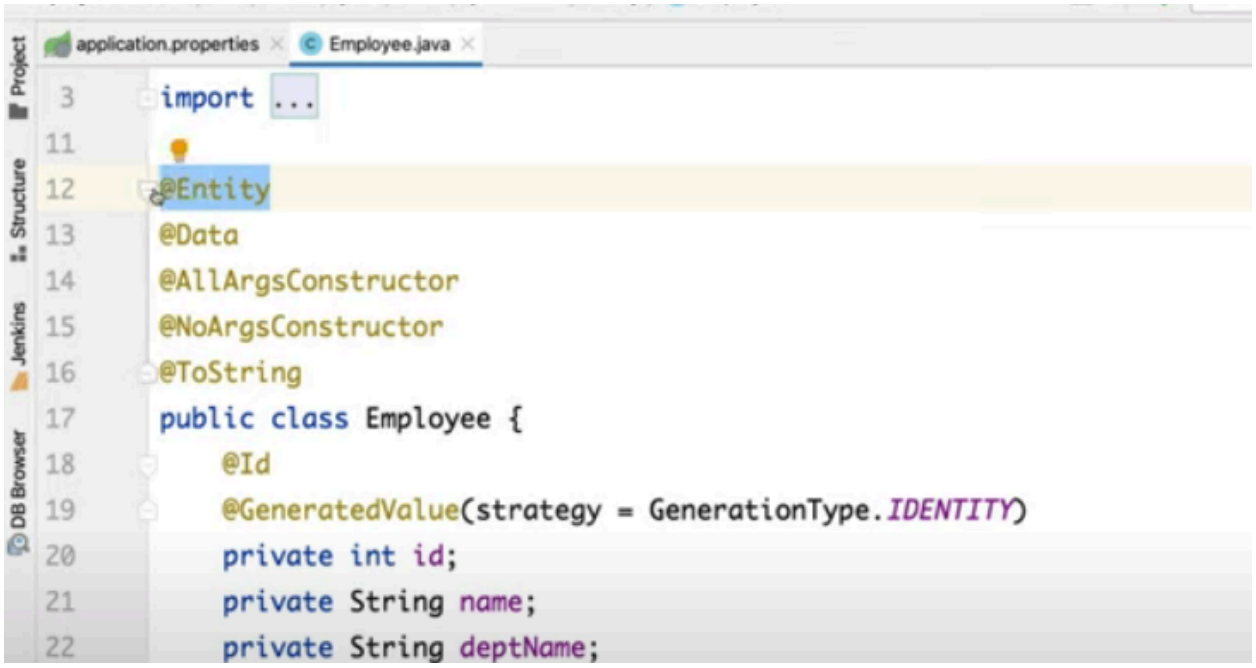
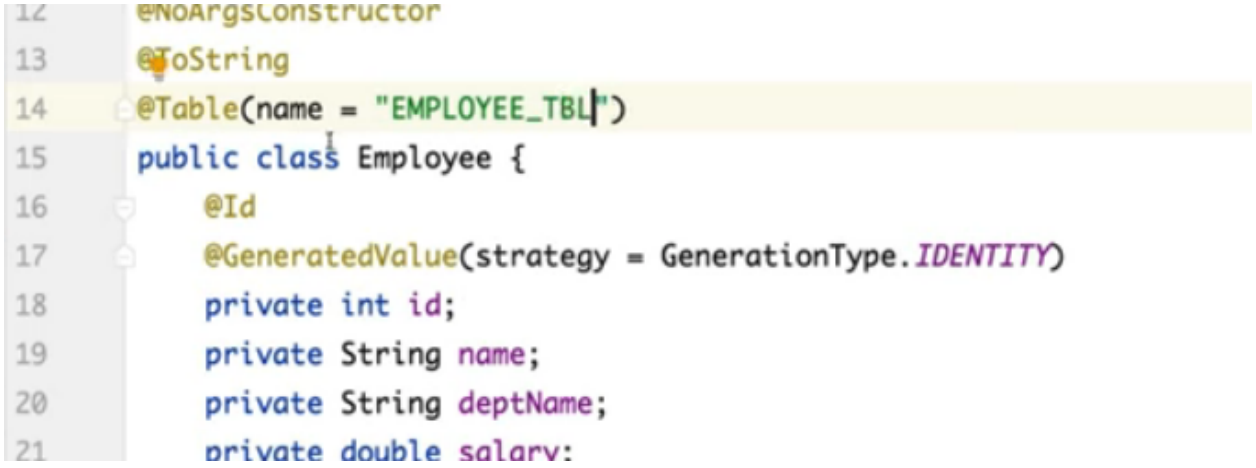
61. How does your application interact with the database and which frameworks are you using?

Create entity class. If you want to define any POJO as an entity which we want to store in DB then use this annotation. It will create table with the class name.

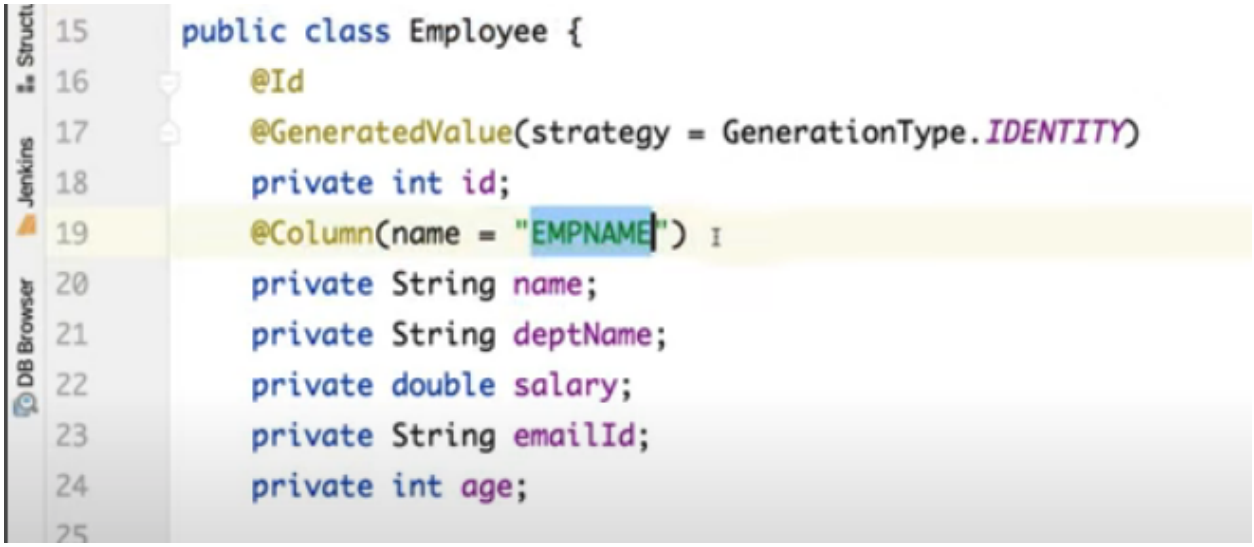


For custom table name use @Table annotation

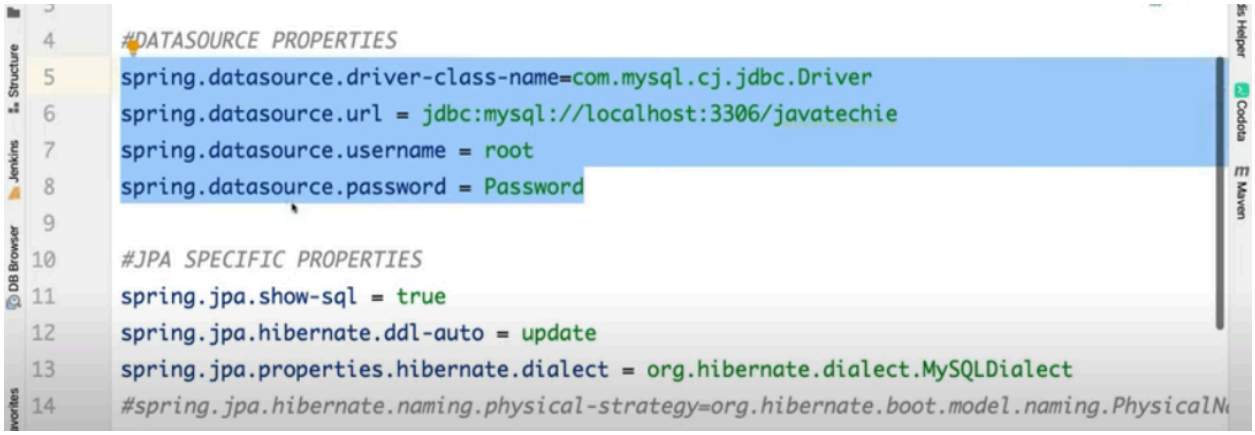


For primary we have to use @Id and @GeneratedValue annotations Identity for auto increment.

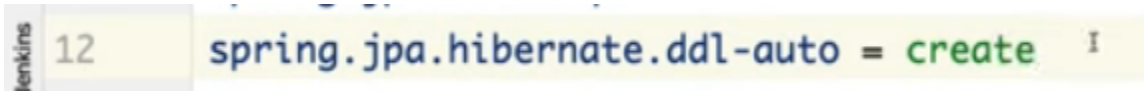
For customized column name use @Column annotation.



```
15  public class Employee {
16      @Id
17      @GeneratedValue(strategy = GenerationType.IDENTITY)
18      private int id;
19      @Column(name = "EMPNAME")
20      private String name;
21      private String deptName;
22      private double salary;
23      private String emailId;
24      private int age;
```

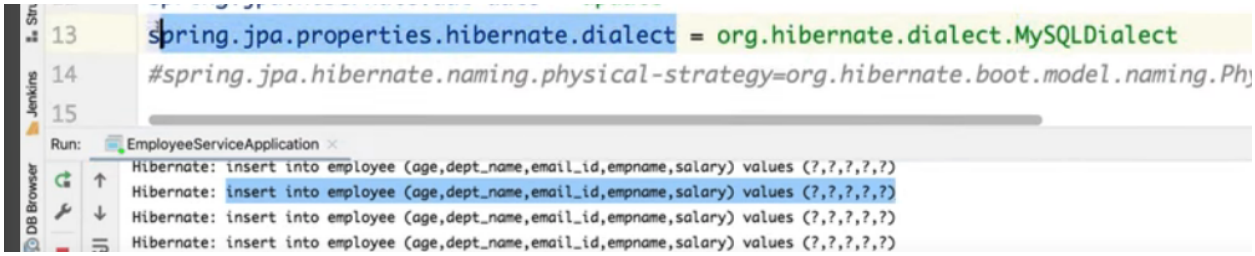


```
#DATASOURCE PROPERTIES
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/javatechie
spring.datasource.username = root
spring.datasource.password = Password

#JPA SPECIFIC PROPERTIES
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect
#spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalN
```

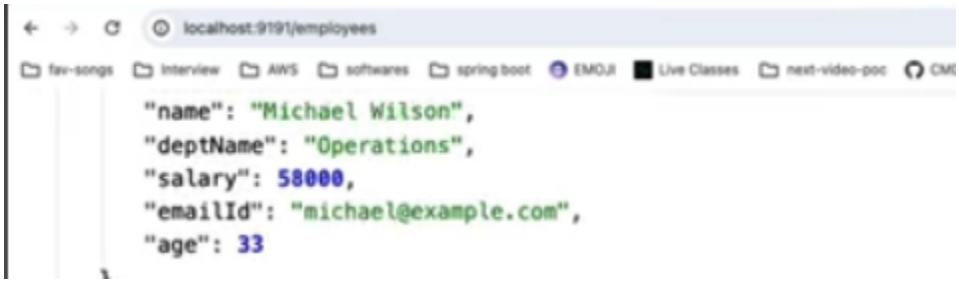


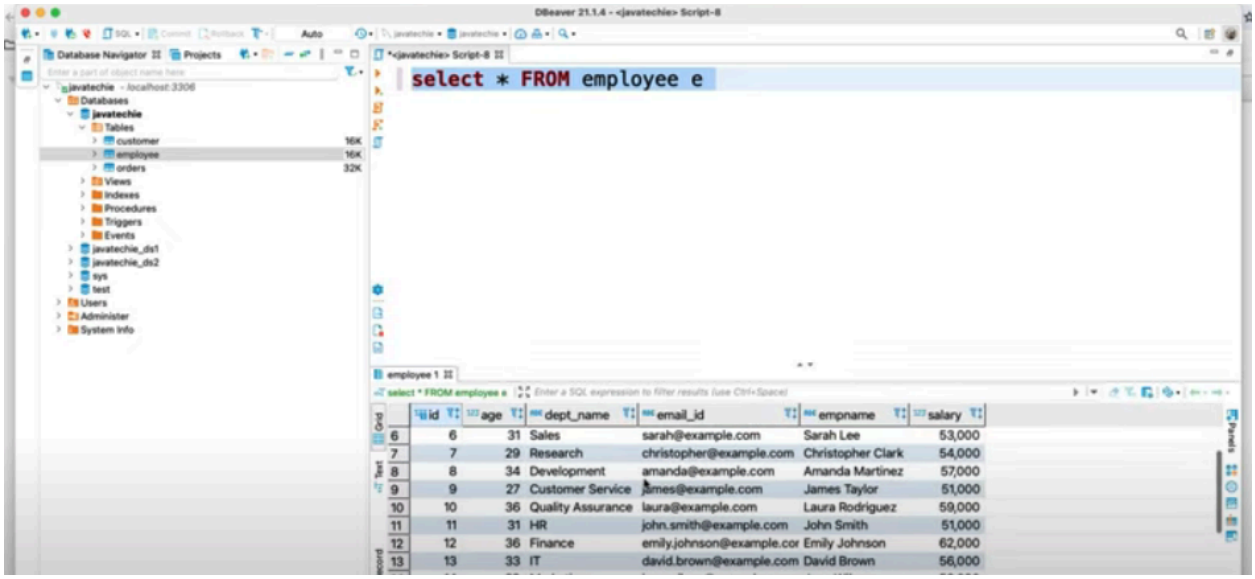
```
12  spring.jpa.hibernate.ddl-auto = create
```

