Note:: Plz watch navinreddy sir youtube video of ====> SpringMongoDB
link::
https://www.youtube.com/watch?v=kYiLzIiHVY8

LOB
 a.BLOB(Binary Large Object ===> audio file,video file, image file,pdf and etc)
 b.CLOb(Character Large Object ===> Text file,Rich Text file,CSV file and etc)

To work with LOB's we need to map entity class to DBTable using annotations
      BLOB => Take a byte[] property in Entity class and mark the annotation called @Lob
    CLOB => Take a char[] property in Entity class and mark the annotation called @Lob
    The boolean property data(true/false) of Entity class object will be stored in db table as bit column which holds the
    value 0 or 1,

MarriageSeeker.java
===================

```
package in.ineuron.bo;

import java.io.Serializable;
import java.time.LocalDateTime;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@RequiredArgsConstructor
public class MarriageSeeker implements Serializable {

      private static final long serialVersionUID = 1L;

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

      @NonNull
      private String name;

      @NonNull
      private String address;

      @Lob
      @NonNull
      private byte[] photo;
      @NonNull
      private LocalDateTime dob;

      @Lob
      @NonNull
      private char[] bioData;

      @NonNull
      private Boolean indian;

}
```

Code to read the data into byte[] and char[]
==

```
InputStream inputStream = new FileInputStream(photoPath);
      byte[] photoData = new byte[inputStream.available()];
      inputStream.read(photoData);

File file = new File(bioData);
      Reader reader = new FileReader(file);
      char[] bioDataContent = new char[(int) file.length()];
      reader.read(bioDataContent);
```

Code to write the data from byte[] and char[]
===

```
OutputStream os = new FileOutputStream("retrieve_image.jpg");
      os.write(seeker.getPhoto());
      os.flush();

Writer writer = new FileWriter("retrive_biodata.txt");
      writer.write(seeker.getBioData());
      writer.flush();
```

Working with @Query methods
===========================
On the top of custom methods declared in our repository interface, we need to add
@Query annotation either having JPQL/HQL/
Native SQL Query.
Methods can have flexible signature and no need of following any naming
conventions.

Syntax
======

```
            @Query("HQL/JPQL/nativesqlqueries")
            <return_type><method_name><params....)
```

Note:

```
SQL> SELECT * FROM CORONA_VACCINE;
HQL> FROM  in.ineuron.bo.CoronaVaccine

SQL> UPDATE CORONA_VACCINE SET price = price + ? where company = ?
HQL> UPDATE in.ineuron.bo.CoronaVaccine SET price = price+:addOnPrice WHERE company
=:manufacture

SQL> SELECT REGNO,COMPANY,NAME FROM CORONA_VACCINE WHERE COMPANY IN (?,?)
```

```
HQL> SELECT regNo,company,name FROM in.ineuron.bo.CoronaVaccine WHERE company
IN(:comp1,:comp2)
HQL> SELECT cv.regNo,cv.company,cv.name FROM in.ineuron.bo.CoronaVaccine as cv
WHERE cv.company IN(:comp1,:comp2)
```

Note:

```
  @Query("FROM CoronaVaccine WHERE COMPANY = ?1")
  @Query("From in.ineuron.bo.CoronaVaccine WHERE company=:vendor")
  public List<CoronaVaccine> searchVaccinesByCompany(String vendor);
```

The method parameter values will be bounded with named param values automatically
if there names are matching, otherwise we need
to use @Param explicitly.

eg::

```
 @Query("FROM CoronaVaccine WHERE company=:comp")
 public List<CoronaVaccine> searchVaccinesByCompany(@Param("comp")String vendor);
```

eg::

```
public interface ICoronaVaccineRepo extends JpaRepository<CoronaVaccine, Long> {

      @Query("FROM CoronaVaccine WHERE company=:comp")
    public List<CoronaVaccine> searchVaccinesByCompany(@Param("comp")String
vendor);

      @Query("FROM CoronaVaccine WHERE company IN(:comp1,:comp2)")
      public List<CoronaVaccine> searchVaccinesByComapnies(String comp1,String
comp2);

      @Query("SELECT name FROM CoronaVaccine WHERE price BETWEEN :min AND :max")
      public List<String> searchVaccinesByPriceRange(double min,double max);

      @Query("SELECT name,company,price from CoronaVaccine WHERE name
IN(:name1,:name2)")
      public List<Object[]> searchVaccinesByName(String name1,String name2);

}
```

```
1. service.fetchVaccinesByCompany("bharathbiotech").forEach(System.out::println);

2. service.fetchVaccinesByCompanies("bharathbiotech",
"serum").forEach(System.out::println);
   System.out.println();

3. List<Object[]> names = service.fetchVaccinesByName("covidshield", "covacin");
      for (Object[] objects : names) {
            for (Object obj : objects) {
                        System.out.print(obj + " ");
            }
            System.out.println();
      }
      System.out.println();

4. List<String> vaccineNames = service.fetchVaccinesByPriceRange(250.0, 750.0);
   System.out.println(vaccineNames);
```

refer:: DAO-SpringDataJPA-@QueryMethods_App