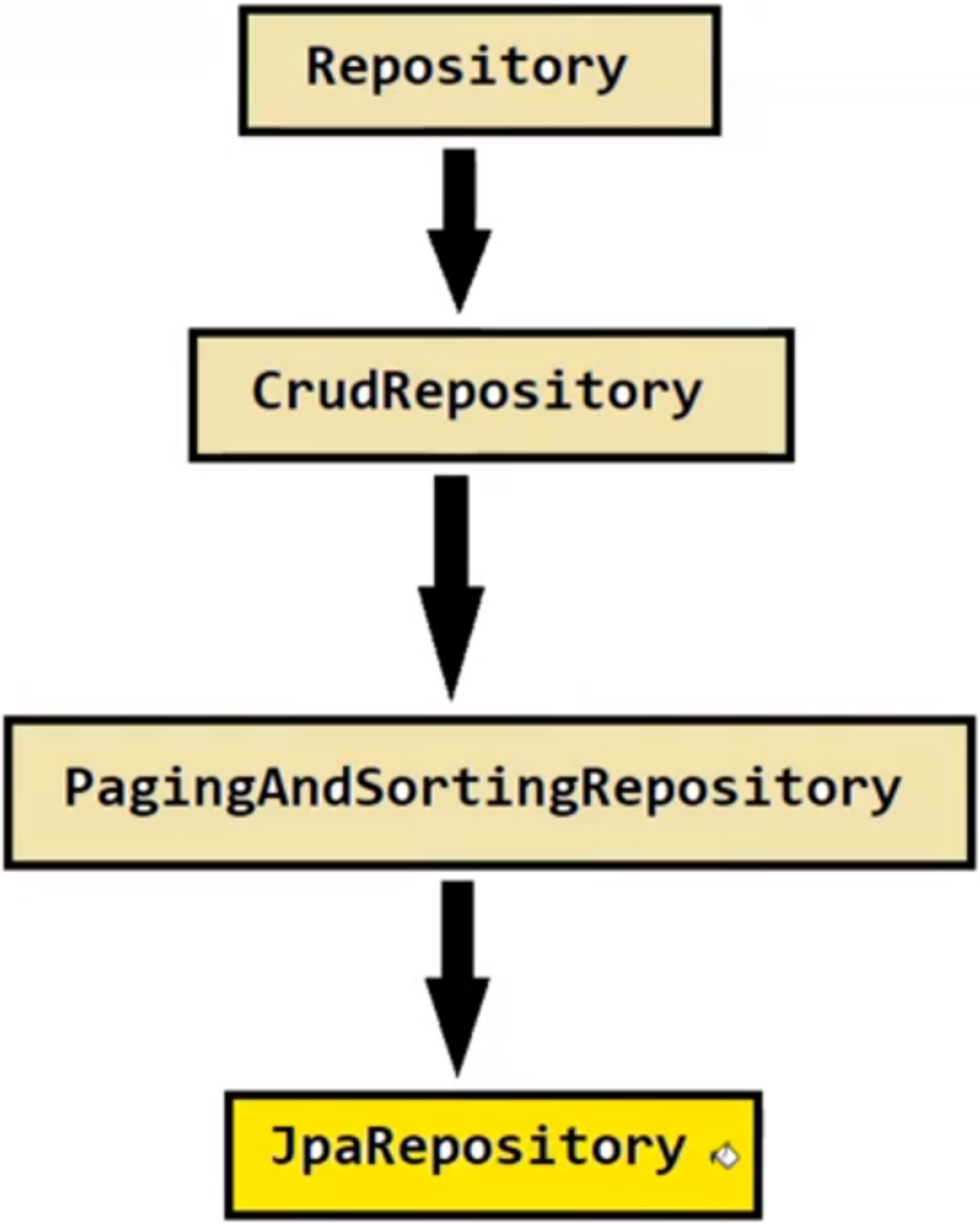
=> Spring Data Access :-
-> Spring Data Access is the part of spring framework which aims to simplifies data access in spring applications by offering abstractions and tools for relational databases, NoSQL databases etc
-> Modules in Spring Data Access are :-
1. Spring Data JPA
= Focus on relational database access using JPA
2. Spring Data MongoDB
= Focus on NoSQL access using MongoDB
3. Spring Data Cassandra
= Focus on working with Apache Cassandra i.e. NoSQL database
4. Spring Data Redis
= Focus on caching and key-valye data storage in Redis
5. Spring Data Neo4j
= Focus on graph database access using Neo4j
etc


