Esempi di query derivate e custom

```
package com.example.manuale;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
import java.math.BigDecimal;
import java.util.List;

/* ----- ENTITÀ ----- */
@Data
@NoArgsConstructor
@AllArgsConstructor
class Autore {
    private Long id;
    private String nome;
}

@Data
@NoArgsConstructor
@AllArgsConstructor
class Editrice {
    private Long id;
    private String nome;
}

@Data
@NoArgsConstructor
@AllArgsConstructor
class Categoria {
    private Long id;
    private String descrizione;
```

```
}

@Data
@NoArgsConstructor
@AllArgsConstructor
class Libro {
    private Long id;
    private String titolo;
    private int annoPubblicazione;
    private BigDecimal prezzo;
    private Autore autore;
    private Editrice editrice;
    private List<Categoria> categorie;
}

/* ----- QUERY DERIVATE FINDBY ----- */
// 1. Trova libri con titolo esatto
List<Libro> findByTitolo(String titolo);

// 2. Trova libri con titolo che inizia con il parametro
List<Libro> findByTitoloStartingWith(String prefix);

// 3. Trova libri con titolo che termina con il parametro
List<Libro> findByTitoloEndingWith(String suffix);

// 4. Trova libri il cui titolo contiene il parametro
List<Libro> findByTitoloContaining(String infix);

// 5. Trova libri con titolo incluso in una lista
List<Libro> findByTitoloIn(List<String> titoli);

// 6. Trova libri con prezzo compreso tra due valori
List<Libro> findByPrezzoBetween(BigDecimal minimo, BigDecimal massimo);

// 7. Trova libri con prezzo maggiore o uguale
List<Libro> findByPrezzoGreaterThanEqual(BigDecimal prezzo);
```

```java
// 8. Trova libri con prezzo minore
List<Libro> findByPrezzoLessThan(BigDecimal prezzo);

// 9. Trova libri filtrando per anno e prezzo, ordinati per titolo in modo ascendente
List<Libro> findByAnnoPubblicazioneAndPrezzoBetweenOrderByTitoloAsc(int anno, BigDecimal prezzoMin, BigDecimal prezzoMax);

// 10. Trova libri ordinati per anno (desc) e titolo (asc)
List<Libro> findByTitoloOrderByAnnoPubblicazioneDescTitoloAsc(String titolo);

// 11. Trova libri scritti da un determinato autore (ricerca per oggetto)
List<Libro> findByAutore(Autore autore);

// 12. Trova libri in cui il nome dell'autore è uguale al parametro
List<Libro> findByAutore_Nome(String nome);

// 13. Trova libri pubblicati da una determinata editrice (ricerca per oggetto)
List<Libro> findByEditrice(Editrice editrice);

// 14. Trova libri con nome della editrice che contiene il parametro
List<Libro> findByEditrice_NomeContaining(String nomeEditrice);

// 15. Trova libri che appartengono a una categoria con descrizione specifica
List<Libro> findByCategorie_Descrizione(String descrizioneCategoria);


/* ----- QUERY DERIVATE EXISTSBY (ritornano boolean) ----- */
// 1. Verifica l'esistenza di un libro con titolo esatto
boolean existsByTitolo(String titolo);

// 2. Esistenza di libri con titolo che inizia con il parametro
boolean existsByTitoloStartingWith(String prefix);

// 3. Esistenza di libri con titolo che termina con il parametro
boolean existsByTitoloEndingWith(String suffix);
```

```java
// 4. Esistenza di libri il cui titolo contiene il parametro
boolean existsByTitoloContaining(String infix);

// 5. Esistenza di libri con titolo in una lista
boolean existsByTitoloIn(List<String> titoli);

// 6. Esistenza di libri con prezzo tra due valori
boolean existsByPrezzoBetween(BigDecimal minimo, BigDecimal massimo);

// 7. Esistenza di libri con prezzo maggiore o uguale
boolean existsByPrezzoGreaterThanEqual(BigDecimal prezzo);

// 8. Esistenza di libri con prezzo minore
boolean existsByPrezzoLessThan(BigDecimal prezzo);

// 9. Esistenza di libri scritti da un determinato autore (oggetto)
boolean existsByAutore(Autore autore);

// 10. Esistenza di libri in cui il nome dell'autore corrisponde
boolean existsByAutore_Nome(String nome);

// 11. Esistenza di libri pubblicati da una determinata editrice (oggetto)
boolean existsByEditrice(Editrice editrice);

// 12. Esistenza di libri che contengono una categoria con la descrizione data
boolean existsByCategorie_Descrizione(String descrizioneCategoria);


/* ----- QUERY DERIVATE COUNTBY (ritornano long) ----- */
// 1. Conta i libri con titolo esatto
long countByTitolo(String titolo);

// 2. Conta i libri con titolo che inizia con il parametro
long countByTitoloStartingWith(String prefix);

// 3. Conta i libri con titolo che termina con il parametro
```

```java
long countByTitoloEndingWith(String suffix);

// 4. Conta i libri il cui titolo contiene il parametro
long countByTitoloContaining(String infix);

// 5. Conta i libri con titolo in una lista
long countByTitoloIn(List<String> titoli);

// 6. Conta i libri con prezzo compreso tra due valori
long countByPrezzoBetween(BigDecimal minimo, BigDecimal massimo);

// 7. Conta i libri con prezzo maggiore o uguale
long countByPrezzoGreaterThanEqual(BigDecimal prezzo);

// 8. Conta i libri con prezzo minore