If we use ddl-auto as create, it will always create table.
If we use ddl-auto as an update, it will create a table if it does not exist. If it exists it only updates the table.
Best practice is always update.
Use create only in integration / test / cucumber properties but do not for main properties - explore



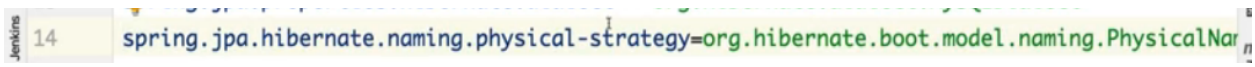
Every DB has dialect property its use is fine tune the query on behalf of developer





Camel case properties are created with underscore separator column names
Example property deptName = dept_name

62. Why is it important to configure a physical naming strategy?



```
14 }.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
15
```

employee 1

select * FROM employee e

	id	age	deptName	emailId	EMPNAME	salary
1	1	30	HR	john@example.com	John Doe	50,000
2	2	35	Finance	jane@example.com	Jane Smith	60,000

If you do not want hibernate to create column names and we want java property as column name then use physical-strategy as shown below. Now see the column name.

63. What are the key benefits of using Spring Data JPA ?

```
1  package com.javatechie.repository;
2
3  import ...
10
11 public interface EmployeeRepository extends JpaRepository<Employee,Integer> {
12
13 }
14
```

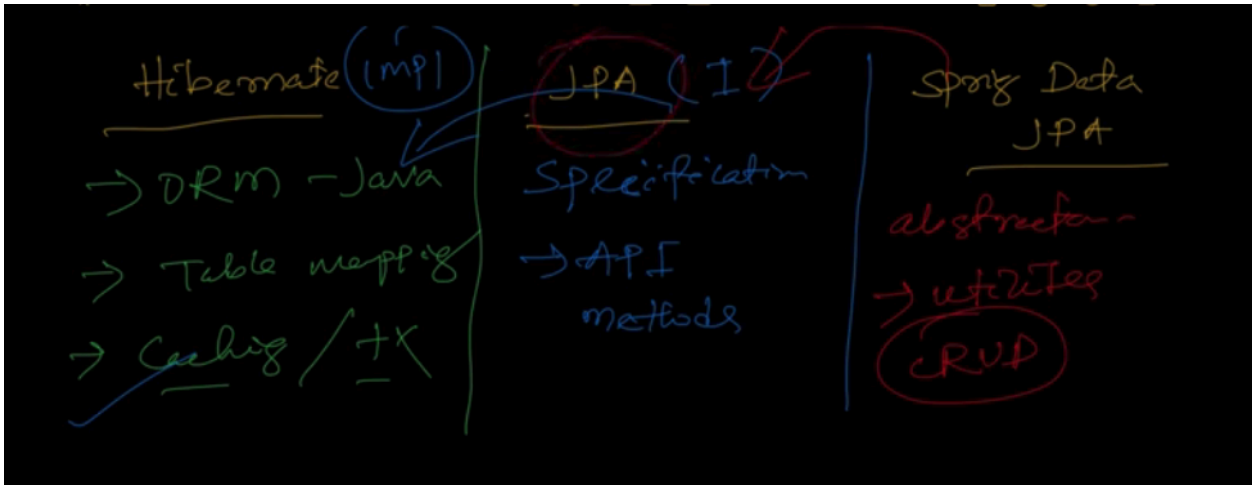
This Employee Repository will perform all CURD operations on the entity Employee.
It reduced Boiler plate code. We have direct methods to play with repository

```
@Autowired
private EmployeeRepository repository;

public Employee saveEmployee(Employee employee) {
    repository.find
```

findAll(Sort sort) List<Employee>
findAll(Pageable pageable) Page<Employee>
findAll(Example<S> example) List<S>
findAll(Example<S> example, Sort sort) List<S>
findAll(Example<S> example, Pageable pageable) Page<S>
findAllById(Iterable<Integer> ids) List<Employee>
findBy(Example<S> example, Function<FetchableFluentQuery<... R
findById(Integer id) Optional<Employee>
findOne(Example<S> example) Optional<S>

We do not need to write a query in the code. We can cover the query into methods very easily.

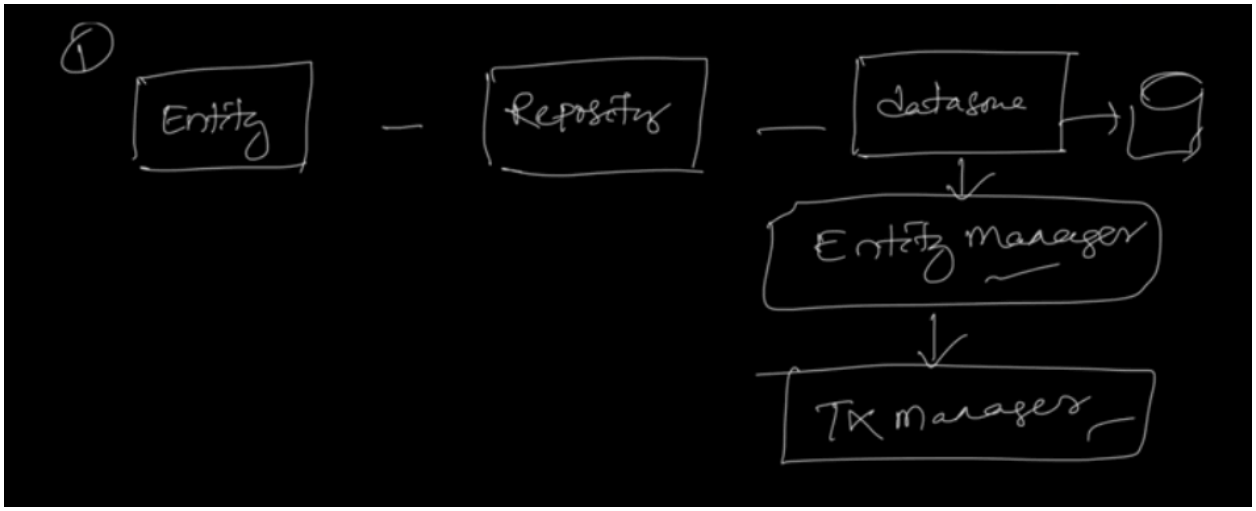
64. What are the differences between Hibernate, JPA, and Spring Data JPA?



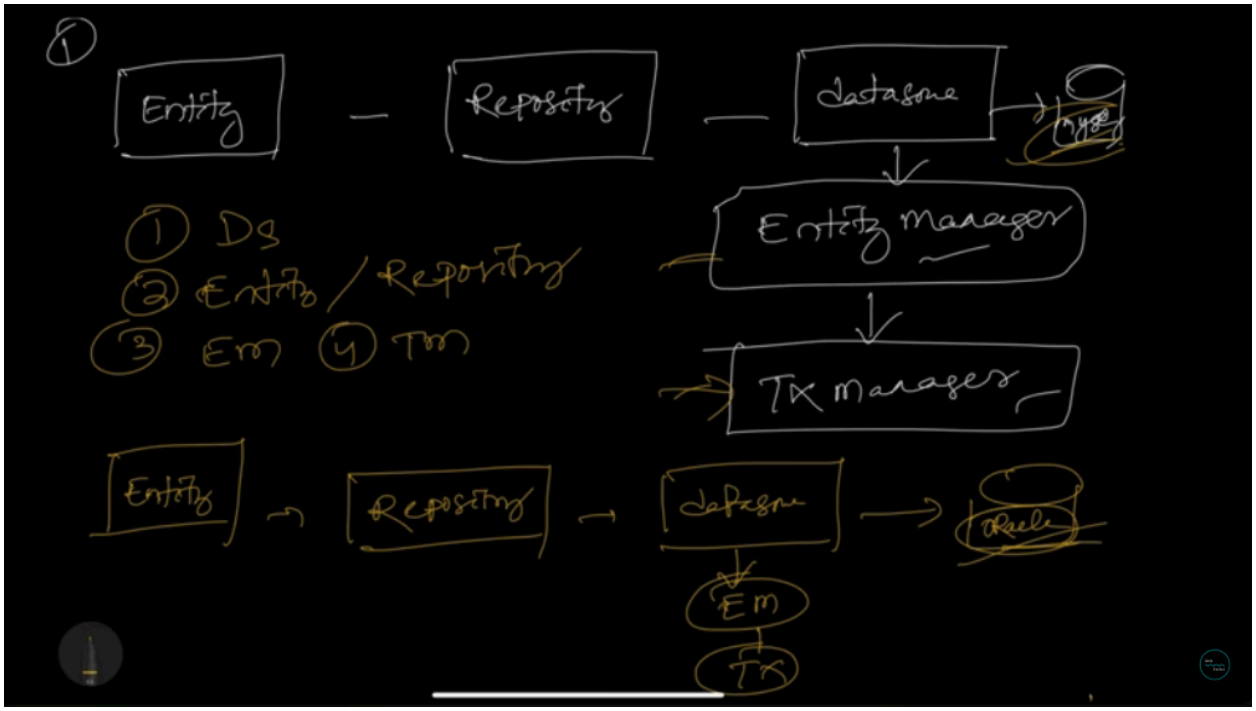
- Hibernate is ORM Tool For Java, it takes care of java to table mapping and caching and transaction.
- JPA is Interface / specification. Hibernate is implementation class of JPA interface
- Spring Data JPA is another level of abstraction on JPA which allows us to perform CRUD operations and custom methods like findByDeptName / findBySalaryAndAge

65. How can you connect multiple databases or data sources in a single application?

Oracle table for payments and sql table for orders. How you connect them.



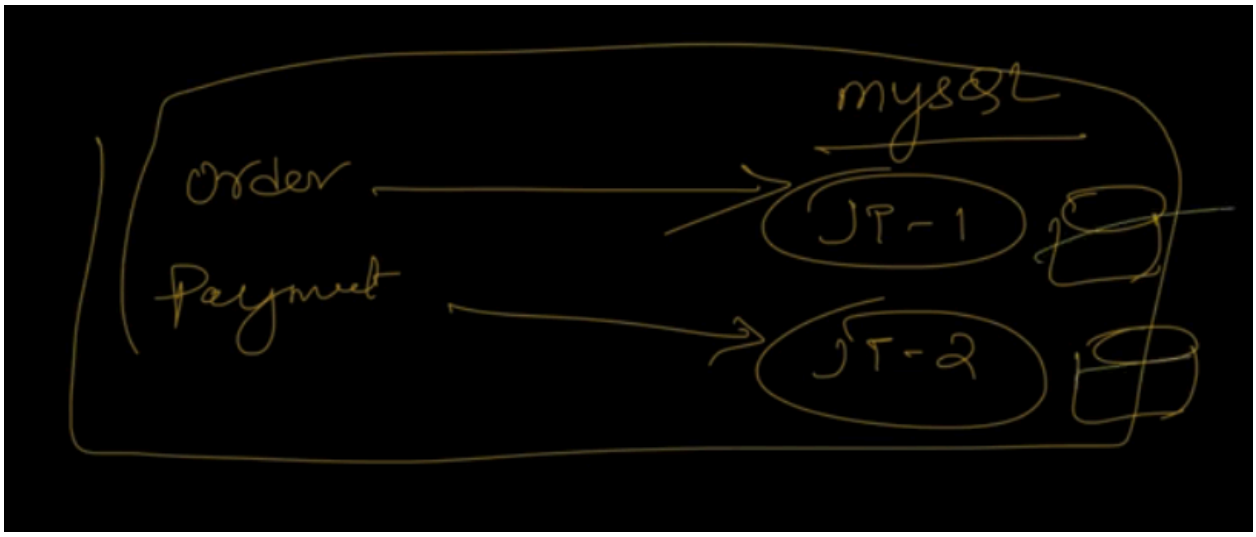
So far we have created an entity, a repository and datasource for one DB. Datasource will create entity manager and transaction manager.



