QUERY

1. Tìm khóa học (`Course`) có số tín chỉ nằm giữa `minCredits` và `maxCredits`

```
SELECT c
FROM Course c
WHERE c.credits BETWEEN :minCredits AND :maxCredits
```

2. Tìm TƯƠNG ĐỐI khóa học (`Course`) thuộc khoa có tên chứa `deptName`

```
SELECT c
FROM Course c
JOIN c.department d
WHERE d.name LIKE CONCAT("%", :deptName, "%")
```

3. Thống kê số khóa học theo khoa

```
SELECT d, COUNT(c)
FROM Department d
JOIN d.courses c
GROUP BY d
```

```
return query
        .getResultList()
        .stream()
        .collect(Collectors.toMap(
                obj -> (Department) obj[0],
                obj -> (Long) obj[1],
                (o1, o2) -> o1,
                LinkedHashMap::new
        ));
```

4. Tìm phòng ban (`Department`) có ngân sách lớn nhất

```
SELECT d
FROM Department d
WHERE d.budget = (
    SELECT MAX(d2.budget)
    FROM Department d2
)
```

5. Tìm Project mà có Staff

```
SELECT p
FROM Project p
WHERE SIZE(p.staffs) > 0
```

6. Project có số nhân viên > trung bình số nhân viên tất cả project

```
SELECT p
FROM Project p
WHERE SIZE(p.staffs) > (
    SELECT AVG(SIZE(p2.staffs))
    FROM Project p2
)
```

7. Staff tham gia project có ngân sách lớn nhất

```
SELECT s
FROM Staff s
JOIN s.projects p
WHERE p.budget >= (
    SELECT MAX(p2.budget)
    FROM Project p2
)
```

8. Department có số staff >= số staff của 1 phòng ban bất kỳ có ID

```
SELECT d
FROM Department d
WHERE SIZE(d.staffs) > (
    SELECT SIZE(d2.staffs)
    FROM Department d2
    WHERE d2.id = :deptId
)
```

9. Staff không tham gia bất kỳ project nào có ngân sách < xxx

```
SELECT p
FROM Project p
WHERE SIZE(p.staffs) > (
    SELECT AVG(SIZE(p2.staffs))
    FROM Project p2
)
```

10. Tìm Department mà tồn tại ít nhất 1 Course có credits > :minCredits

```
SELECT d
FROM Department d
WHERE EXISTS (
    SELECT c
    FROM Course c
    WHERE c.department = d
      AND c.credits > :minCredits
)
```

11. Tổng ngân sách của Project theo Staff, chỉ lấy những Staff có tổng budget ≥ :minTotalBudget

```
SELECT s, SUM(p.budget)
FROM Staff s
JOIN s.projects p
GROUP BY s
HAVING SUM(p.budget) >= :minTotalBudget
```

12. Lấy 3 ký tự đầu của tên Staff

```
SELECT s, SUBSTRING(s.name, 1, 3)
FROM Staff s

Với SUBSTRING(thuộc tính, vị trí, số lượng lấy)
```

13. Tìm Course bắt đầu trong tháng hiện tại

```
SELECT c
FROM Course c
WHERE YEAR(c.startDate) = YEAR(CURRENT_DATE)
  AND MONTH(c.startDate) = MONTH(CURRENT_DATE)
```

14. Lấy tháng-năm từ paymentDate dưới dạng số

```
SELECT MONTH(p.paymentDate), YEAR(p.paymentDate)
FROM Payment p
```

15. Tìm Project đã kết thúc trước ngày hiện tại

```
SELECT p
FROM Project p
WHERE p.endDate < CURRENT_DATE
```

Lombook

```
@AllArgsConstructor
@NoArgsConstructor
@Data
@EqualsAndHashCode(onlyExplicitlyIncluded = true,
callSuper = true)
@ToString(callSuper = true)
```

- Đối với class khác (mqh): `@ToString.Exclude`
- Đối với field ID: `@EqualsAndHashCode.Include`

JPA HIBERNATE

1. Có 2 loại kế thừa thường dùng là:

- `InheritanceType.SINGLE_TABLE`
- `InheritanceType.JOINED`

2. Cách làm kế thừa

Lớp cha: `@Inheritance(strategy = <loại kế thừa>)`
Lớp con: `extends` lớp cha

3. Đối với *SINGLE_TABLE*

- Đặt tên cột phân biệt (để ở lớp cha):

```
@DiscriminatorColumn(name = "tên")
```

- Đặt giá trị cho lớp con (nếu không đặt mặc địch lấy theo tên class) (để ở lớp con)

```
@DiscriminatorValue("tên")
```

4. Ngoài ra còn 1 loại kế thừa không tạo ra bảng chung như 2 cách trên đó là `@MappedSuperclass`

```
@MappedSuperclass
public abstract class Base {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    // getter, setter
}

@Entity
public class User extends Base {
    private String name;
    // getter, setter
}

@Entity
public class Product extends Base {
    private String title;
    // getter, setter
}
```

Các lớp extends từ lớp Base sẽ tạo thành bảng có thuộc tính giống base và thêm thuộc tính lớp đó. Base sẽ không tạo ra bảng.

6. Đánh dấu field không phải entity mà là collection các giá trị cơ bản

```
@ElementCollection
    @CollectionTable(name = "post_tags",
                     joinColumns = @JoinColumn(name =
"post_id"))
    @Column(name = "tag", nullable=false)
    private List<String> tags = new ArrayList<>();
```

7. đánh dấu một class không phải entity riêng mà sẽ được "nhúng" vào bảng của entity khác.

```
@Embeddable
public class Address {
    private String street;
    private String city;
}

@Entity
public class User {
    @Embedded
    private Address address;
}
```

8. Khóa phức

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderItemId implements Serializable {
    private Long orderId;
    private Long productId;
}

@Entity
@IdClass(OrderItemId.class)
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderItem {
    @Id
    private Long orderId;

    @Id
    private Long productId;

    private Integer quantity;
}
```

9. MapsID

```
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
}

@Entity
public class UserProfile {
    @Id
    private Long id;

    @OneToOne
    @MapsId
    @JoinColumn(name = "id")
    private User user;

    private String bio;
}
```