | Furniture | 3500.0 | 15
5 | Table | Furniture | 8000.0 | 5
6 | Pen | Stationery | 20.0 | 100
Hibernate: select count(product0_.productId) as col_0_0_ from product product0_

Total Products: 6
Hibernate: select count(product0_.productId) as col_0_0_ from product product0_ where product0_.quantity>0
Available Products: 6

--- Group By Description ---
Hibernate: select product0_.description as col_0_0_, count(product0_.productId) as col_1_0_ from product product0_ group b
y product0_.description
Electronics -> 3
Furniture -> 2
Stationery -> 1
Hibernate: select min(product0_.price) as col_0_0_, max(product0_.price) as col_1_0_ from product product0_

Min Price: 20.0
Max Price: 55000.0

--- Price Between 1000 and 10000 ---
Hibernate: select product0_.productId as producti1_0_, product0_.description as descript2_0_, product0_.name as name3_0_,
product0_.price as price4_0_, product0_.quantity as quantity5_0_ from product product0_ where product0_.price between ? an
d ?
3 | Keyboard | Electronics | 1200.0 | 30
```

```
--- Names Starting with L ---
Hibernate: select product0_.productId as producti1_0_, product0_.description as descript2_0_, product0_.name as name3_0_,
product0_.price as price4_0_, product0_.quantity as quantity5_0_ from product product0_ where product0_.name like 'L%'
1 | Laptop | Electronics | 55000.0 | 10
Jan 01, 2026 2:28:25 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/experiment?allowPublicKeyRetrieval=true&useSSL
=false&serverTimezone=UTC]
[WARNING] thread Thread[#53,mysql-cj-abandoned-connection-cleanup,5,com.example.main.HQLDemo] was interrupted but is still
 alive after waiting at least 15000msecs
[WARNING] thread Thread[#53,mysql-cj-abandoned-connection-cleanup,5,com.example.main.HQLDemo] will linger despite being as
ked to die via interruption
[WARNING] NOTE: 1 thread(s) did not finish despite being asked to via interruption. This is not a problem with exec:java,
it is a problem with the running code. Although not serious, it should be remedied.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.067 s
[INFO] Finished at: 2026-01-01T14:28:40+05:30
[INFO] ------------------------------------------------------------------------
```

```
Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_experiment |
+----------------------+
| employee             |
| product              |
| student              |
| student_auto         |
| student_auto_seq     |
| student_identity     |
| student_sequence     |
| student_sequence_gen |
+----------------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM product;
+-----------+-------------+----------+-------+----------+
| productId | description | name     | price | quantity |
+-----------+-------------+----------+-------+----------+
|         1 | Electronics | Laptop   | 55000 |       10 |
|         2 | Electronics | Mouse    |   500 |       50 |
|         3 | Electronics | Keyboard |  1200 |       30 |
|         4 | Furniture   | Chair    |  3500 |       15 |
|         5 | Furniture   | Table    |  8000 |        5 |
|         6 | Stationery  | Pen      |    20 |      100 |
+-----------+-------------+----------+-------+----------+
6 rows in set (0.00 sec)

mysql>
```