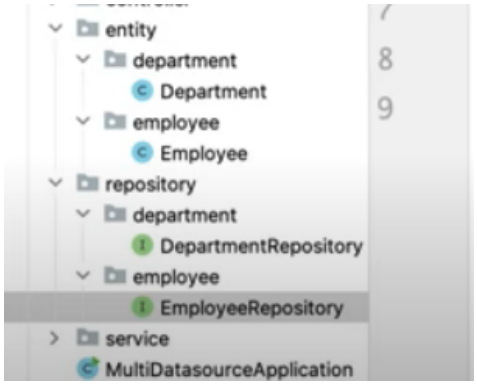
Similarly we have to create an entity, repository and data source for the other DB. Datasource will create entity manager and transaction manager.

Because we are using different DBs we have to do customization for entity manager and transaction manager to handle separately.

Let us see how this can be performed. But tutor used only SQL with different schemas to explain this concept

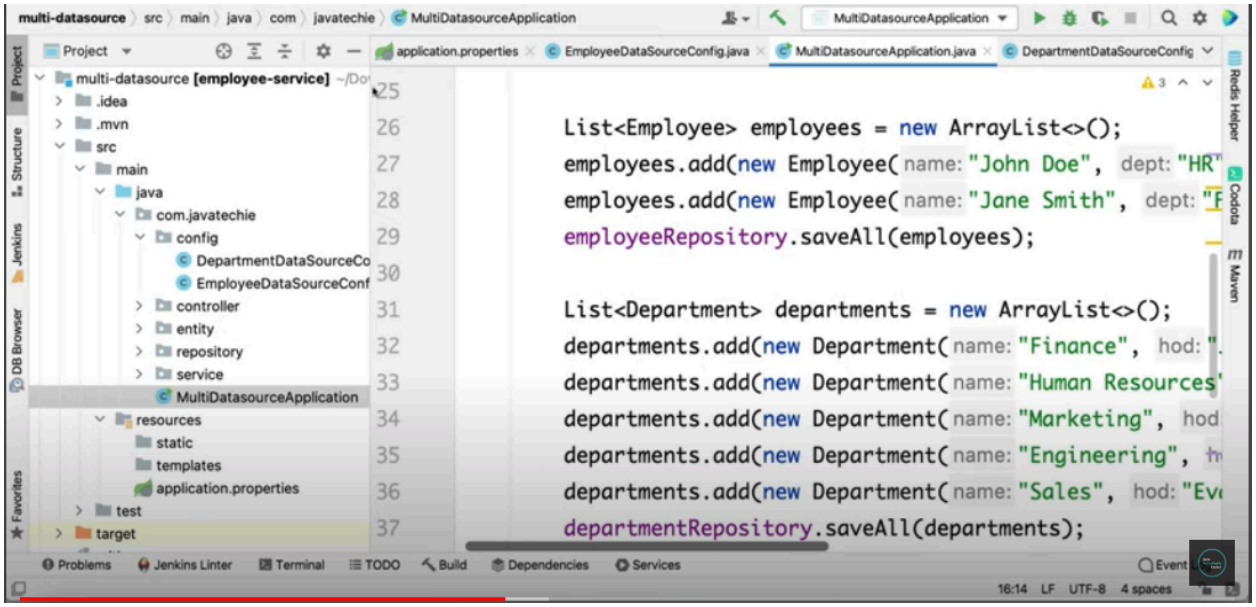


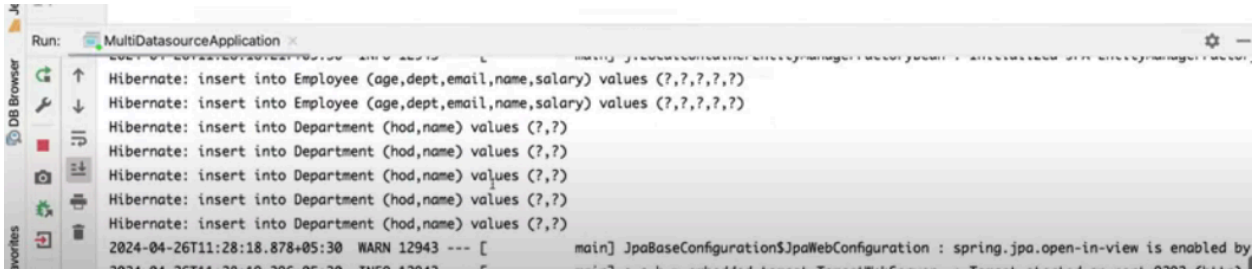
We have two entity and repository and data source

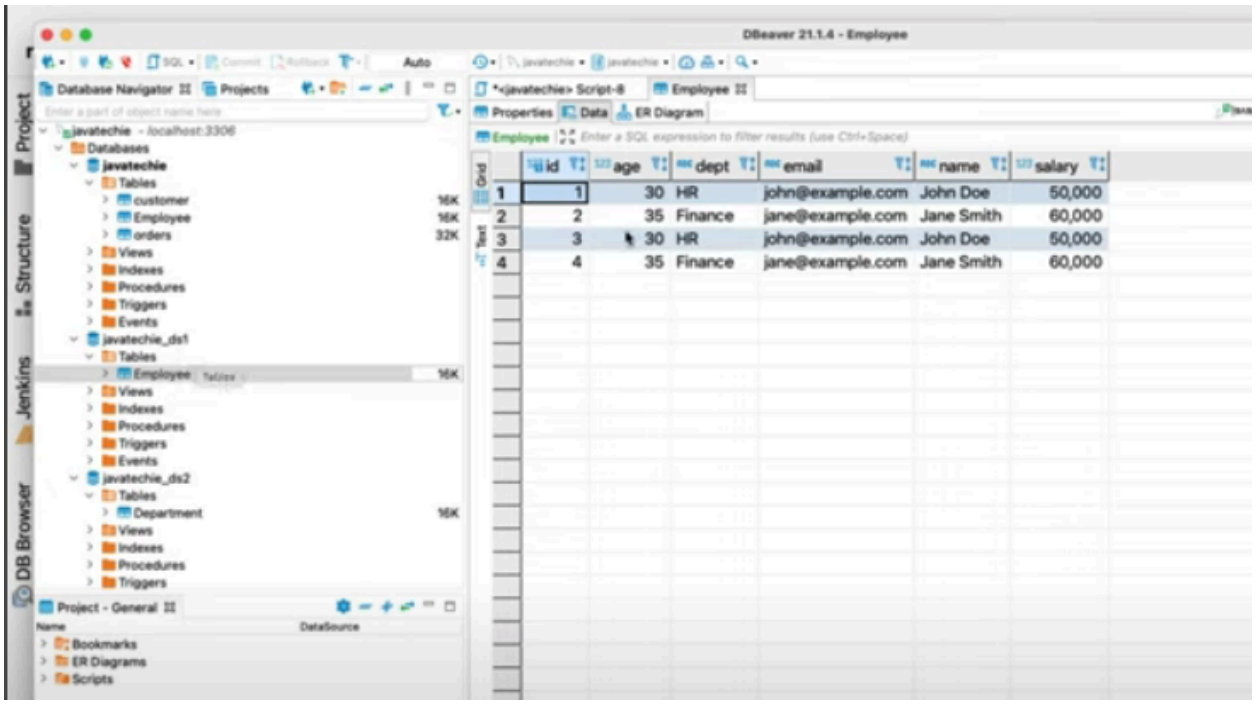


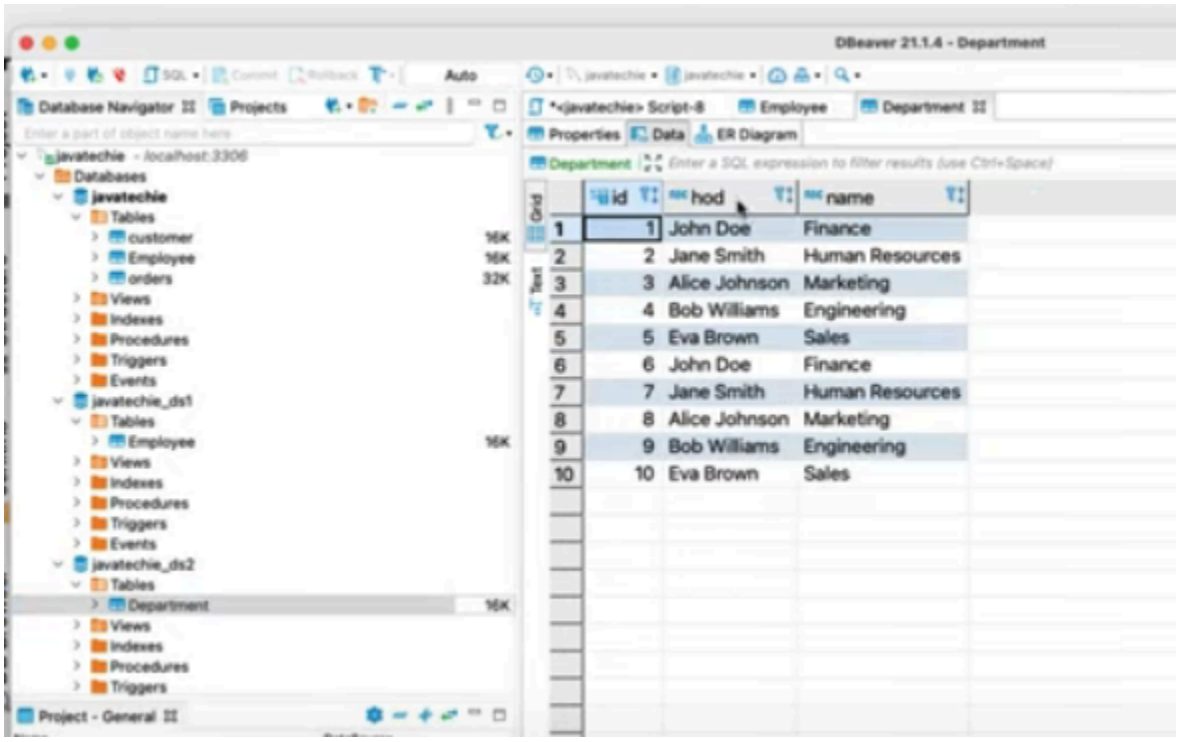

```
#EMPLOYEE DATASOURCE PROPERTIES
spring.datasource.employee.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.employee.url = jdbc:mysql://localhost:3306/javatechie_ds1
spring.datasource.employee.username = root
spring.datasource.employee.password = Password

#DEPARTMENT DATASOURCE PROPERTIES
spring.datasource.department.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.department.url = jdbc:mysql://localhost:3306/javatechie_ds2
spring.datasource.department.username = root
spring.datasource.department.password = Password
```