Repository
CrudRepository
PagingAndSortingRepository
JpaRepository

=> Spring Data JPA :-
-> Spring Data JPA is the part of larger Spring Data Project which provides a simplified and consistent way to work with JPA in spring-based application

-> Spring Data JPA provides some "Repository Interfaces" which provides a convenient way to perform common database operations without writing the actual database queries or boilerplate code
-> Some Repository Interfaces present in Spring Data JPA are :-
1. Repository
2. CrudRepository
3. PagingAndSortingRepository
4. JpaRepository
etc
-> Hierarchy of Repository Interfaces :-
= diagram

-> All Spring Data JPA Repository Interfaces provides 2 types of methods :-
1. Core CRUD Operation Methods
= Used for basic data manipulation
2. Query Methods
= Used to define custom queries using method naming conventions or custom JPQL/SQL queries

==========

```
=> Core CRUD Operation Methods in Spring Data JPA Repository Interfaces are :-
    -> Insert (Create) Operations :-
        = save(S entity)
        = saveAll(Iterable<S> entity)
    -> Update Operations :-
        = save(S entity)
    -> Delete Operations :-
        = delete(S entity)
        = deleteAll(Iterable<? extends S> entities)
        = deleteAll()
        = deleteById(Id id)
    -> Read (Retrieve) Operations :-
        = findById(ID id)
        = findAll()
        = findAllById(Iterable<ID> ids)
        = count()
        = existsById(ID id)
```

```java
public void updateStdDetailsService(int id, float marks)
{
    try
    {
        Optional<Student> optional = studentRepository.findById(id);
        Student std = optional.get();
        std.setMarks(marks);

        studentRepository.save(std);
        System.out.println("Updation success");
    }
    catch(Exception e)
    {
        System.out.println("Updation failed");
        e.printStackTrace();
    }
```

```
stdService.updateStdDetailsService(4, 96.34f);
```

```java
public void deleteStdDetailsService(int id)
{
    try
    {
        studentRepository.deleteById(id);
        System.out.println("Deletion success");
    }
    catch(Exception e)
    {
        System.out.println("Deletion failed");
        e.printStackTrace();
    }
}
```

```
stdService.deleteStdDetailsService(4);
```

```
public Student getStdDetailsService(int id)
{
    Optional<Student> optional = studentRepository.findById(id);
    Student std = optional.get();
    return std;
}
```

```
Student std = stdService.getStdDetailsService(2);
System.out.println("Name : "+std.getName());
System.out.println("Rollno : "+std.getr());
System.out.println("Marks : "+std.getName());
```

```
public List<Student> getAllStdDetailsService()
{
    return studentRepository.findAll();
}
```

```
List<Student> std_list = stdService.getAllStdDetailsService();
for(Student std : std_list)
{
  System.out.println("Name : "+std.getName());
  System.out.println("Rollno : "+std.getRollno());
  System.out.println("Marks : "+std.getMarks());
  System.out.println("--------------------------------");
}
```

*C:\Users\Deepesh Panwar\Downloads\Notes - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Spring with JPA.txt | Spring Data JPA Introduction.txt | Spring Data JPA Program - 1.txt | new 1 | Notes

```
==>> Syntax for Query Methods Using Method Naming Conventions <<==

1. findBy{Property}(Type property)
    = For example : findByEmail(String email)
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_email = ?

2. findBy{Property}IgnoreCase(Type property)
    = For example : findByEmailIgnoreCase(String email):
    = Generated SQL Query: SELECT * FROM student_marks WHERE LOWER(std_email) = LOWER(?)

3. findBy{Property1}And{Property2}(Type property1, Type property2)
    = For example : findByNameAndEmail(String name, String email):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name = ? AND std_email = ?

4. findBy{Property1}Or{Property2}(Type property1, Type property2)
    = For example : findByNameOrEmail(String name, String email):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name = ? OR std_email = ?

5. findBy{Property}GreaterThan(Type property)
    = For example : findByMarksGreaterThan(float marks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks > ?

6. findBy{Property}LessThan(Type property)
    = For example : findByMarksLessThan(float marks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks < ?

7. findBy{Property}GreaterThanEqual(Type property)
    = For example : findByMarksGreaterThanEqual(float marks):
```

