JPA- Queries 2 Övningar

Översikt

I denna övning kommer vi att arbeta med mer avancerade frågor i JPA. Vi kommer att:

- Navigera genom relationer.
- Arbeta med samlingar och member of.
- Använda join för att hantera many-relationer.
- Kedja metoder för att effektivisera kod.

Steg 1: Navigera genom relationer

- Skapa en fråga för att hämta alla studenter som tillhör en specifik tutor.
- Hämta resultatet och lagra det i en lista av Student-objekt.
- Loopa igenom listan och skriv ut varje student.
- Kör koden och verifiera att du får rätt resultat.

Tips:

- Du har en @OneToMany-relation mellan Tutor och Student, vilket gör att Tutor-klassen innehåller all information om studenterna.
- Skapa en fråga som hämtar alla studenter för en given tutor baserat på tutor-namnet.

Steg 2: Arbeta med samlingar - member of

- Skapa en fråga för att hitta vilken Tutor som undervisar ett specifikt ämne.
- Hämta ämnet från databasen genom att använda dess ID.
- Skapa en fråga som använder member of för att hitta alla tutor som undervisar i detta ämne.
- Hämta resultatet och skriv ut namnen på tutorerna.
- Kör koden och verifiera att du får rätt resultat.

Tips:

- “member of” kan användas när du arbetar med samlingar i JPA.
- Sätt en parameter i frågan för att definiera vilket ämne du vill hämta tutorer för.
- Kontrollera att du har ämnen kopplade till tutorer innan du kör frågan.

Steg 3: Använda join

- Skapa en fråga för att hämta alla tutor som har minst en student som bor i en viss stad.
- Använd join för att koppla samman Tutor och Student via teachingGroup.
- Filtrera frågan så att den endast returnerar tutorer där minst en student bor i en specifik stad.
- Hämta resultatet och skriv ut namnen på tutorerna.
- Kör koden och verifiera att du får rätt resultat.

Tips:

- join används när du har many-relationer och behöver ansluta tabeller.
- Kontrollera att minst en student har en bostadsadress i den angivna staden innan du kör frågan.

Steg 4: Eliminera duplicering med distinct

- Ändra den tidigare join-frågan så att varje tutor endast returneras en gång.
- Använd “distinc”t i frågan för att ta bort duplicerade resultat.
- Hämta resultatet och skriv ut tutorerna.
- Kör koden och verifiera att varje tutor endast returneras en gång.

Tips:

- Utan “distinct” kan en tutor dyka upp flera gånger om den har flera studenter i samma stad.
- “distinct” gör att varje tutor endast returneras en gång, oavsett hur många studenter den har i staden.

Steg 5: Kedja metoder för effektiv kod

- Skapa en fråga som hämtar alla tutor i en viss stad.
- Använd parameterisering så att staden kan varieras utan att ändra frågan.
- Sätt parametern direkt i frågan genom att kedja metoder.
- Hämta resultatet och skriv ut tutorerna.
- Kör koden och verifiera att du får rätt resultat.

Tips:

- Använd .setParameter() direkt i frågan.
- getResultList() kan kallas direkt i kedjan utan att skapa en separ