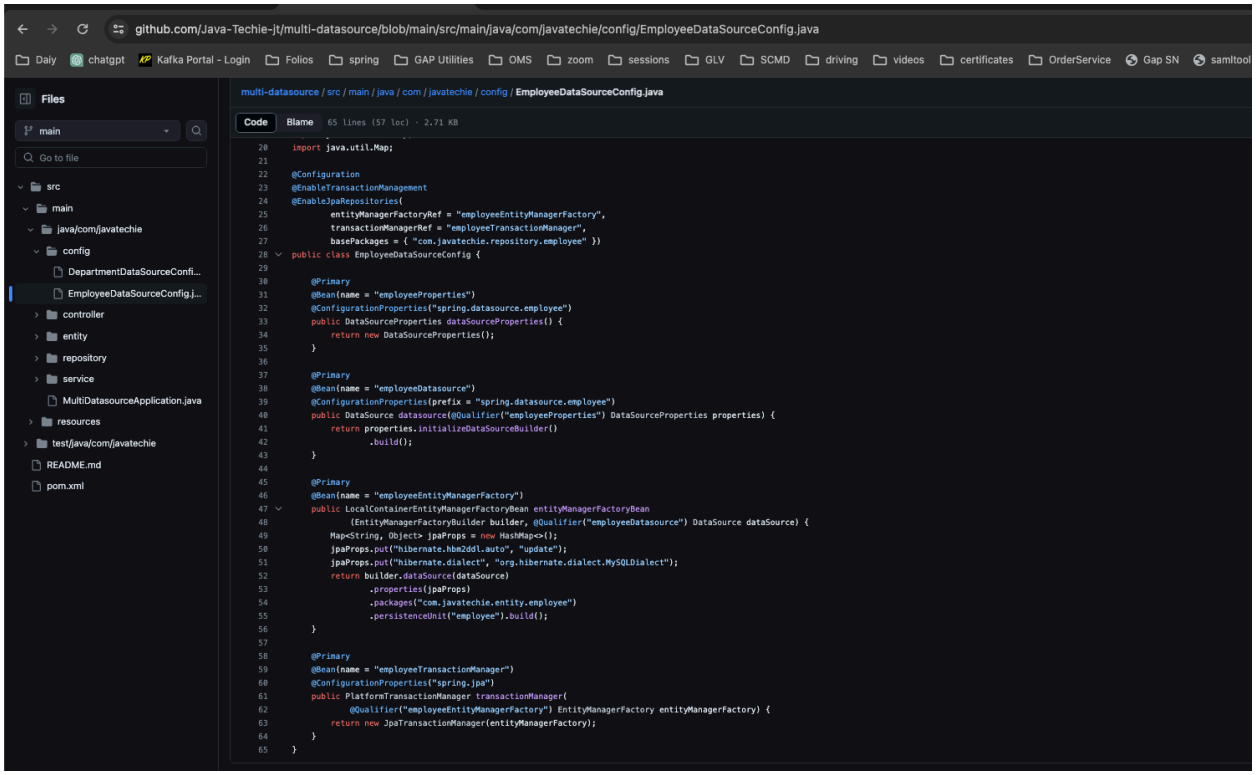
Let us create custom configuration



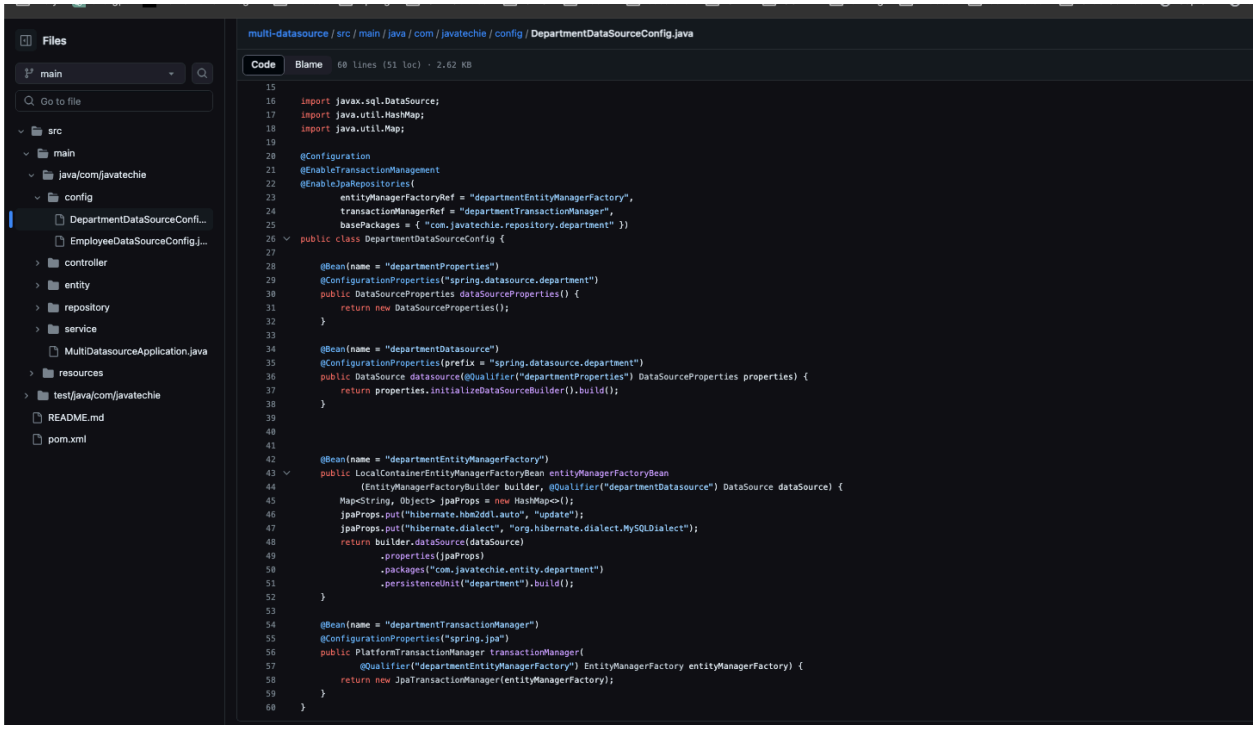








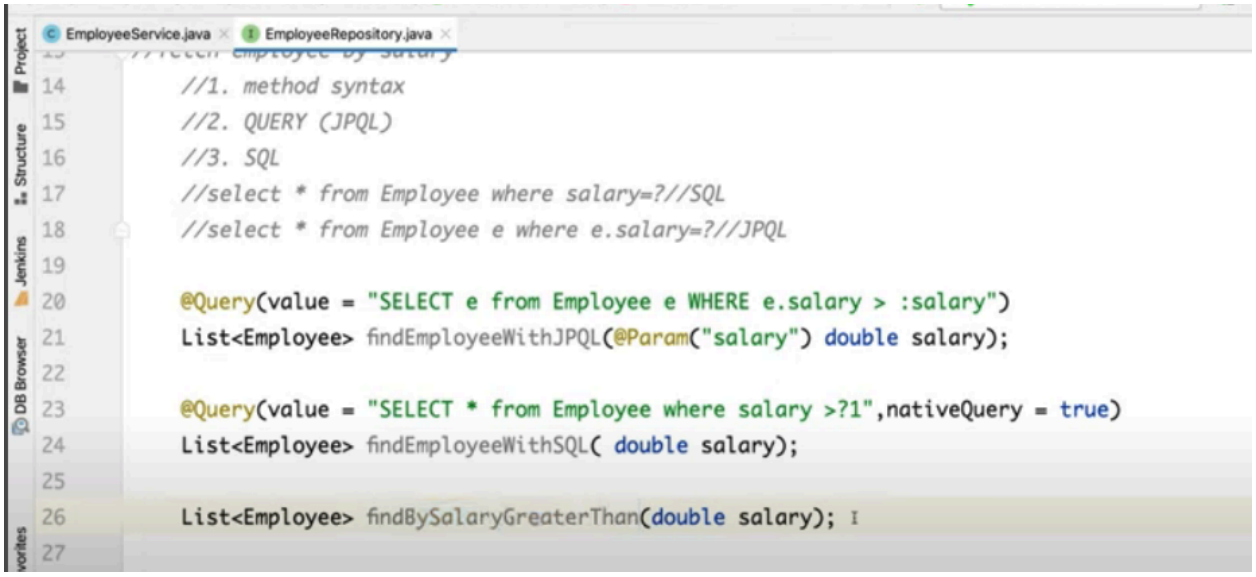
Whenever we have multiple databases we have make one as primary/default



66. What are the different ways to define custom queries in Spring Data JPA ?

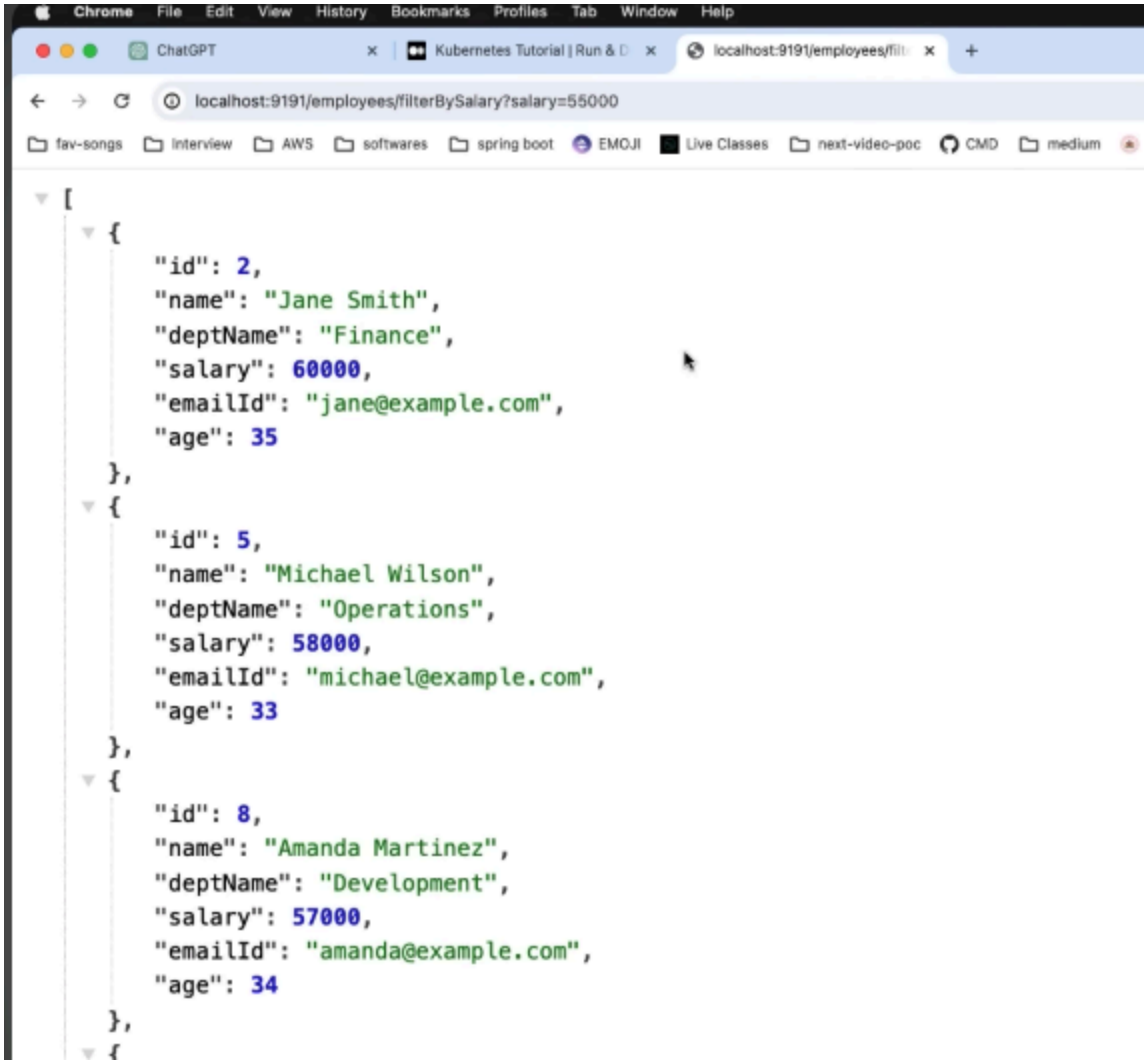
This can be done in three different ways

Using method syntax / JPQL (Java Persistence Query Language) / SQL



For JPQL Query and SQL - method names can be anything. For Method Syntax, the syntax name should have a specific format as shown in the above screen. Method syntax internally uses JPQL during execution.

Create Control for this.



If you keenly notice the logs for method and JPQL the queries are the same. For sql the query is different.

