```
                              Date :27/01/2021
                              Spring Boot 9AM
                                Mr. RAGHU
                    -----------------------------------
                Spring Data JPA : Custom Query Coding

=> Repository interfaces has provided pre-defined methods todo
   DB Operations, like findAll(), save(), deleteById()..etc

=> We can define our own Query, using conept 'Custom Query'.
  This can be implemnted using

a) @Query [SELECT + non-SELECT(Update/Delete)]
b) findBy [SELECT]

----@Query---------------------------------------------
-> HQL/JPQL [Hibernate Query Language/JPA Query Language]
  Concept is used to define custom query which is
  Database independent.

  We writre : HQL/JPQL --> Dialect --> Converted to SQL.

-> SQL queries are database dependent. So, recomanded to use
   HQL/JPQL.

-> Even @Query supports PureSQL queries/Native SQL.

SQL: Table Name and ColumnNames.
HQL/JPQL: Class Name and Variable Name.

(SQL->HQL/JPQL)
Replace : TableName ---> className, ColumnName--> VariableName

--Examples------------
1.
SQL:
  select eid,ename from emptab where esal>?
HQL/JPQL:
  select empId,empName from in.nit.model.Employee where empSal>?0

2. SQL is case-insensitive.
SQL:
  select eid,ename from emptab where esal>?
  SELECT EID,ENAME FROM EMPTAB WHERE ESAL>?

HQL/JPQL - Partially Case-sensitive :
    Java words case-sensitive(class,variable,package)
    SQL words are case-insensitive (select, where, from..)

  SELECT empId, empName FROM in.nit.model.Employee WHERE empSal>?0

3. DO NOT WRITE * SYMBOL IN HQL/JPQL
   (HQL/JPQL is java so * indicates multiply, not all columns)

SQL:
   select * from emptab;
```

```
HQL/JPQL:
   select * from in.nit.model.Employee (invalid)

   FROM in.nit.model.Employee (VALID)
   SELECT e FROM in.nit.model.Employee e (VALID) -alias naming is
valid

4. Package name is optional while using Model className in HQL/JPQL

SQL:
  select eid,ename from emptab where esal>?
HQL/JPQL:
  SELECT empId, empName FROM in.nit.model.Employee WHERE empSal>?0
  SELECT empId, empName FROM Employee WHERE empSal>?0

5. Parameters are allowed in HQL/JPQL, but use ?0, ?1, ?2 inplace of
   Simple ? symbols.

6. Even named Parameters also supported. Syntax=>  :name
7. Even Non-select operations Update and DELETE supported.
----------------------------------------------------------------------
                  IntelliJ IDEA- Setup
#1 Download :
  https://www.jetbrains.com/idea/download/#section=windows
  Choose: Community Option

#2. Run Executable file for install (ideaIC-2020.3.1.exe)
  > Next > Next > Finish

#3. Open IntelliJ and close project if already exist.
    (File > Close Project)

#4. Create Spring Boot application using Spring Initializer
   (https://start.spring.io/)
   -> Fill details
   -> Choose Dependencies
   -> Generate as ZIP File
   -> Extract to a folder

#5. Enable Lombok in IntelliJ IDEA
    https://projectlombok.org/setup/intellij

#6. Restart IntelliJ IDEA and start coding

-> First create packages then create class/interface..

a. Model
package in.nareshit.raghu.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;

@Data
```

```
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Employee {
    @Id
    private Integer empId;
    private String empName;
    private Double empSal;
}
```

b. Repository Interface

```
package in.nareshit.raghu.repo;

import in.nareshit.raghu.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepo extends JpaRepository<Employee,Integer>
{
}
```

c. Runner class

```
package in.nareshit.raghu.runner;

import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.EmployeeRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class EmployeeInsertRunner implements CommandLineRunner {
    @Autowired
    private EmployeeRepo repo;
    @Override
    public void run(String... args) throws Exception {
        repo.save(new Employee(10,"A",2.2));
        repo.save(new Employee(11,"B",3.2));
        repo.save(new Employee(12,"C",4.2));
    }
}
```

4. Run main class (shift+F10)
 Open mail class > Run Menu > Run Option

*) Note : Setup JDK/SDK
> Right click on Project > open Module Settings
> Click on Project > select SDK > Choose JDK Version
> Apply > OK

--For Custom Query follow below steps---
S#1. Add one abstract method in Repository Interface

S#2. Provide @Query("HQL/JPQL") over abstract method

S#3. Call this method in Runner class for testing

```
--Repository Interface--
package in.nareshit.raghu.repo;

import in.nareshit.raghu.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.List;

public interface EmployeeRepo extends JpaRepository<Employee,Integer>
{
    @Query("SELECT e FROM in.nareshit.raghu.model.Employee e")
    List<Employee> getAllEmps();

    @Query("SELECT e.empName FROM in.nareshit.raghu.model.Employee e")
    List<String>  getAllEmpNames();

    @Query("SELECT e.empId,e.empName FROM
in.nareshit.raghu.model.Employee e")
    List<Object[]> getAllEmpIdAndNames();
}


--Runner class--
package in.nareshit.raghu.runner;

import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.EmployeeRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import java.util.Iterator;
import java.util.List;

@Component
public class EmployeeInsertRunner implements CommandLineRunner {
    @Autowired
    private EmployeeRepo repo;
    @Override
    public void run(String... args) throws Exception {
        /*repo.save(new Employee(10,"A",2.2));
        repo.save(new Employee(11,"B",3.2));
        repo.save(new Employee(12,"C",4.2));
        */
        //List<Employee> list = repo.getAllEmps();
        //list.forEach(System.out::println);
        //List<String> list = repo.getAllEmpNames();
        //list.forEach(System.out::println);
        List<Object[]> list = repo.getAllEmpIdAndNames();
        //java #8 Stream
        /*list.stream()
                .map(ob->ob[0]+"-"+ob[1])
                .forEach(System.out::println);*/
        Iterator<Object[]> itr = list.iterator();
        while (itr.hasNext()) {
            Object[] ob=itr.next();
            System.out.println(ob[0]+"-"+ob[1]);
        }
        /*for(Object[] ob:list) {
            System.out.println(ob[0]+"-"+ob[1]);
        }*/
    }
}
```