Derived Queries

Derived Queries in Spring Data JPA — one of its most powerful features that lets you write queries **just by naming methods properly**. No need for **@Query or raw SQL!**

What is a Derived Query?

A **derived query** is a query **automatically generated** by Spring Data based on the **method name** in your repository interface.

You get powerful query behavior without writing any **SQL or JPQL**.

Basic Examples

Assume we have a User entity like:

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQueries;
import jakarta.persistence.NamedQuery;

@Entity
public class User {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	private String username;

	@Column(unique = true)
	private String email;
```

1. Find by Username

```java
Optional<User> findByUsername(String username);
```

2. Find by Email

```java
Optional<User> findByEmail(String email);
```

More Derived Query Examples

3. Find All by Email Containing (like %keyword%)

```java
List<User> findByEmailContaining(String keyword);
```

4. Find All by Username Starting With

```java
List<User> findByUsernameStartingWith(String prefix);
```

5. Find All by Username and Email

```java
List<User> findByUsernameAndEmail(String username, String
email);
```

6. Count Users by Email Domain

```java
Long countByEmailEndingWith(String domain);
```

7. Exists by Email

```java
boolean existsByEmail(String email);
```

8. Delete by Username

```java
void deleteByUsername(String username);
```

Derived Query Keywords

You can mix and match these keywords:

Keyword	Meaning
findBy	Retrieve entity
readBy	Same as findBy
getBy	Same as findBy
existsBy	Check if entity exists
deleteBy	Delete by condition
countBy	Count by condition
...And...	Combine conditions
...Or...	OR condition
...Between	Range query
...LessThan	Less than
...GreaterThan	Greater than
...In	IN clause
...Containing	LIKE %value%
...StartingWith	LIKE value%
...EndingWith	LIKE %value

Best Practices

- **Keep method names readable.**
- **Use Optional<T> when expecting 0 or 1 result.**
- **Use List<T> when expecting multiple results.**
- **Use existsBy for existence checks (fast and clean).**
- **Don't go overboard: if method name gets too long, use @Query.**