Sales Statistics

Description

Objective:

To work with One to Many Relationship between entities and implement as Bidirectional and usage of group function in query method

Concept Explanation:

1. When one instance of an entity is associated with multiple instances of another entity, then we call that relationship **one-to-many.**
2. That relationship will be **bi-directional** if both the entities are aware of each other and can navigate from one entity to another.
3. The **Group Function Query** Method in JPA involves using aggregate functions like `COUNT`, `MAX`, `MIN`, `SUM`, etc., along with grouping criteria to perform operations across groups of data

Concept Implementation:

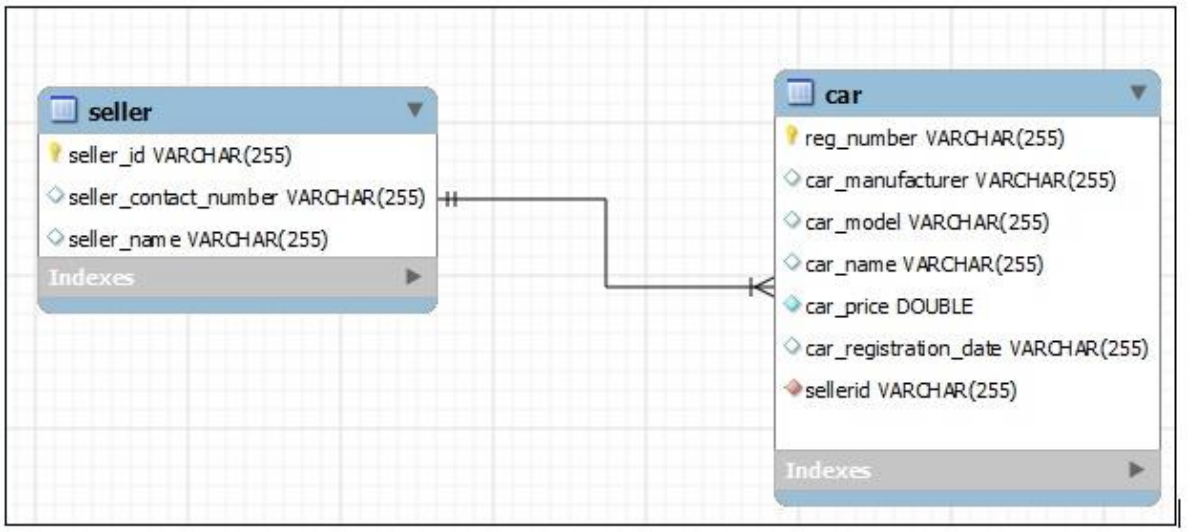
1. We establish a one-to-many relationship between the Seller and Car entities by providing appropriate annotation with the proper attribute pointing to the Seller above the carList attribute in the Seller entity.
2. The Car entity holds a reference to the Seller entity, establishing a many-to-one relationship, where each Car is associated with one Seller. Provide proper annotation that specifies the foreign key column in the Car entity, enabling bidirectional retrieval of Seller and Car details.
3. Use JPA Custom Query methods with the proper group functions to find the seller with the highest sales counts in the appropriate repository interface

E Sales Statistics

Click here to download the code skeleton

Warner Used Car Showroom wants to manipulate and store the details of the cars sold through them by the different sellers. Develop a Spring Data JPA application to perform the task.

Database Design:



Note:

- Seller and Car has one to many relationships.
- Establish the relationship between the Seller and the Car as Bi-directional.

Functionalities:

1. Add Seller

Warner Used Car Showroom will store the seller information by providing details like the seller id, seller name and the seller contact number. This information will be stored in the Seller table.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SellerDAO	addSeller	This method stores the seller details in the	Seller seller	Void

		Seller table		

Model class

Component Name	Attributes	Methods
Seller	sellerId - String sellerName - String sellerContactNumber - String The car details - the attribute name should be "carList"	getter and setter methods
Car	regNumber - String carName - String carManufacturer - String carModel - String carRegistrationDate - String carPrice - double The seller information- the attribute name should be "seller".	getter and setter methods

2. Allocate a Car to the Seller

To the already existing seller**,** the Warner Used Car Showroom wants to add a car by providing the details like the registration number, car name, car manufacturer, car model, car price and the car registration date.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SellerDAO	buyCar	To the already existing seller add the car. This method should add the Car object in the "Car" table.	String sellerId, Car car	void

3. Find the seller with the highest sales count

This functionality should list the sellers who have the highest sales count. (Hint: One or more sellers might have the highest sales count)

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
SellerDAO	sellerWithMaximumSalesCount	This method should return the list of sellers who have the highest sales count.		List<Seller>

Design Constraints

1. All the classes and methods should have public access Specifiers.

2. All the attributes should have private access Specifiers.

3. Use Annotation for all the mapping.

4. seller_id and reg_number should be marked as a primary key.

5. Use Spring data JPA API for persisting the object into the database.

6. The tables below should get created by hibernate automatically with the proper parent and child relationship.

7. The table should be created with the correct table name and column name as specified in the database design diagram.

8. Do not alter the code skeleton and do not include any additional packages.

9. The column names and method name should be given as specified in the document.

10. Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.

11. In the pom.xml we are given with all the dependencies needed for developing the application.