# Native querys - Spring Data JPA

Usar **recursos prontos do SDJ** como *named queries* e *métodos comuns já implementados* é muito útil, mas tem momentos onde **é necessário que uma query seja escrita por nós mesmos**. Seja porque a *named querie é gigantesca e não vale a pena mantê-la, seja para garantir uma melhor performance, legibilidade e segurança e etc...*

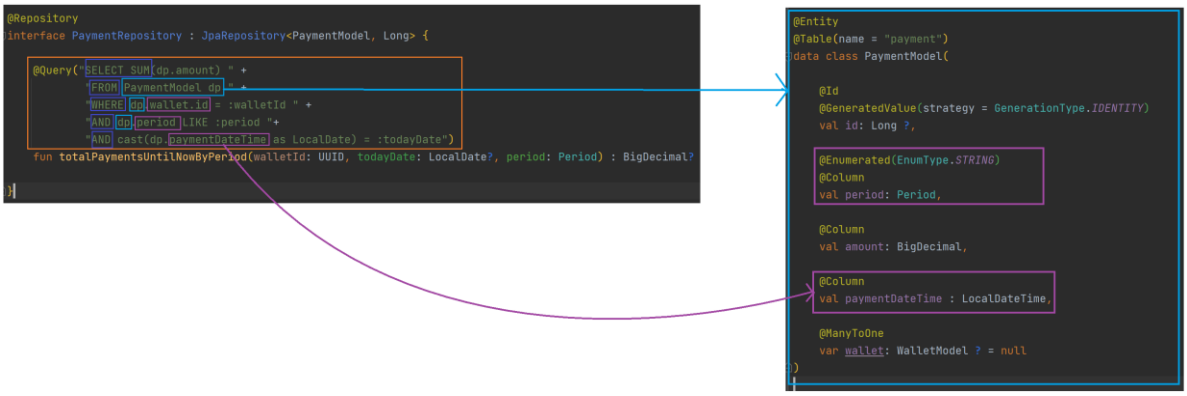
Sempre aparece motivos para que essa situação aconteça, e é aí que as **native querys** ajudam a gente. É muito simples implementar um comportamento **nativo de sql ou JPQL** em um método do nosso repositório, basta usar a anotação **@Query** e descrever o comportamento, quando o método for chamado a JPQL será executada:

```
@Repository
public interface GoalRepository extends JpaRepository<Goal, Long > {

    Page<Goal> findAllByUserId(Long userId, Pageable pageable);

    @Query("FROM Goal g WHERE g.dateHistory.expectedFinalizationDate BETWEEN :startOfTheDay AND :endOfTheDay " +
            "AND g.status <> 'COMPLETED'" )
    List<Goal> findGoalsInDeadLine(LocalDateTime startOfTheDay, LocalDateTime endOfTheDay);
}
```

**JPQL** é **um jeito de escrever querys de sql com uma sintaxe mais voltada pra orientação a objetos**, então basta usar o **nome das Entidades do projeto** para se referenciar as entidades do banco de dados, **navegar entre seus atributos(colunas)** também é mais simples, enquanto **palavras chaves se mantém quase que inalteradas** nas querys nativas e em jpql:




JPQLS garantem um pouco mais de flexibilidade na query **(como nessa de cima onde eu pude fazer um cast de um LocalDateTime para LocalDate)**, uma legibilidade maior em relação a named queries em consultas que são mais complexas.