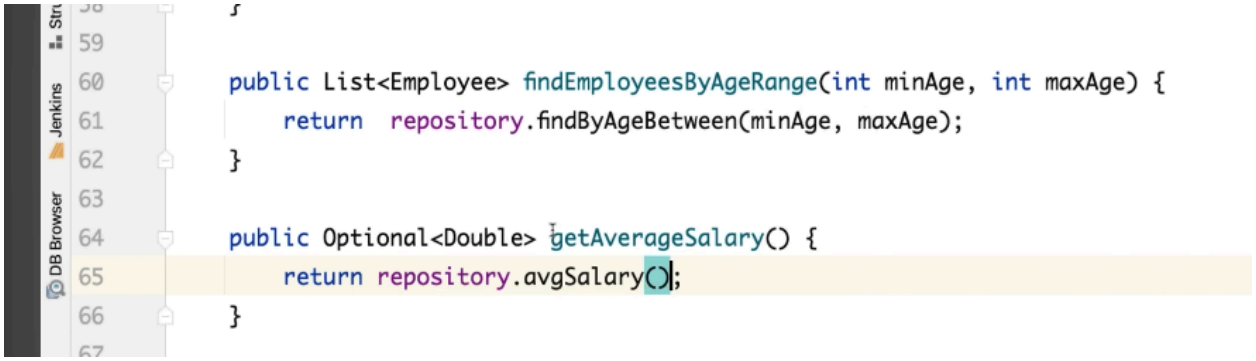
67. Can you provide example to find results based on some range

Ex 1 : Find list of employees whose age in between 30 lakh to 35 lakh
Ex 2 : find avg salary of employees in an organization

In the repository class

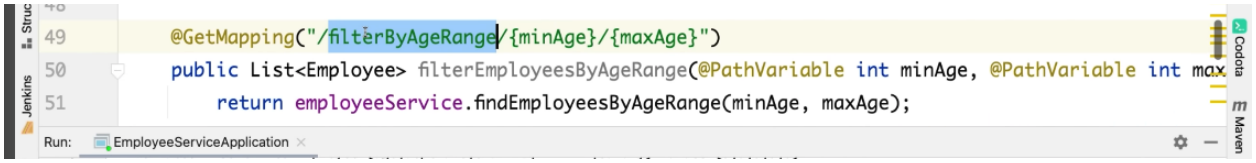
```
28
29      List<Employee> findByAgeBetween(int min, int max);
30
31  //    findBySalaryAvg
32  //    avgSalary();
33
34      @Query(value = "SELECT AVG(e.salary) FROM Employee e")
35      double avgSalary();
36
37  }
```

In the service class



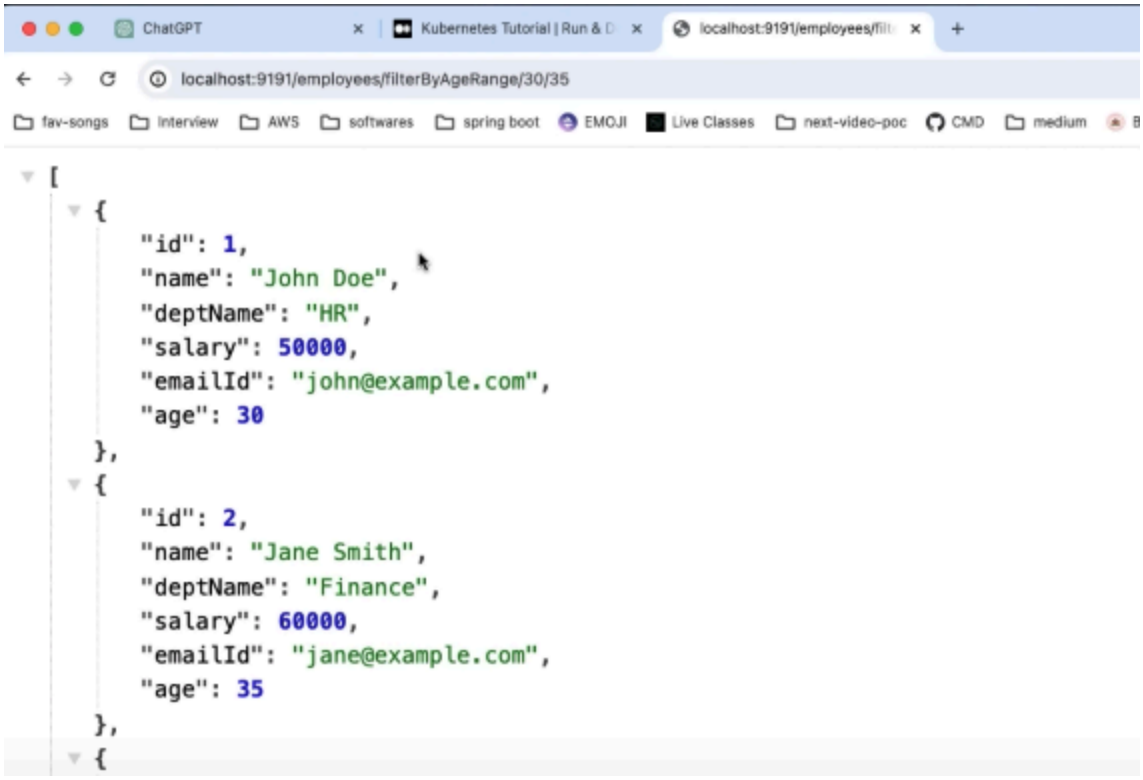
```
59
60      public List<Employee> findEmployeesByAgeRange(int minAge, int maxAge) {
61          return  repository.findByAgeBetween(minAge, maxAge);
62      }
63
64      public Optional<Double> getAverageSalary() {
65          return repository.avgSalary();
66      }
```

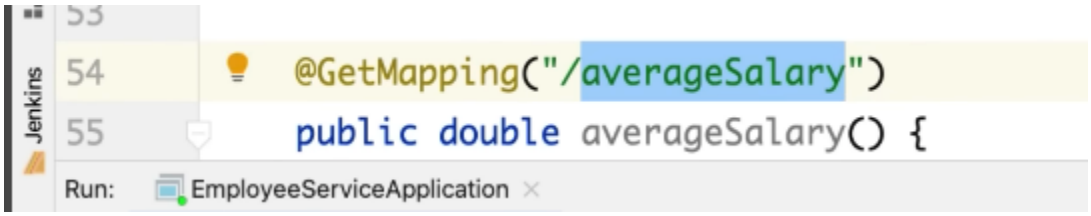
Controller

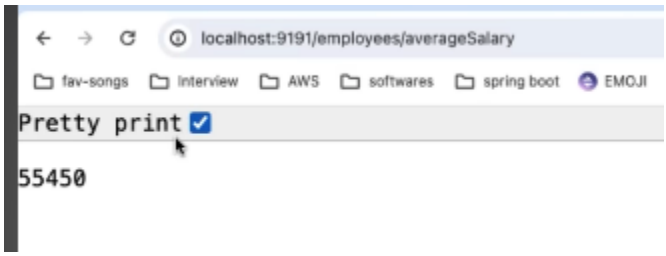


```
49      @GetMapping("/filterByAgeRange/{minAge}/{maxAge}")
50      public List<Employee> filterEmployeesByAgeRange(@PathVariable int minAge, @PathVariable int max
51          return employeeService.findEmployeesByAgeRange(minAge, maxAge);
```

Run: EmployeeServiceApplication

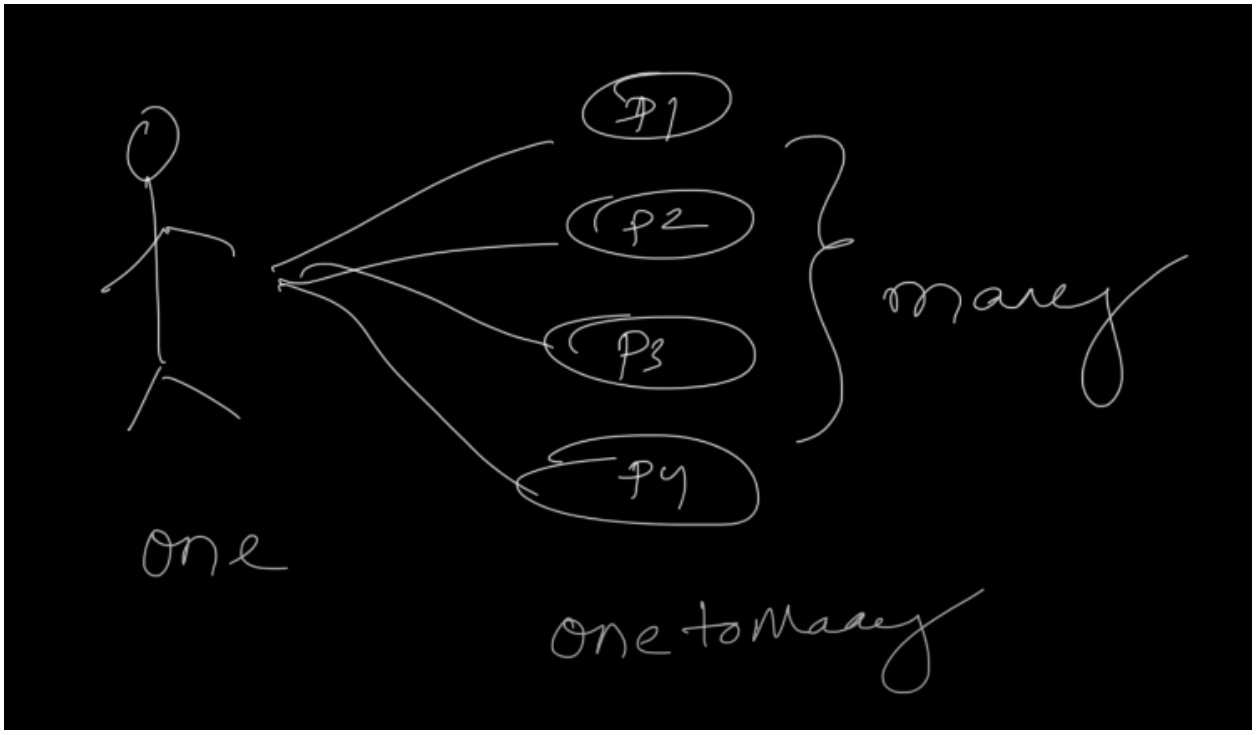


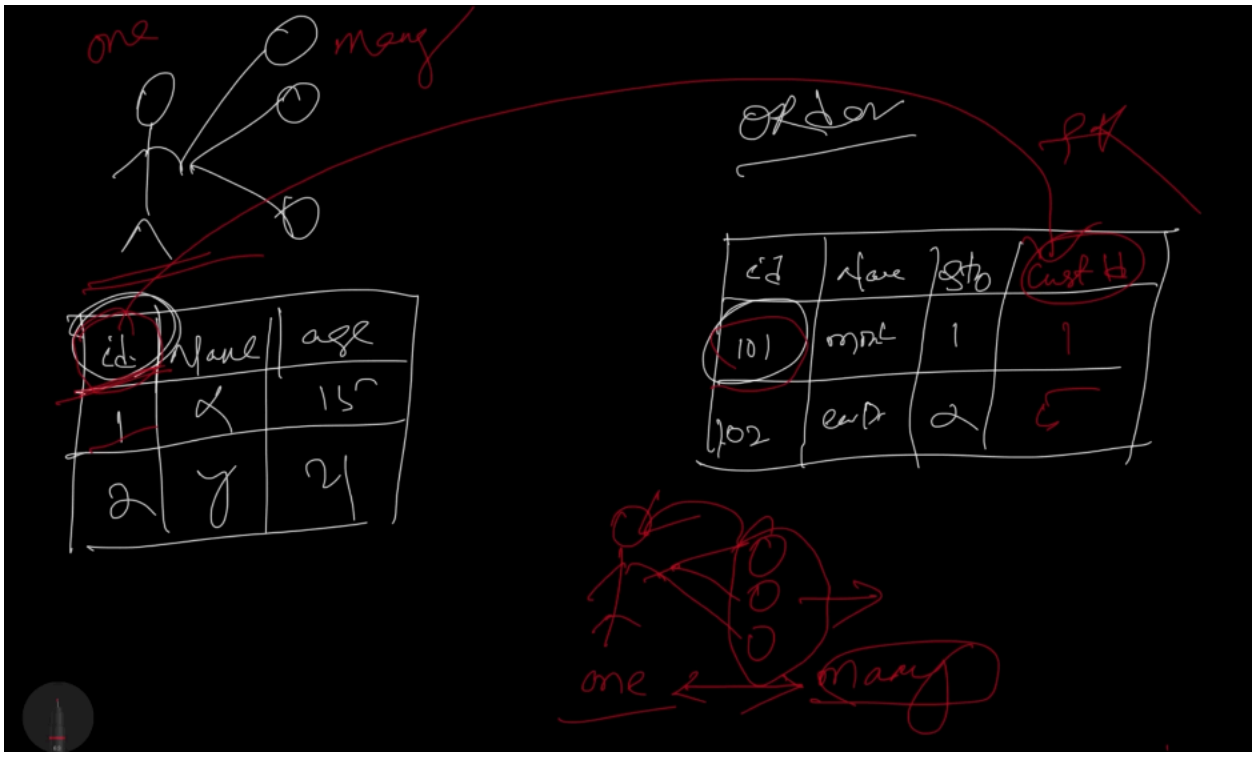




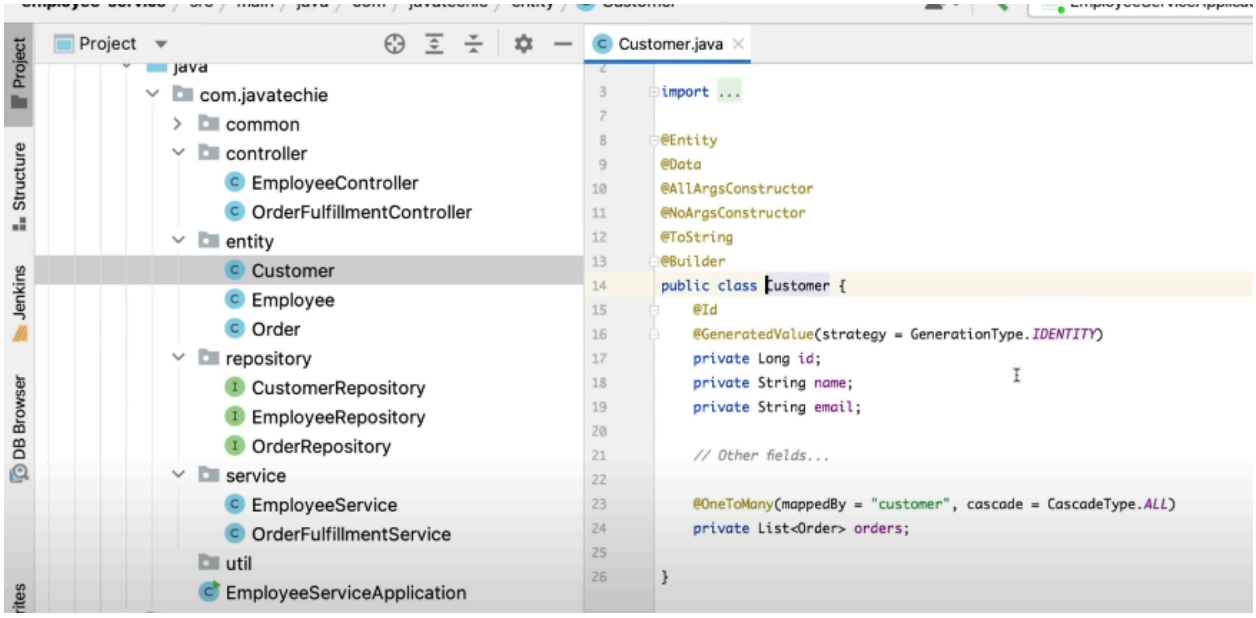
68. How will you define entity relationships or association mapping in Spring Data JPA?