long countByPrezzoLessThan(BigDecimal prezzo);

// 9. Conta i libri di un determinato autore (oggetto)
long countByAutore(Autore autore);

// 10. Conta i libri in cui il nome dell'autore corrisponde
long countByAutore_Nome(String nome);

// 11. Conta i libri pubblicati da una determinata editrice (oggetto)
long countByEditrice(Editrice editrice);

// 12. Conta i libri che contengono nella lista di categorie una descrizione specifica
long countByCategorie_Descrizione(String descrizioneCategoria);


/* ----- QUERY DERIVATE DELETEBY ----- */
// 1. Elimina libri con titolo esatto
void deleteByTitolo(String titolo);

// 2. Elimina libri con titolo che inizia con il parametro
void deleteByTitoloStartingWith(String prefix);
```

```java
// 3. Elimina libri con titolo che termina con il parametro
void deleteByTitoloEndingWith(String suffix);

// 4. Elimina libri il cui titolo contiene il parametro
void deleteByTitoloContaining(String infix);

// 5. Elimina libri con titolo in una lista
void deleteByTitoloIn(List<String> titoli);

// 6. Elimina libri con prezzo compreso tra due valori
void deleteByPrezzoBetween(BigDecimal minimo, BigDecimal massimo);

// 7. Elimina libri con prezzo maggiore o uguale
void deleteByPrezzoGreaterThanEqual(BigDecimal prezzo);

// 8. Elimina libri con prezzo minore
void deleteByPrezzoLessThan(BigDecimal prezzo);

// 9. Elimina libri scritti da un determinato autore (oggetto)
void deleteByAutore(Autore autore);

// 10. Elimina libri in cui il nome dell'autore corrisponde
void deleteByAutore_Nome(String nome);

// 11. Elimina libri appartenenti a una categoria con descrizione specifica
void deleteByCategorie_Descrizione(String descrizioneCategoria);


/* ----- QUERY CUSTOM (con @Query e parametri posizionali, senza usare @Param) ----- */
import org.springframework.data.jpa.repository.Query;

// 1. Trova libri con titolo esatto
@Query("select l from Libro l where l.titolo = ?1")
List<Libro> findCustomByTitolo(String titolo);
```

```java
// 2. Trova libri con titolo che contiene il parametro usando LIKE
@Query("select l from Libro l where l.titolo like %?1%")
List<Libro> findCustomByTitoloContaining(String infix);

// 3. Trova libri pubblicati tra due anni
@Query("select l from Libro l where l.annoPubblicazione between ?1 and ?2")
List<Libro> findCustomByAnnoPubblicazioneBetween(int startYear, int endYear);

// 4. Trova libri con prezzo maggiore di un valore, ordinati per titolo (asc)
@Query("select l from Libro l where l.prezzo > ?1 order by l.titolo asc")
List<Libro> findCustomByPrezzoGreaterThanOrderByTitoloAsc(BigDecimal prezzo);

// 5. Trova libri scritti da un autore con nome specifico
@Query("select l from Libro l where l.autore.nome = ?1")
List<Libro> findCustomByAutoreNome(String nomeAutore);

// 6. Trova libri pubblicati da una editrice con nome specifico
@Query("select l from Libro l where l.editrice.nome = ?1")
List<Libro> findCustomByEditriceNome(String nomeEditrice);

// 7. Trova libri che hanno una categoria con descrizione specifica
@Query("select l from Libro l join l.categorie c where c.descrizione = ?1")
List<Libro> findCustomByCategoriaDescrizione(String descrizioneCategoria);


// 8. Trova libri con titolo in una lista di titoli
@Query("select l from Libro l where l.titolo in ?1")
List<Libro> findCustomByTitoloIn(List<String> titoli);

// 9. Trova libri filtrando per nome autore e nome editrice, ordinati per anno (desc) e titolo (asc)
@Query("select l from Libro l where l.autore.nome = ?1 and l.editrice.nome = ?2 order by l.annoPubblicazione desc, l.titolo as
List<Libro> findCustomByAutoreNomeAndEditriceNomeOrderByAnnoDescTitoloAsc(String nomeAutore, String nomeEditrice);


/* ----- QUERY SU CAMPI DI UN OGGETTO IN RELAZIONE ----- */
// 1. Trova libri in base al nome dell'autore esatto
```

```
List<Libro> findByAutore_Nome(String nome);

// 2. Trova libri in cui il nome dell'autore inizia con il parametro
List<Libro> findByAutore_NomeStartingWith(String prefix);

// 3. Trova libri in cui il nome dell'autore termina con il parametro
List<Libro> findByAutore_NomeEndingWith(String suffix);

// 4. Trova libri in cui il nome dell'autore contiene il parametro
List<Libro> findByAutore_NomeContaining(String infix);

// 5. Trova libri in cui il nome dell'autore è in una lista di nomi
List<Libro> findByAutore_NomeIn(List<String> nomi);

// 6. Trova libri in cui il nome dell'autore non corrisponde al parametro
List<Libro> findByAutore_NomeNot(String nome);

// 7. Trova libri in cui il nome dell'autore rispetta un pattern (LIKE)
List<Libro> findByAutore_NomeLike(String pattern);

// 8. Trova libri in base al nome della editrice esatto
List<Libro> findByEditrice_Nome(String nome);

// 9. Trova libri in cui il nome della editrice contiene il parametro
List<Libro> findByEditrice_NomeContaining(String infix);

// 10. Trova libri in cui il nome della editrice inizia con il parametro
List<Libro> findByEditrice_NomeStartingWith(String prefix);
```