**HANDS ON 2:**

**Write queries on stock table using Query Methods** 

**Application.properties:**

spring.application.name=orm\_learn

spring.datasource.url=jdbc:h2:mem:orm\_learn;DB\_CLOSE\_DELAY=-1

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.hibernate.ddl-auto=create

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

server.port=8081

**Stock.java:**

**package** com.cognizant.orm\_learn.model;

**import** jakarta.persistence.\*;

**import** java.math.BigDecimal;

**import** java.time.LocalDate;

@Entity

@Table(name = "stock")

**public** **class** Stock {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "st\_id")

**private** **int** id;

@Column(name = "st\_code")

**private** String code;

@Column(name = "st\_date")

**private** LocalDate date;

@Column(name = "st\_open")

**private** BigDecimal open;

@Column(name = "st\_close")

**private** BigDecimal close;

@Column(name = "st\_volume")

**private** BigDecimal volume;

**public** **int** getId() { **return** id; }

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

**public** String getCode() { **return** code; }

**public** **void** setCode(String code) { **this**.code = code; }

**public** LocalDate getDate() { **return** date; }

**public** **void** setDate(LocalDate date) { **this**.date = date; }

**public** BigDecimal getOpen() { **return** open; }

**public** **void** setOpen(BigDecimal open) { **this**.open = open; }

**public** BigDecimal getClose() { **return** close; }

**public** **void** setClose(BigDecimal close) { **this**.close = close; }

**public** BigDecimal getVolume() { **return** volume; }

**public** **void** setVolume(BigDecimal volume) { **this**.volume = volume; }

}

**StockRepository.java:**

**package** com.cognizant.orm\_learn.repository;

**import** com.cognizant.orm\_learn.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> {

List<Stock> findByCodeAndDateBetween(String code, LocalDate startDate, LocalDate endDate);

List<Stock> findByCodeAndCloseGreaterThan(String code, BigDecimal close);

List<Stock> findTop3ByOrderByVolumeDesc();

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**OrmLearnApplication.java:**

**package** com.cognizant.orm\_learn;

**import** com.cognizant.orm\_learn.model.Stock;

**import** com.cognizant.orm\_learn.repository.StockRepository;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

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

**import** java.math.BigDecimal;

**import** java.time.LocalDate;

**import** java.util.List;

@SpringBootApplication

**public** **class** OrmLearnApplication **implements** CommandLineRunner {

@Autowired

**private** StockRepository stockRepository;

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.**class**, args);

}

@Override

**public** **void** run(String... args) **throws** Exception {

System.***out***.println("=== Facebook Stocks in Sep 2019 ===");

List<Stock> fbStocks = stockRepository.findByCodeAndDateBetween(

"FB", LocalDate.*of*(2019, 9, 1), LocalDate.*of*(2019, 9, 30));

fbStocks.forEach(**this**::printStock);

System.***out***.println("\n=== Google Stocks with Close > 1250 ===");

List<Stock> googStocks = stockRepository.findByCodeAndCloseGreaterThan(

"GOOGL", **new** BigDecimal("1250"));

googStocks.forEach(**this**::printStock);

System.***out***.println("\n=== Top 3 Highest Volume Transactions ===");

List<Stock> topVolume = stockRepository.findTop3ByOrderByVolumeDesc();

topVolume.forEach(**this**::printStock);

System.***out***.println("\n=== 3 Lowest Netflix Closing Stocks ===");

List<Stock> lowestNetflix = stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");

lowestNetflix.forEach(**this**::printStock);

}

**private** **void** printStock(Stock s) {

System.***out***.printf("%s | %s | Open: %.2f | Close: %.2f | Volume: %.0f%n",

s.getCode(), s.getDate(), s.getOpen(), s.getClose(), s.getVolume());

}

}

**OUTPUT:**