Normal text file | length : 7,040 lines : 153 | Ln : 6 Col : 1 Pos : 224 | Windows (CR LF) | UTF-8 | INS

ENG

```
8. findBy{Property}LessThanEqual(Type property)
    = For example : findByMarksLessThanEqual(float marks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks <= ?

9. findBy{Property}Between(Type property)
    = For example : findByMarksBetween(float minMarks, float maxMarks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks BETWEEN ? AND ?

10. findBy{Property}Not(Type property)
    = For example : findByMarksNot(float marks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks <> ?

11. findBy{Property}Like(Type property)
    = For example : findByNameLike(String name):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name LIKE ?

12. findBy{Property}StartingWith(Type property)
    = For example : findByNameStartingWith(String prefix):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name LIKE ?

13. findBy{Property}EndingWith(Type property)
    = For example : findByNameEndingWith(String suffix):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name LIKE ?

14. findBy{Property}Containing(Type property)
    = For example : findByNameContaining(String keyword):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name LIKE ?
```

*C:\Users\Deepesh Panwar\Downloads\Notes - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

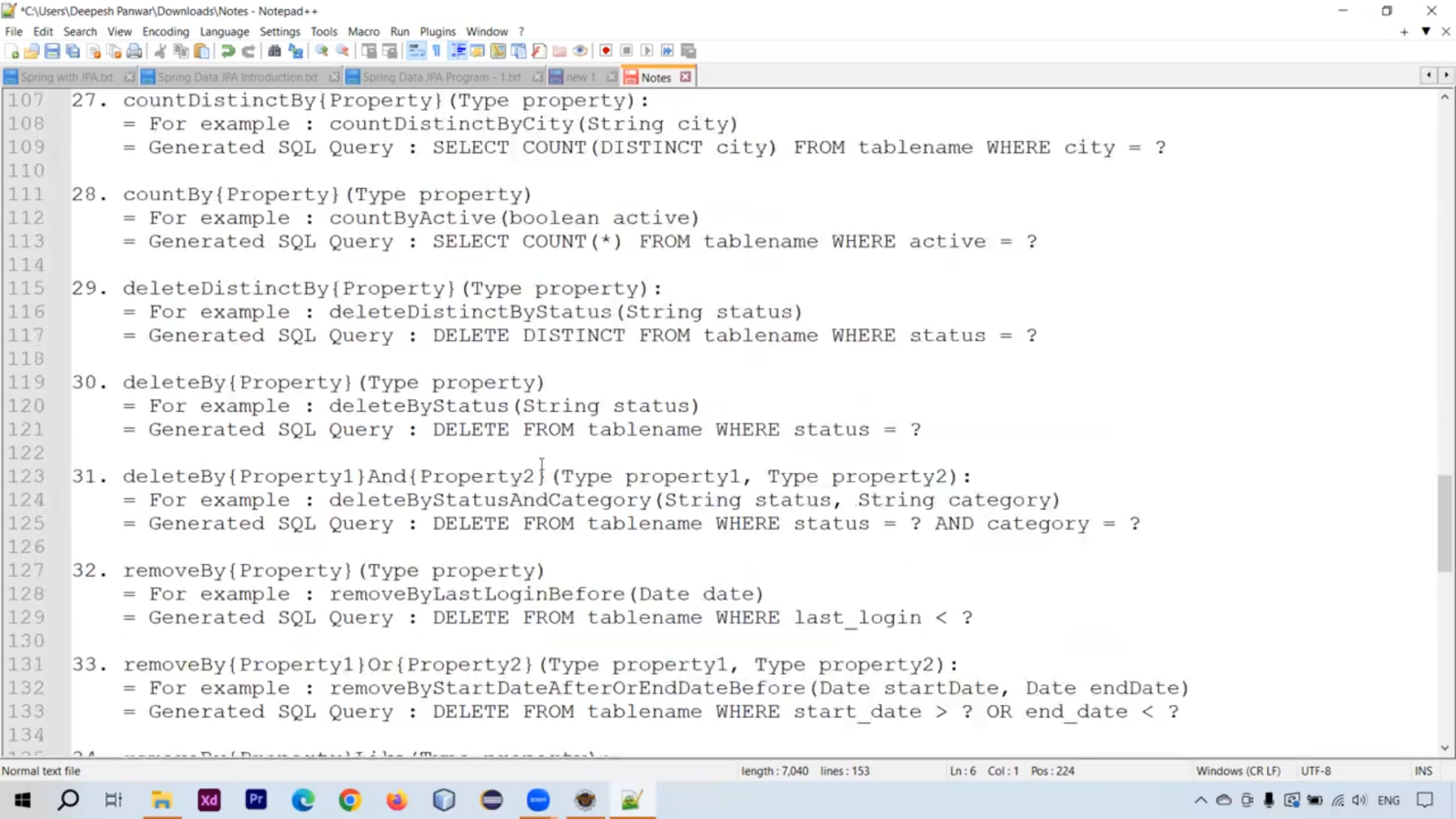
Spring with JPA.txt | Spring Data JPA Introduction.txt | Spring Data JPA Program - 1.txt | new 1 | Notes

```
14. findBy{Property}Containing(Type property)
    = For example : findByNameContaining(String keyword):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_name LIKE ?

15. findFirstByOrderBy{Property}Asc()
    = For example : findFirstByOrderByMarksAsc():
    = Generated SQL Query: SELECT * FROM student_marks ORDER BY std_marks ASC LIMIT 1

16. findFirstByOrderBy{Property}Desc()
    = For example : findFirstByOrderByMarksDesc():
    = Generated SQL Query: SELECT * FROM student_marks ORDER BY std_marks DESC LIMIT 1