(Use Case: Consider an e-commerce application where each customer can have multiple orders. Here, a Customer entity can be associated with multiple Order entities.)

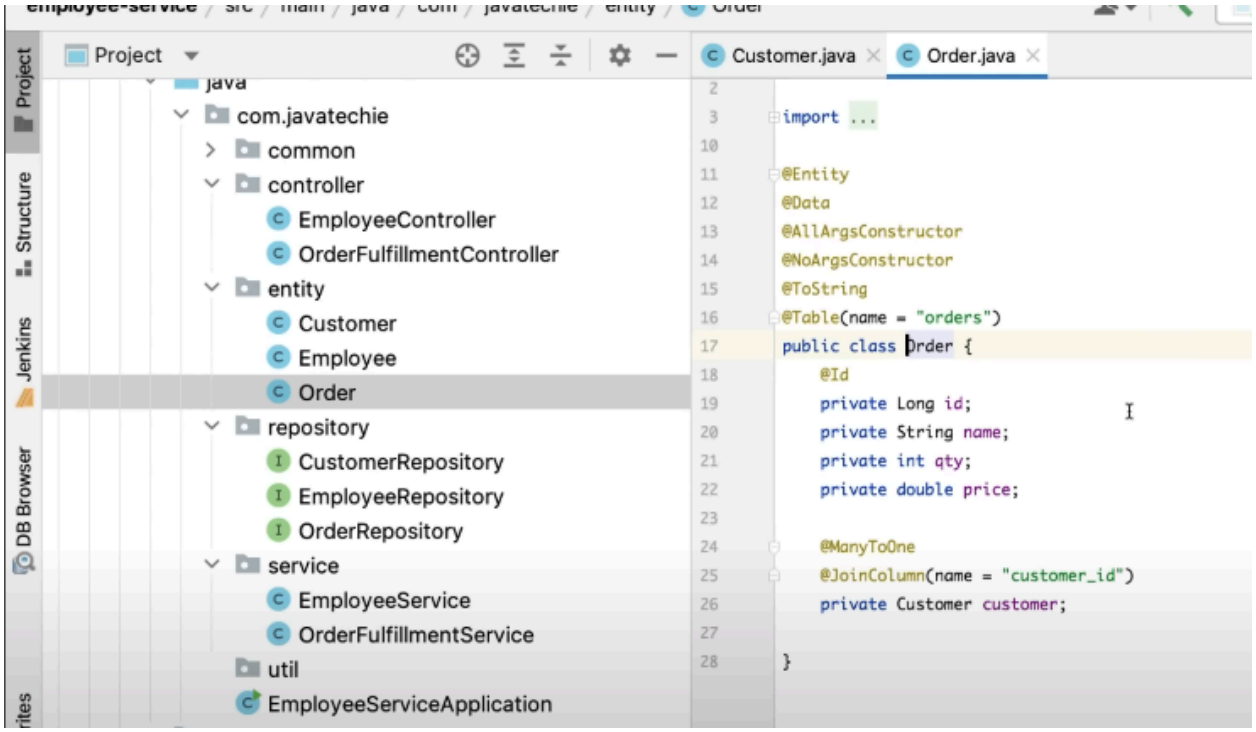

P1
P2
P3
P4
many
one
one to many


one
many
Order
FK
id
Name
age
cid
Name
Sno
Cust H
101
1
1
102
2
one
many

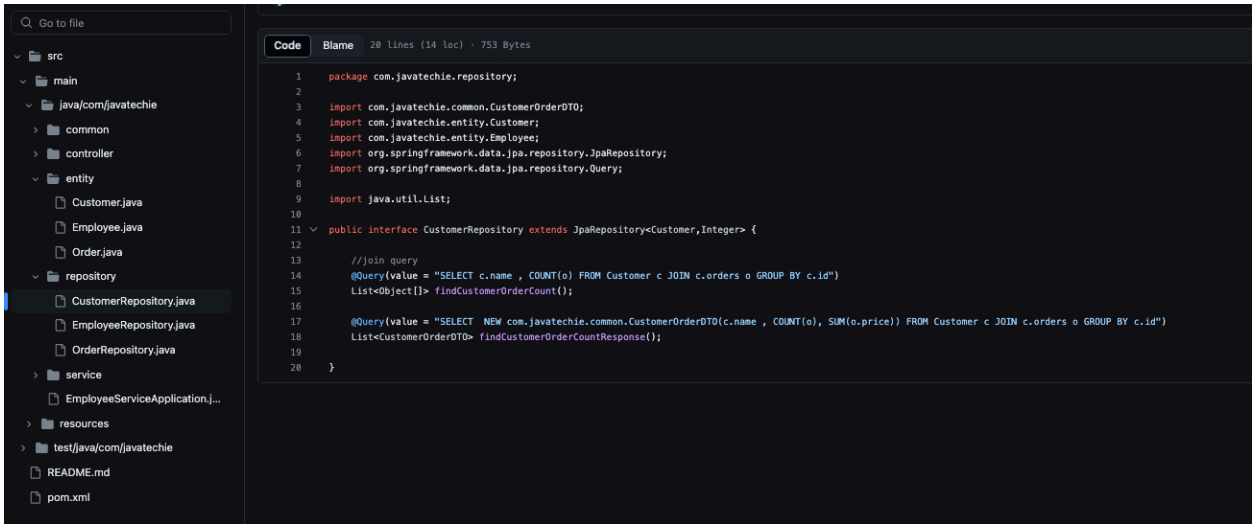
Notice line number 23 in customer entity where we specified relationship with order entity CascadeType all will insert data into order entity.

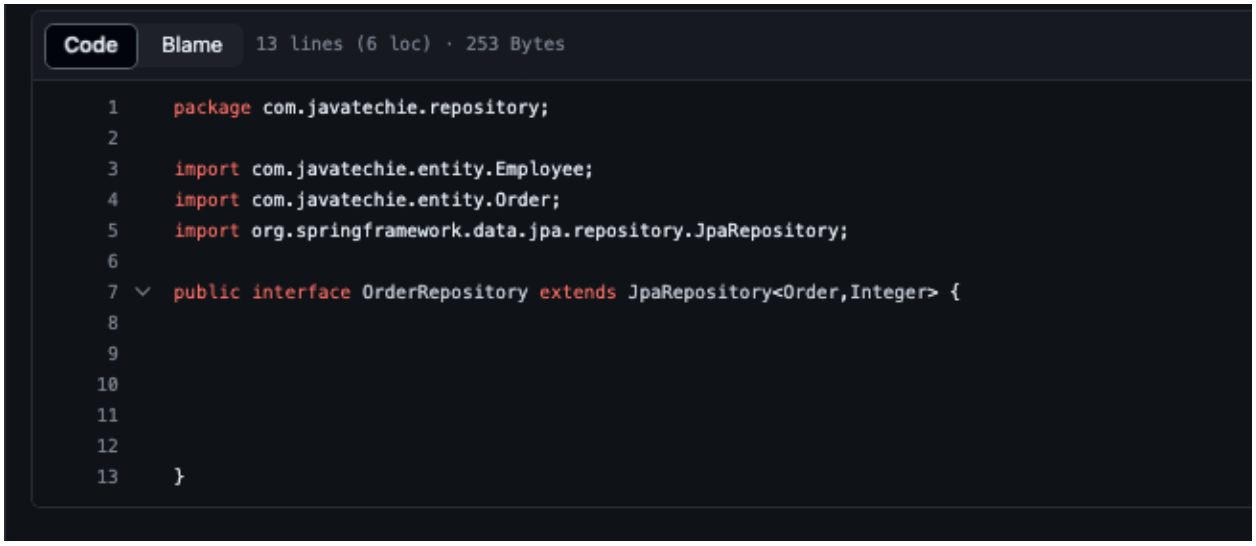


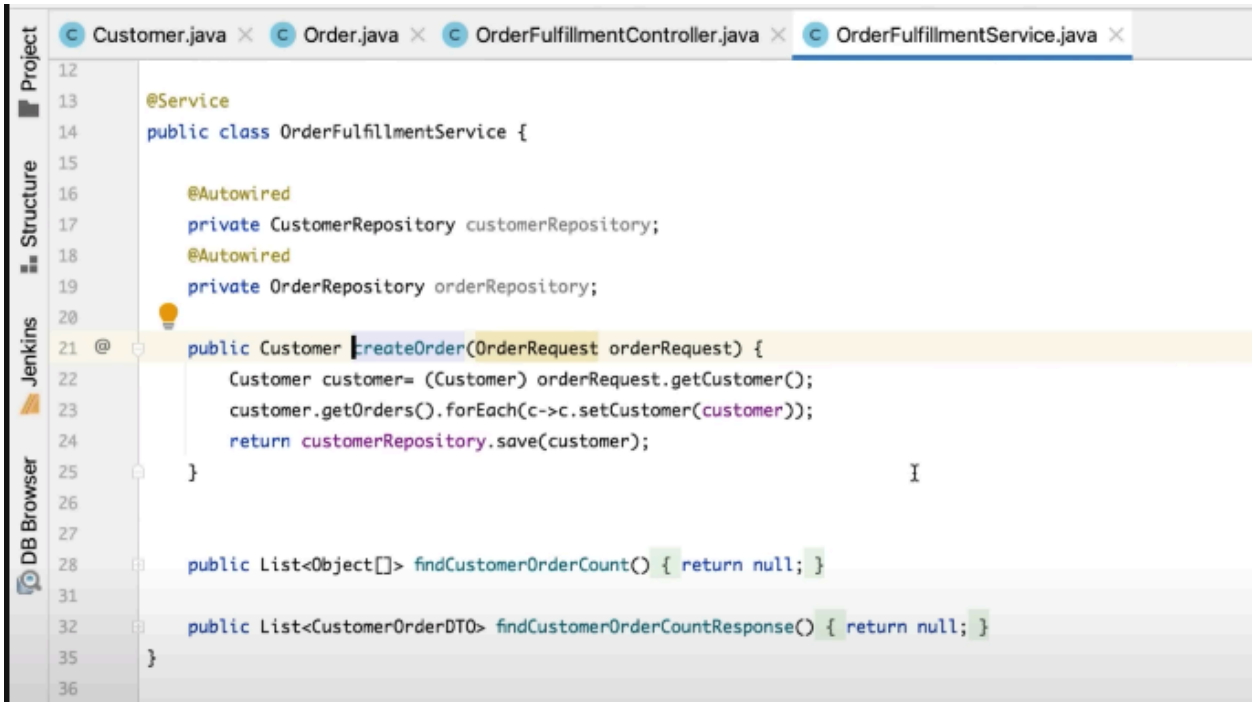
Notice line number 24 in order entity where we specified relationship with customer entity



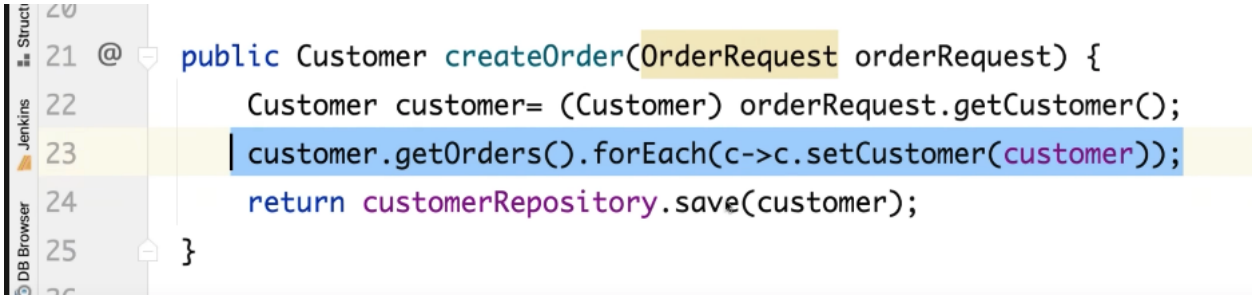
And we have corresponding two repositories

