Exercise

1. Create a Person entity with instance variables Firstname, Lastname, salary, age and Id.

```java
/**
 * 1.Create a Person entity with instance variables Firstname, Lastname, salary, age and Id.
 */
@Entity
public class Person {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String firstName;

    private String lastName;

    private Integer age;

    private Integer salary;
```

2. Implement CrudRepository for it.

```java
 */
//  Exercise 2
public interface PersonRepository extends CrudRepository<Person, Integer> {

}
```

3. Perform all the methods inside CrudRepository for Person Class.

```
====================Single Record Added====================
Person{id=1, firstName='yatin', lastName='ajmani', age=24, salary=450000}
====================Multiple Records Added====================
Person{id=2, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=3, firstName='harsh', lastName='jain', age=25, salary=420000}
Person{id=4, firstName='fName', lastName='lName', age=24, salary=420000}
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
====================Find one with id 1====================
Person{id=1, firstName='yatin', lastName='ajmani', age=24, salary=450000}
====================Exists with id 1====================
true
====================Find All Records====================
Person{id=1, firstName='yatin', lastName='ajmani', age=24, salary=450000}
Person{id=2, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=3, firstName='harsh', lastName='jain', age=25, salary=420000}
Person{id=4, firstName='fName', lastName='lName', age=24, salary=420000}
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
====================Find All Records having ids 2,3,4====================
Person{id=2, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=3, firstName='harsh', lastName='jain', age=25, salary=420000}
Person{id=4, firstName='fName', lastName='lName', age=24, salary=420000}
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
No. of Persons : 5
====================Delete with id 4====================
Person{id=1, firstName='yatin', lastName='ajmani', age=24, salary=450000}
Person{id=2, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=3, firstName='harsh', lastName='jain', age=25, salary=420000}
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
====================Delete with id 1====================
Person{id=2, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=3, firstName='harsh', lastName='jain', age=25, salary=420000}
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
====================Delete using iterable with id 2,3====================
Person{id=5, firstName='Peter', lastName='parker', age=26, salary=410000}
====================Delete All====================
```

4. For class Person find person declare methods in repository to find person by firstname, lastname and Id.

```java
//  Exercise 4
List<Person> findByFirstName(String firstName);

List<Person> findByLastName(String lastName);

Person findById(Integer id);
```

5. Use the methods declared above to fetch the person.

```
Person with Id 6 : Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Persons with FirstName yatin : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
Persons with lastName ajmani : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
```

6. Use @Query to fetch firstname of the Person whose age is 25.

```java
//  Exercise 6
@Query("select firstName from Person where age=25")
List<Person> findByAgeViaQuery();
```

```
Persons with age = 25 : [yatin, harsh]
```

7. Use @Query to fetch firstname and lastname of the Person whose age is 25.

```java
//  Exercise 7
@Query("select CONCAT(p.firstName, ' ', p.lastName) as fullname from Person p where age=25")
List<String> findFullNameWithAge25ViaQuery();
```

```
Persons full Name with age = 25 :
yatin ajmani
harsh jain
```

8. Get complete information of the Employee whose age is 25 using @Query.

```
//  Exercise 8
@Query("from Person where age=25")
List<Person> findPersonWithAgeViaQuery();
```

```
Persons with age = 25 :
Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}
Person{id=7, firstName='harsh', lastName='jain', age=25, salary=420000}
```

9. Count Person where name is "Peter" using @Query.

```
//  Exercise 9
@Query("select count(p) from Person p where firstName='Peter'")
Integer findPersonWithNamePeterViaQuery();
```

```
Persons with first Name = Peter : 1
```

10. Implement following methods for Person repository:
    1. count
    2. distinct
    3. or
    4. and
    5. between
    6. LessThan
    7. GreaterThan
    8. Like
    9. Not
    10. In
    11. IgnoreCase

```
//  Exercise 10
Integer countAllByAge(Integer age);

List<Person> getDistinctByFirstNameOrAge(String fName, Integer age);

List<Person> getAllByFirstNameAndAge(String fName, Integer age);

List<Person> getByAgeBetween(Integer after, Integer before);

List<Person> getBySalaryLessThan(Integer salary);

List<Person> getBySalaryGreaterThan(Integer salary);

List<Person> getByFirstNameLike(String firstName);

List<Person> getByFirstNameNot(String firstName);

List<Person> getByFirstNameIn(List<String> firstNames);

List<Person> getByFirstNameIgnoreCase(String firstName);
```

```
Persons with age = 25, Using count : 2
Persons with age = 26 or firstName = yatin, Using distinct : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}, Person{id=9,
 firstName='Peter', lastName='parker', age=26, salary=410000}]
Persons with age = 25 and firstName = yatin, Using distinct : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
Persons age between 25 and 28= yatin, Using distinct : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}, Person{id=7, firstName='harsh',
 lastName='jain', age=25, salary=420000}, Person{id=9, firstName='Peter', lastName='parker', age=26, salary=410000}]
Persons salary less than 420000 : [Person{id=9, firstName='Peter', lastName='parker', age=26, salary=410000}]
Persons salary greater than 420000 : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
Persons with name like yatin : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
Persons with name not like yatin : [Person{id=7, firstName='harsh', lastName='jain', age=25, salary=420000}, Person{id=8, firstName='fName', lastName='lName',
 age=24, salary=420000}, Person{id=9, firstName='Peter', lastName='parker', age=26, salary=410000}]
Persons with name not like yatin : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}, Person{id=7, firstName='harsh', lastName='jain',
 age=25, salary=420000}]
Persons with name YATIN ignoring case : [Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}]
```

11. Get the persons greater than age 25 and sort them in descending order according to id by method query.

```
//  Exercise 11
List<Person> findByAgeGreaterThanOrderByIdDesc(Integer age);
```

```
Persons with age greater than 25 sorted by descending order by id : [Person{id=9, firstName='Peter', lastName='parker', age=26, salary=410000}]
```

12. Do the question above using the Sort class.

```
//  Exercise 12
List<Person> findByAgeGreaterThan(Integer age,Sort sort);
```

```
 Persons with age greater than 25 sorted by descending order by id using sort class : [Person{id=9, firstName='Peter', lastName='parker', age=26, salary=410000}]
```

13. Apply Pagination on Person entities.

```
//  Exercise 13
Page<Person> findAll(Pageable pageable);
```

```
Paginated Persons : Page 1 of 2 containing entity.Person instances
[Person{id=6, firstName='yatin', lastName='ajmani', age=25, salary=430000}, Person{id=7, firstName='harsh', lastName='jain', age=25, salary=420000}]
```