Hands-on 2

Write queries on stock table using Query Methods

Folder structure

```
ormlearn/
├── src/
│   ├── main/
│   │   ├── java/com/example/ormlearn/
│   │   │   ├── OrmLearnApplication.java
│   │   │   ├── model/Stock.java
│   │   │   ├── repository/StockRepository.java
│   │   │   └── test/StockQueryTest.java
│   └── resources/
│       └── application.properties
└── pom.xml
```

1. Entity: Stock.java

```
package com.example.ormlearn.model;

import jakarta.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDate;

@Entity
@Table(name = "stock")
public class Stock {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int stId;

    private String stCode;
    private LocalDate stDate;
    private BigDecimal stOpen;
    private BigDecimal stClose;
    private Long stVolume;

    // Getters and Setters
    public int getStId() { return stId; }
    public void setStId(int stId) { this.stId = stId; }

    public String getStCode() { return stCode; }
```

```
    public void setStCode(String stCode) { this.stCode = stCode; }

    public LocalDate getStDate() { return stDate; }
    public void setStDate(LocalDate stDate) { this.stDate = stDate; }

    public BigDecimal getStOpen() { return stOpen; }
    public void setStOpen(BigDecimal stOpen) { this.stOpen = stOpen; }

    public BigDecimal getStClose() { return stClose; }
    public void setStClose(BigDecimal stClose) { this.stClose = stClose; }

    public Long getStVolume() { return stVolume; }
    public void setStVolume(Long stVolume) { this.stVolume = stVolume; }

    @Override
    public String toString() {
        return stCode + " | " + stDate + " | " + stOpen + " | " + stClose + " | " + stVolume;
    }
}
```

2. Repository: StockRepository.java

```
package com.example.ormlearn.repository;

import com.example.ormlearn.model.Stock;
import org.springframework.data.jpa.repository.JpaRepository;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

    // Query 1: Facebook stocks in September 2019
    List<Stock> findByStCodeAndStDateBetween(String code, LocalDate startDate,
LocalDate endDate);

    // Query 2: Google stocks where price > 1250
    List<Stock> findByStCodeAndStCloseGreaterThan(String code, BigDecimal price);

    // Query 3: Top 3 volumes
    List<Stock> findTop3ByOrderByStVolumeDesc();

    // Query 4: Lowest 3 Netflix closing prices
```

```
    List<Stock> findTop3ByStCodeOrderByStCloseAsc(String code);
}
```

3. Main App: OrmLearnApplication.java

```
package com.example.ormlearn;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication {
    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }
}
```

4. Test Runner: StockQueryTest.java

```
package com.example.ormlearn.test;

import com.example.ormlearn.model.Stock;
import com.example.ormlearn.repository.StockRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import java.math.BigDecimal;
import java.time.LocalDate;
import java.util.List;

@Component
public class StockQueryTest implements CommandLineRunner {

    @Autowired
    private StockRepository stockRepository;

    @Override
    public void run(String... args) {

        System.out.println("\n--- Query 1: Facebook stocks in September 2019 ---");
        List<Stock> fbStocks = stockRepository.findByStCodeAndStDateBetween(
                "FB",
                LocalDate.of(2019, 9, 1),
```

```
            LocalDate.of(2019, 9, 30));
        fbStocks.forEach(System.out::println);

        System.out.println("\n--- Query 2: Google stocks with close price > 1250 ---");
        List<Stock> googleStocks =
stockRepository.findByStCodeAndStCloseGreaterThan("GOOGL", new
BigDecimal("1250"));
        googleStocks.forEach(System.out::println);

        System.out.println("\n--- Query 3: Top 3 stocks by volume ---");
        List<Stock> topVolume = stockRepository.findTop3ByOrderByStVolumeDesc();
        topVolume.forEach(System.out::println);

        System.out.println("\n--- Query 4: Lowest 3 Netflix closing prices ---");
        List<Stock> lowNetflix =
stockRepository.findTop3ByStCodeOrderByStCloseAsc("NFLX");
        lowNetflix.forEach(System.out::println);
    }
}
```

5. application.properties (MySQL Configuration)

```
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=your_password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

6. SQL Table Setup

```
CREATE DATABASE IF NOT EXISTS ormlearn;
USE ormlearn;

CREATE TABLE IF NOT EXISTS stock (
  st_id INT NOT NULL AUTO_INCREMENT,
  st_code VARCHAR(10),
  st_date DATE,
  st_open DECIMAL(10,2),
  st_close DECIMAL(10,2),
  st_volume BIGINT,
  PRIMARY KEY (st_id)
```

);

Sample Output:

--- Query 1: Facebook stocks in September 2019 ---
FB | 2019-09-03 | 184.00 | 182.39 | 9779400
FB | 2019-09-04 | 184.65 | 187.14 | 11308000
...
FB | 2019-09-27 | 180.49 | 177.10 | 14656200

--- Query 2: Google stocks with close price > 1250 ---
GOOGL | 2019-04-23 | 1256.64 | 1270.59 | 1593400
GOOGL | 2019-04-24 | 1270.59 | 1260.05 | 1169800
...

--- Query 3: Top 3 stocks by volume ---
FB | 2019-01-31 | 165.60 | 166.69 | 77233600
FB | 2018-10-31 | 155.00 | 151.79 | 60101300
FB | 2018-12-19 | 141.21 | 133.24 | 57404900

--- Query 4: Lowest 3 Netflix closing prices ---
NFLX | 2018-12-24 | 242.00 | 233.88 | 9547600
NFLX | 2018-12-21 | 263.83 | 246.39 | 21397600
NFLX | 2018-12-26 | 233.92 | 253.67 | 14402700