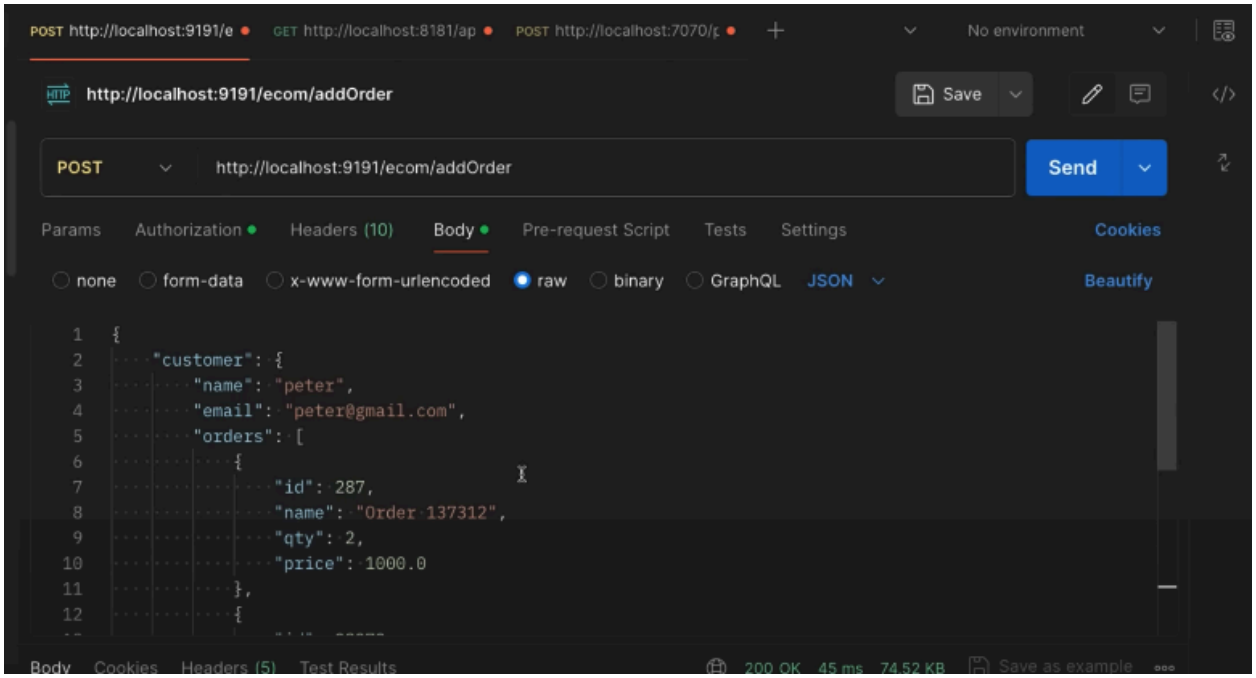

Go to file
src
main
java/com/javatechie
common
controller
entity
Customer.java
Employee.java
Order.java
repository
CustomerRepository.java
EmployeeRepository.java
OrderRepository.java
service
EmployeeServiceApplication.j...
resources
test/java/com/javatechie
README.md
pom.xml
Code
Blame
20 lines (14 loc) · 753 Bytes
1 package com.javatechie.repository;
2
3 import com.javatechie.common.CustomerOrderDTO;
4 import com.javatechie.entity.Customer;
5 import com.javatechie.entity.Employee;
6 import org.springframework.data.jpa.repository.JpaRepository;
7 import org.springframework.data.jpa.repository.Query;
8
9 import java.util.List;
10
11 public interface CustomerRepository extends JpaRepository<Customer,Integer> {
12
13 //join query
14 @Query(value = "SELECT c.name , COUNT(o) FROM Customer c JOIN c.orders o GROUP BY c.id")
15 List<Object[]> findCustomerOrderCount();
16
17 @Query(value = "SELECT NEW com.javatechie.common.CustomerOrderDTO(c.name , COUNT(o), SUM(o.price)) FROM Customer c JOIN c.orders o GROUP BY c.id")
18 List<CustomerOrderDTO> findCustomerOrderCountResponse();
19
20 }

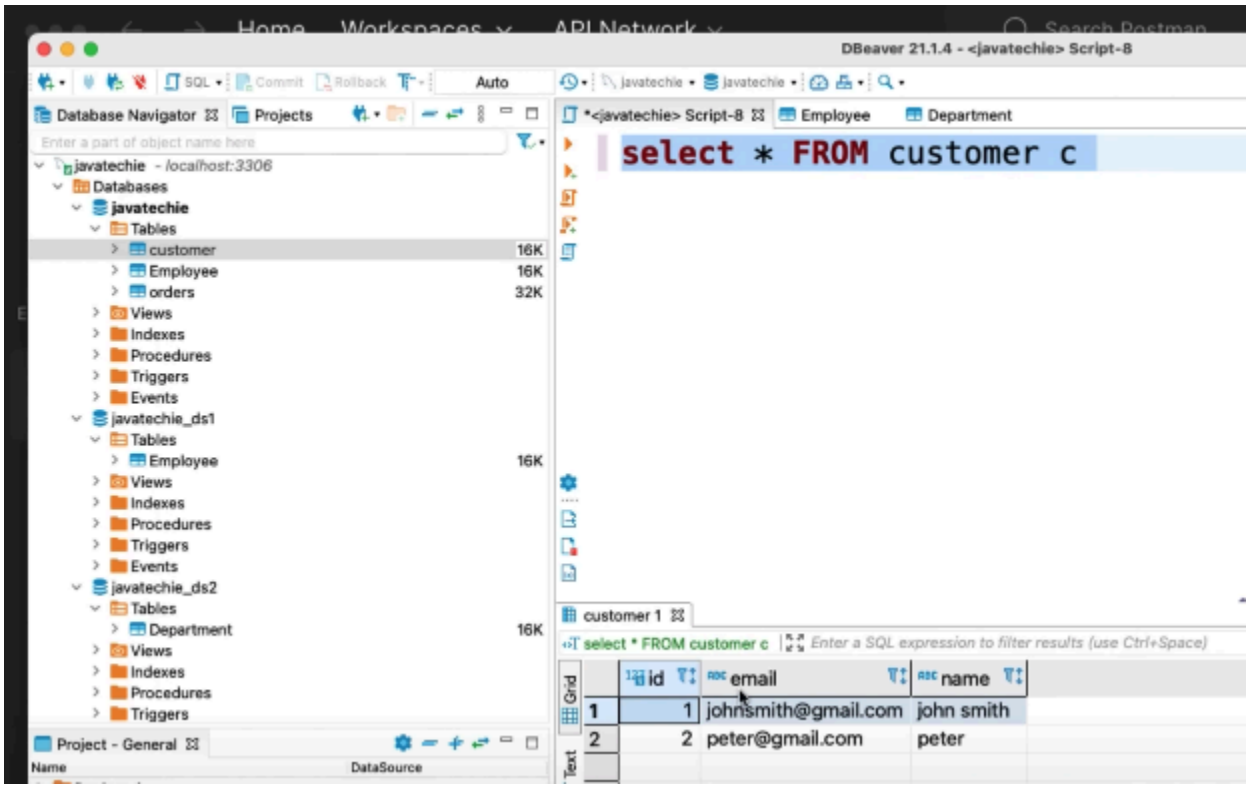

Code
Blame
13 lines (6 loc) · 253 Bytes
1 package com.javatechie.repository;
2
3 import com.javatechie.entity.Employee;
4 import com.javatechie.entity.Order;
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 public interface OrderRepository extends JpaRepository<Order,Integer> {
8
9
10
11
12
13 }



The highlighted area is for many to one logic




POST http://localhost:9191/e
GET http://localhost:8181/ap
POST http://localhost:7070/
No environment
http://localhost:9191/ecom/addOrder
Save
POST
http://localhost:9191/ecom/addOrder
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"customer": {
"name": "peter",
"email": "peter@gmail.com",
"orders": [
{
"id": 287,
"name": "Order 137312",
"qty": 2,
"price": 1000.0
},
{
Body
Cookies
Headers (5)
Test Results
200 OK
45 ms
74.52 KB
Save as example


Home
Workspaces
API Network
Search Postman
DBeaver 21.1.4 - <javatechie> Script-8
Database Navigator
Projects
Enter a part of object name here
javatechie - localhost:3306
Databases
javatechie
Tables
customer 16K
Employee 16K
orders 32K
Views
Indexes
Procedures
Triggers
Events
javatechie_ds1
Tables
Employee 16K
Views
Indexes
Procedures
Triggers
Events
javatechie_ds2
Tables
Department 16K
Views
Indexes
Procedures
Triggers
Project - General
Name
DataSource
*<javatechie> Script-8
Employee
Department
select * FROM customer c
customer 1
select * FROM customer c
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
id
email
name
1 1 johnsmith@gmail.com john smith
2 2 peter@gmail.com peter

```
select * FROM orders o
```

orders 1

select * FROM orders o | Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	price	qty	customer_id
1	112	Order 112	25	2	1
2	186	Order 186	30	1	1
3	273	Order 273	20	3	1
4	287	Order 137312	1,000	2	2
5	298	Order 298	20	3	1
6	432	Order 24e3	12,020	3	2
7	28,372	Order 4t746	30,000	1	2

69. Is this possible to execute Join query in Spring Data JPA ? If yes, how can you add some insights ?

As per tutor it is not recommended - because each micro service has its own database. However in the following way we can address above question

```
//customer name , count of product
```

We want customer name from customer entity and count of product from order table for that customer

```
11  public interface CustomerRepository extends JpaRepository<Customer,Integer> {
12
13      //join query
14      @Query(value = "SELECT c.name , COUNT(o) FROM Customer c JOIN c.orders o GROUP BY c.id")
15      List<Object[]> findCustomerOrderCount();
16
```

Service class

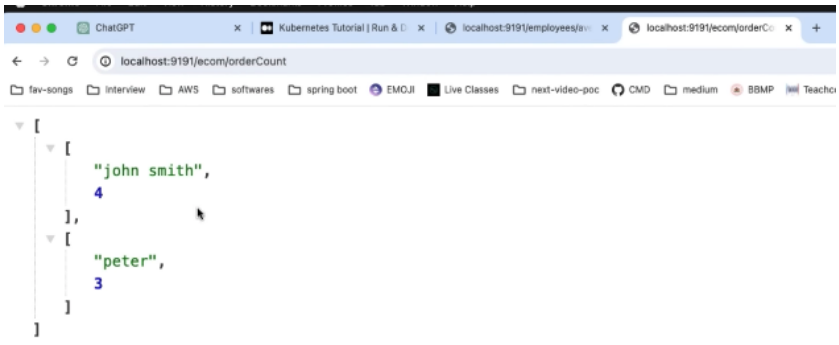
```
28      public List<Object[]> findCustomerOrderCount() {
29          return customerRepository.findCustomerOrderCount();
30      }
```

