**Employee Management System - Defining Query Methods**

This problem describes how the solution implements custom query methods in the **EmployeeRepository** to enhance search functionalities within the Employee Management System.

**Defining Query Methods**

There are two primary approaches to defining custom query methods:

**1. Derived Query Methods:**

Spring Data JPA automatically translates specific method names with patterns into JPQL queries. This method is convenient for basic queries based on entity properties.

**Example:**

* findByName(String name): This method automatically translates to a query that retrieves employees based on their name property.
* findByDepartmentId(Long departmentId): This method retrieves employees based on their department ID.

**2. Custom Queries with @Query Annotation:**

For more complex queries, you can use the @Query annotation on repository methods. This allows defining custom JPQL queries directly within the method.

**Example:**

* findEmployeesByDepartmentName(String departmentName): This method uses JPQL to create a custom query that joins Employee and Department entities and returns a list of EmployeeDTO objects containing employee data and department names.

**Named Queries**

Named queries provide an alternative approach to define reusable queries within the entity class using the @NamedQuery or @NamedQueries annotation.

**1. Define Named Queries:**

Named queries are defined within the entity class (Employee.java in this case) using @NamedQuery here. Each query specifies the JPQL statement and a name for reference.

**Example:**

* Employee.findByEmail: This named query retrieves employees based on their email address.
* Employee.findAllOrderedByName: This named query retrieves all employees ordered by their names in ascending order.

**2. Execute Named Queries:**

One can utilize the named query name and parameters within the repository implementation (EmployeeRepositoryImpl.java) to execute the query using the EntityManager.

**Example:**

* findByEmail(String email): This method retrieves employees by email using the EntityManager to execute the named query Employee.findByEmail with the provided email parameter.

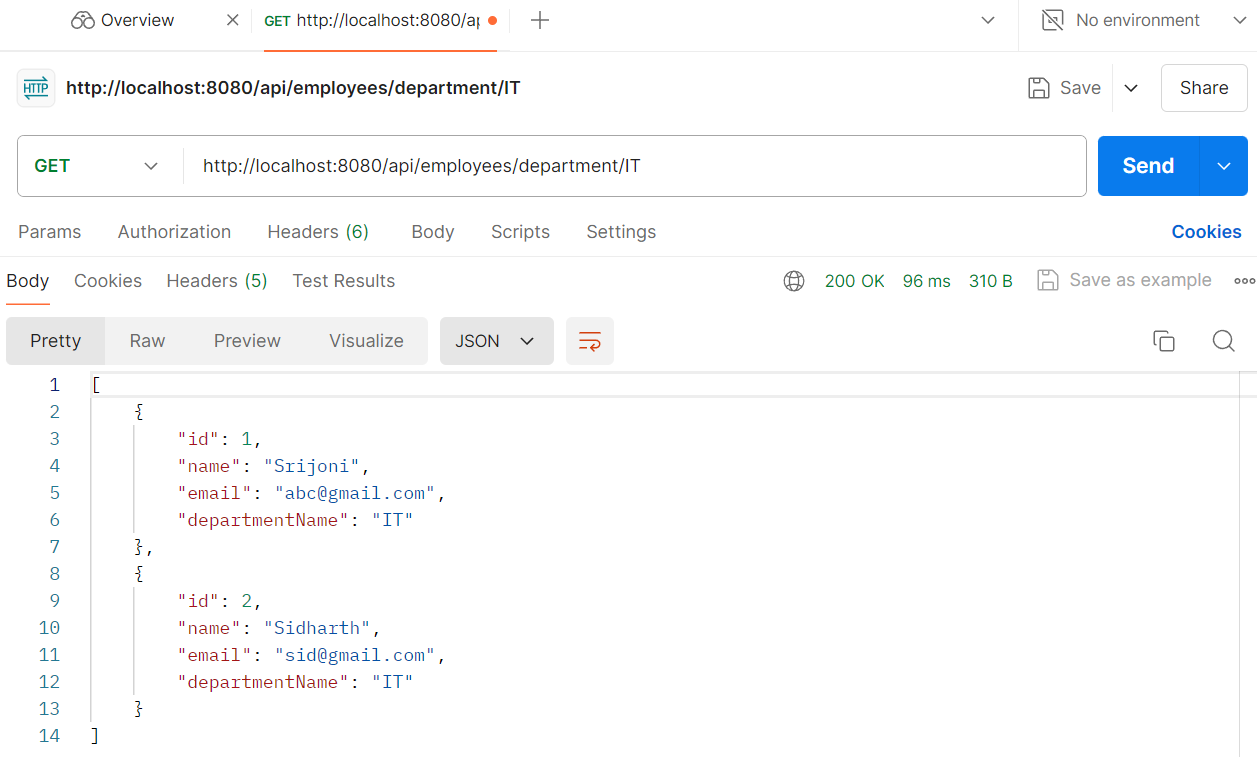
**Summary**

This implementation demonstrates various ways to create custom queries in Spring Data JPA:

* **Derived Query Methods:** Convenient for simple queries based on property names.
* **Custom Queries with @Query:** Allows defining complex JPQL queries directly within methods.
* **Named Queries:** Provides reusable queries defined within the entity class.

These techniques provide flexibility and efficiency for querying data in your Spring Data JPA application.

**GET Employee by Department Name:**



**GET Employee by Email:**

