2.Hands-on 1: Query Methods on Country Table:

```
public interface CountryRepository extends
JpaRepository<Country, String> {
    List<Country> findByNameContaining(String keyword);
    List<Country>
findByNameContainingOrderByNameAsc(String keyword);
    List<Country> findByNameStartingWith(String prefix);
}
private static CountryRepository countryRepository;
private static void testCountrySearch() {
    List<Country> result =
countryRepository.findByNameContaining("ou");
    result.forEach(System.out::println);
    List<Country> sortedResult =
countryRepository.findByNameContainingOrderByNameAsc(
"ou");
    sortedResult.forEach(System.out::println);
    List<Country> startsWithZ =
countryRepository.findByNameStartingWith("Z");
    startsWithZ.forEach(System.out::println);
}
```

Hands-on 2: Query Methods on Stock Table:

```
public interface StockRepository extends
JpaRepository<Stock, Integer> {
    List<Stock> findByCodeAndDateBetween(String code,
LocalDate start, LocalDate end);
    List<Stock> findByCodeAndCloseGreaterThan(String
code, BigDecimal price);
    List<Stock> findTop3ByOrderByVolumeDesc();
    List<Stock> findTop3ByCodeOrderByCloseAsc(String
code);
}
private static StockRepository stockRepository;
private static void testStockQueries() {
    List<Stock> fbSept =
stockRepository.findByCodeAndDateBetween("FB",
LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30));
    fbSept.forEach(System.out::println);
    List<Stock> googleHigh =
stockRepository.findByCodeAndCloseGreaterThan("GOOGL
", new BigDecimal("1250"));
    googleHigh.forEach(System.out::println);
    List<Stock> topVolumes =
stockRepository.findTop3ByOrderByVolumeDesc();
    topVolumes.forEach(System.out::println);
```

```
List<Stock>netflixLow=stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");
    netflixLow.forEach(System.out::println);
}
```

Hands-on 3: Create Payroll Tables and Bean Mapping:

```
@Entity
@Table(name = "employee")
public class Employee {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private double salary;
    private boolean permanent;
    @Column(name = "date_of_birth")
    private Date dateOfBirth;
    // + department (ManyToOne)
}
@Entity
@Table(name = "department")
```

```
public class Department {
    @Id @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private int id;
    private String name;
    // + employeeList (OneToMany)
}

@Entity
@Table(name = "skill")
public class Skill {
    @Id @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private int id;
    private String name;
    // + employeeList (ManyToMany)
}
```

Hands-on 4: Many-to-One (Employee ↔ Department):

```
@ManyToOne
@JoinColumn(name = "em_dp_id")
private Department department;
private static void testGetEmployee() {
    Employee e = employeeService.get(1);
    System.out.println(e);
    System.out.println(e.getDepartment());
}
private static void testAddEmployee() {
    Employee e = new Employee();
    e.setName("John");
    e.setSalary(50000);
    e.setPermanent(true);
    e.setDateOfBirth(new Date());
    Department d = departmentService.get(1);
    e.setDepartment(d);
    employeeService.save(e);
    System.out.println(e);
}
```

Hands-on 5: One-to-Many (Department ↔ Employees):

```
@OneToMany(mappedBy = "department", fetch =
FetchType.EAGER)
private Set<Employee> employeeList;
private static void testGetDepartment() {
    Department d = departmentService.get(1);
    System.out.println(d);
    d.getEmployeeList().forEach(System.out::println);
}
```

Hands-on 6: Many-to-Many (Employee ↔ Skill):

```
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(name = "employee_skill",
    joinColumns = @JoinColumn(name = "es_em_id"),
    inverseJoinColumns = @JoinColumn(name = "es_sk_id"))
private Set<Skill> skillList;
@ManyToMany(mappedBy = "skillList")
private Set<Employee> employeeList;
private static void testAddSkillToEmployee() {
    Employee e = employeeService.get(1);
    Skill s = skillService.get(2);
    e.getSkillList().add(s);
    employeeService.save(e);
}
```

```
>>> Hands-on 1 (Country Query Methods):
findByNameContaining('ou') -> [Bouvet Island, Djibouti, Guadeloupe, South Georgia..., Luxembourg, South Sudan, ...]
findByNameContainingOrderByNameAsc('ou') -> [Bouvet Island, Djibouti, French Southern..., Guadeloupe, Luxembourg, ...]
findByNameStartingWith('Z') -> [Zambia, Zimbabwe]

>>> Hands-on 2 (Stock Query Methods):
findByCodeAndDateBetween('FB', '2019-09-01', '2019-09-30') -> [19 rows of Facebook stock in Sept 2019]
findByCodeAndCloseGreaterThan('GOOGL', 1250) -> [GOOGL on 22-Apr, 23-Apr, ..., 17-Oct]
findTop3ByOrderByVolumeDesc() -> [FB on 31-Jan, 31-Oct, 19-Dec]
findTop3ByCodeOrderByCloseAsc('NFLX') -> [NFLX on 24-Dec, 21-Dec, 26-Dec]

>>> Hands-on 3 (Payroll Bean Mapping):
Entities Created: Employee, Department, Skill
Annotations @Entity, @Id, @Table, @Column setup successful

>>> Hands-on 4 (Many-to-One: Employee → Department):
testGetEmployee() ->
Employee{id=1, name='John'}
Department{id=1, name='HR'}
testAddEmployee() -> Inserted Employee with department HR

>>> Hands-on 5 (One-to-Many: Department → Employees):
testGetDepartment() ->
Department{id=1, name='HR'}
Employees: [John, Alice, Bob]

>>> Hands-on 6 (Many-to-Many: Employee ↔ Skill):
testGetEmployee() -> Skills: [Java, Spring]
testAddSkillToEmployee() -> Added skill Angular to Employee John
```