Controller class

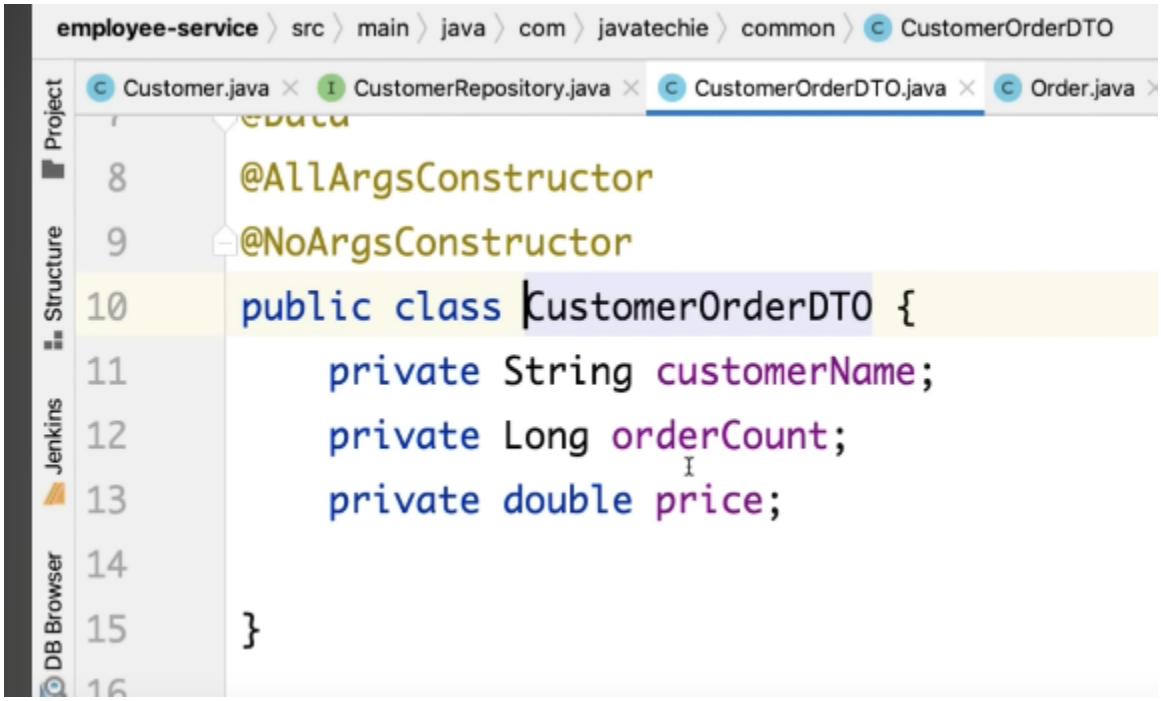
```
12  @RestController
13  @RequestMapping("/ecom")
14  public class OrderFulfillmentController {
15
16      @Autowired
17      private OrderFulfillmentService orderFulfillmentService;
18
19      @PostMapping("/addOrder")
```

```
23
24      @GetMapping("/orderCount")
25      public List<Object[]> getCustomerOrderCount() {
26          return orderFulfillmentService.findCustomerOrderCount();
27      }
```

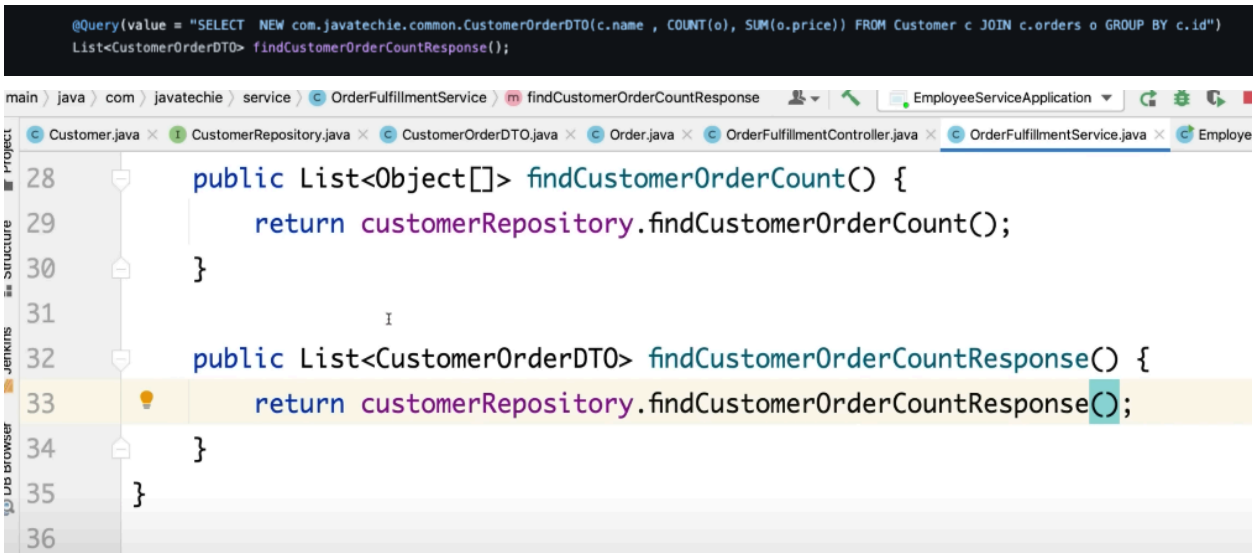
Response came in array format



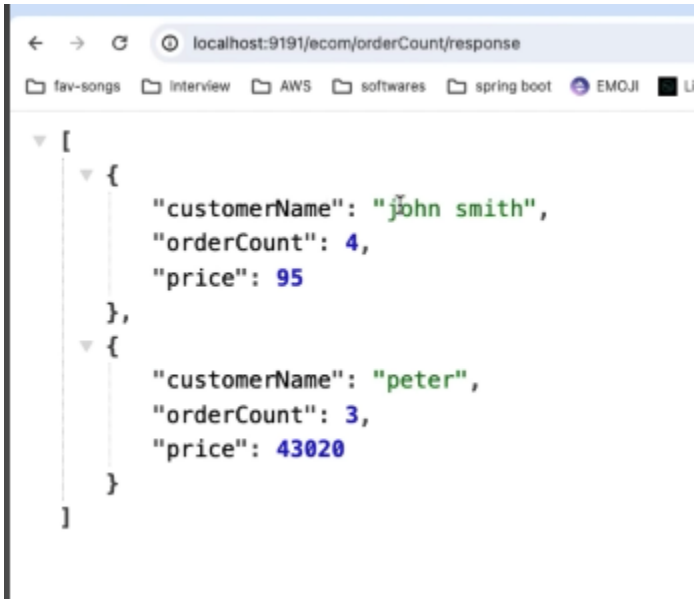
Create DTO to get it as an object



Direct mapping from query to DTO provided below.



```java
@GetMapping("/orderCount/response")
public List<CustomerOrderDTO> getCustomerOrderCountResponse() {
    return orderFulfillmentService.findCustomerOrderCountResponse();
}
}
```



Below is the log with JPQL query

70. How will you implement pagination & Sorting in Spring Data JPA ?

sorting
Pagination
sorting and pagination

```java
public List<Employee> findEmployeesWithSorting(String field){
    return  repository.findAll(Sort.by(Sort.Direction.ASC,field));
}

public Page<Employee> findEmployeesWithPagination(int pageNumber,int pageSize){
    return  repository.findAll(PageRequest.of(pageNumber, pageSize));
    //1000
    //pageSize : 1-100, 101-200 , 201-300
    //offset : 0,1,2,3,4,
    //20
    // pageSize=5 / 10 (0,1)
    //pageNumber : 0 (0-5) , 1 (6-10) , 2 (11-15) , 3 (16-20)
```

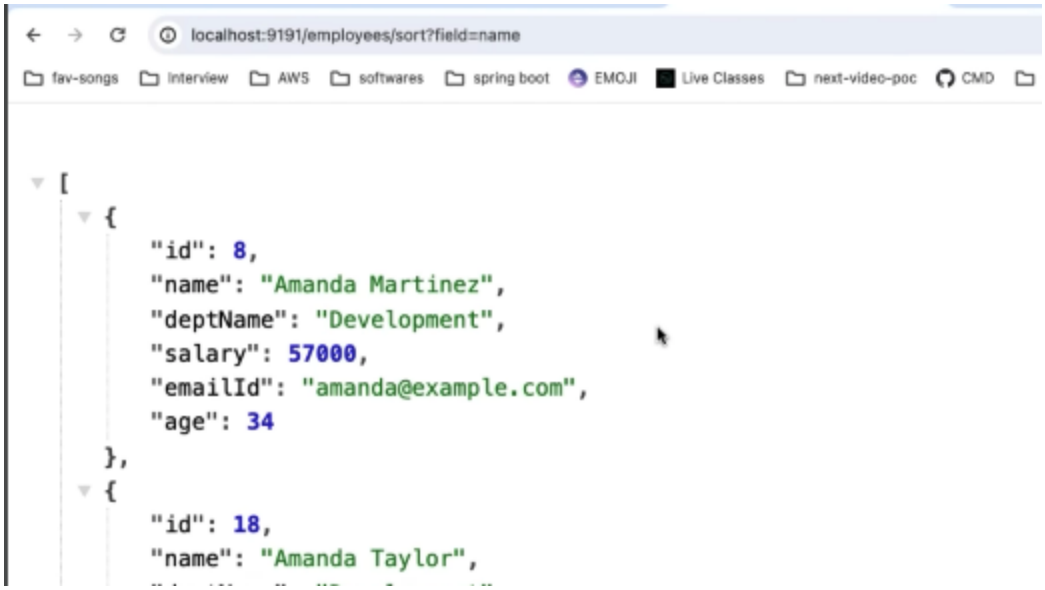
```java
public Page<Employee> findEmployeesWithPaginationAndSorting(int pageNumber, int pageS
    return repository.findAll(PageRequest.of(pageNumber, pageSize).withSort(Sort.by(fi
}
```

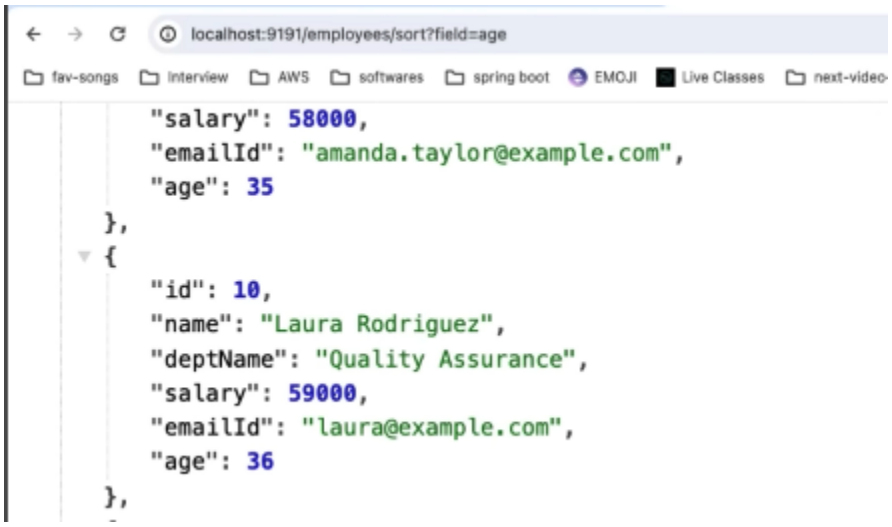
```java
ndSorting(int pageNumber, int pageSize, String field){
umber, pageSize).withSort(Sort.by(field)));
```

Controllers

```java
@GetMapping("/sort")
public List<Employee> findEmployeesWithSorting(@RequestParam String field) {
    return employeeService.findEmployeesWithSorting(field);
}

@GetMapping("/page")
public Page<Employee> findEmployeesWithPagination(@RequestParam int offset, @RequestP
    return employeeService.findEmployeesWithPagination(offset, pageSize);
}
```

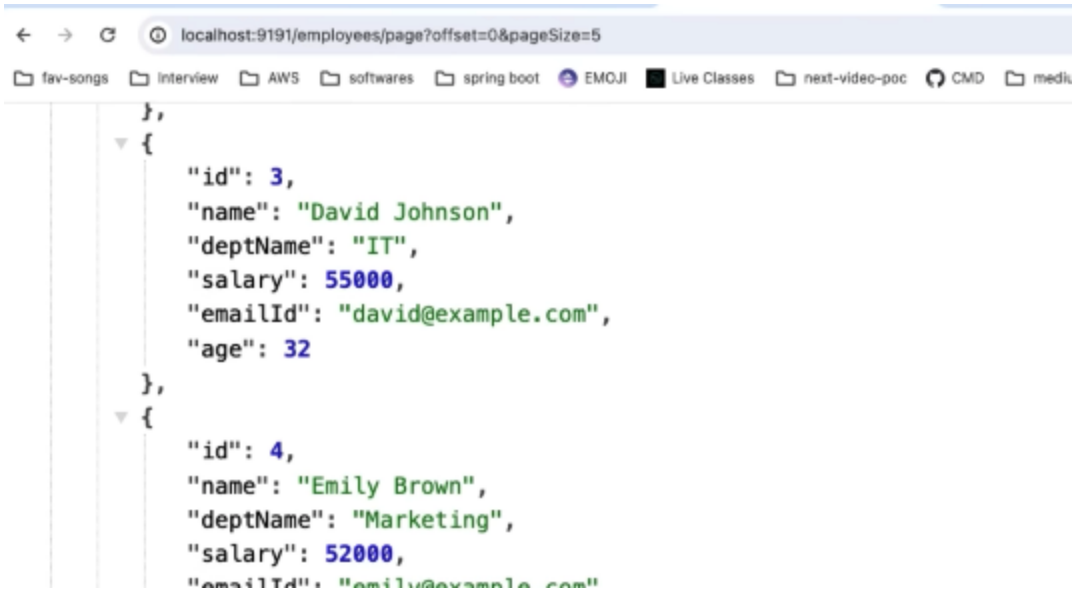
Sortby name
Sortby age
Sort by department
Any sorting we can perform





Pagination

1st page with size 5



Pagination and sorting