The Eurovision Song Contest is an international song writing competition organised annually by the European Broadcasting Union (EBU), featuring participants representing primarily European countries. Each participating country submits an original song to be performed on live television and radio, with competing countries then casting votes for the other countries' songs to determine a winner. It is recognised by the Guinness Book of World Records as the longest running annual TV "music" competition. The most recent competition was held in Turin in May 2022 with Ukraine emerging as winners.

From 2004 the contest expanded to become a multi-programme event, with a semi-final at the 49th contest allowing all interested countries to compete each year; a second semi-final was added to each edition from 2008.

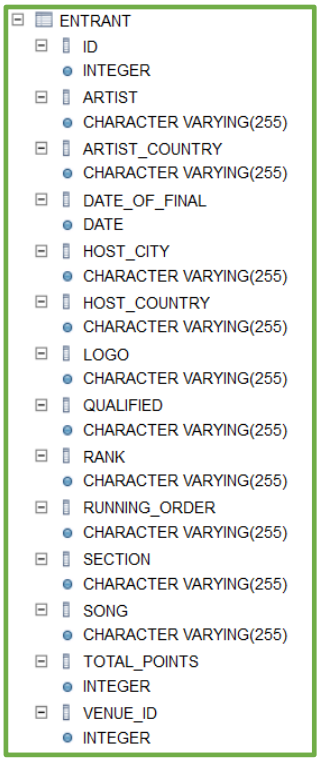
I have provided you with an SQL script for the '*eurovision*' database which contains a two tables, '*entrant*' and *'venue'*. The structure of the tables and their corresponding Entity classes are as follows:

```java
@Entity
public class Entrant implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private Integer venue_ID;
    private String logo;
    private String host_City;

    @Temporal(TemporalType.DATE)
    private Date date_Of_Final;

    private String host_Country;
    private String section;
    private String artist;
    private String song;
    private String artist_Country;
    private String running_Order;
    private Integer total_Points;
    private String rank;
    private String qualified;
}
```

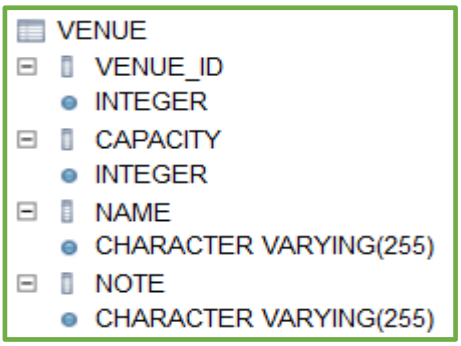


SELECT * FROM ENTRANT;

ID	ARTIST	ARTIST_COUNTRY	DATE_OF_FINAL	HOST_CITY	HOST_COUNTRY	LOGO	QUALIFIED	RANK	RUNNING_ORDER	SECTION	SONG	TOTAL_POINTS	VENUE_ID
1	Kalush Orchestra	Ukraine	2022-05-24	Turin	Italy	Turin2022.jpg	TRUE	1	6	first-semi-final	Stefania	337	1
2	S10	Netherlands	2022-05-24	Turin	Italy	Turin2022.jpg	TRUE	2	8	first-semi-final	De Diepte	179	1
3	Amanda Georgiadi Tenfjord	Greece	2022-05-24	Turin	Italy	Turin2022.jpg	TRUE	3	15	first-semi-final	Die Together	211	1

```java
@Entity
public class Venue {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Integer venue_ID;
    String name;
    Integer capacity;
    String note;
}
```



SELECT * FROM VENUE;

VENUE_ID	CAPACITY	NAME	NOTE
1	15657	PalaOlimpico	Opened on December 3 2005
2	16426	Rotterdam Ahoy	Inaugurated on 15 January 1971. The 1997 and 2016 MTV Europe Music Awards have been held here.
3	9628	Expo Tel Aviv	Opened in 1959

Note, that Spring will create the structure of the DB tables based on the entity classes I have provided you with. I am also distributing an SQL script *('data.sql')* in the */resources* folder within your project. The script will be executed every time you restart your project. I am also providing you with a series of images that you must use in your solution (*resources/static/assets/images)*

To Do

1. Facilitate the user to search the entrants table by specifying a date range and a country. The results (if any) should be displayed neatly and ordered by *total_Points*. A partial search on the country name should be permissible. For example, display all records in the *entrant* table between 2016-5-14 and 2008-5-24 for Ireland.

   (15 Marks)

2. You must provide **one** other means (criteria) of searching the database. The more creative and in-depth this feature is, the more marks you will accumulate. Again, all search results should be displayed neatly.

   (25 Marks)

3. For tasks 1 and 2, the user should be permitted to drill down on an individual search result. On the drilldown page, data from both the *entrant* and *venue* tables, along with the corresponding image should be displayed.

(15 Marks)

4. Develop functionality which will allow the user to select an entrant record (this can be as simple as passing the id for the record into the controller as a path parameter). The selected record should be presented in a prepopulated form, where the end user can edit the displayed records details. Ensure you have sufficient levels of bean validation in place in your entity class (you can decide on the validation criteria yourself).

(15 Marks)

5. Your application must be localised for two locales – Ireland\English and another locale of your choosing. The user must be permitted to switch locales at any time using a suitable **GUI component/widget**. All locale sensitive information must be localised to the selected locale. All errors and/exceptions must be handled within your application and any subsequent error pages must be localised as well. The information within the database does not have to be localised.

(30 Marks)

Stipulations:

1. You must use Spring Boot for this assignment.
2. You must use the H2 Database.
3. You must use the CrudRepository.
4. I am using Github classroom to manage this assignment and you are **required** to commit your work to the repository **before the end** of your double class each week. The reality is that you will more than likely commit much more regularly than that. A penalty of 3% will be applied for each commit that you miss. You are required to document the development effort in the provided README.
5. You must upload your final solution to Moodle. The version you upload to Moodle should be identical to the final commit on GitHub.
6. You are not permitted to use *DataTables* (or an equivalent) in this assignment.
7. You are not permitted to use JSP/JSTL as your view technology.

Note:

1. There is starter code, an SQL script, and a series of images available for this assignment on GitHub.
2. Consider using Springs Pagination for presenting search results.
3. This reference document for H2 might prove useful.
4. The deadline for this assignment is **6:00pm on Friday, March 3rd**.
5. You are required to demonstrate your work in class to me on the week of, **Monday, March 6th**. During the demonstration, you *may be* required to extend the functionality you have developed.