17. findTop{N}ByOrderBy{Property}Desc(Type property)
    = For example : findTop5ByOrderByMarksDesc():
    = Generated SQL Query: SELECT * FROM student_marks ORDER BY std_marks DESC LIMIT 5

18. findTop{N}ByOrderBy{Property}Desc(Type property)
    = For example : findTop10ByOrderByMarksDesc():
    = Generated SQL Query: SELECT * FROM student_marks ORDER BY std_marks DESC LIMIT 10

19. findBy{Property}In(List<Type> property)
    = For example : findByEmailIn(List<String> emails):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_email IN (?)

20. findBy{Property}IsNull()
    = For example : findByEmailIsNull():
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_email IS NULL
```

Normal text file length : 7,040 lines : 153 Ln : 6 Col : 1 Pos : 224 Windows (CR LF) UTF-8 INS

ENG

```
20. findBy{Property}IsNull()
    = For example : findByEmailIsNull():
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_email IS NULL

21. findBy{Property}IsNotNull()
    = For example : findByEmailIsNotNull():
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_email IS NOT NULL

22. findBy{Property1}BetweenOrderBy{Property2}Asc(Type property1, Type property2)
    = For example : findByMarksBetweenOrderByMarksAsc(float minMarks, float maxMarks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks BETWEEN ? AND ? ORDER BY
    std_marks ASC

23. findBy{Property1}BetweenOrderBy{Property2}Desc(Type property1, Type property2)
    = For example : findByMarksBetweenOrderByMarksDesc(float minMarks, float maxMarks):
    = Generated SQL Query: SELECT * FROM student_marks WHERE std_marks BETWEEN ? AND ? ORDER BY
    std_marks DESC

24. findDistinctBy{Property}(Type property)
    = For example : findDistinctByName(String name):
    = Generated SQL Query: SELECT DISTINCT * FROM student_marks WHERE std_name = ?

25. find{EntityName}By{Property}(...)
    = For example : findStudentByAgeGreaterThan(int age)
    = Generates a custom query based on the method name.

26. queryBy{Property}(...)
    = For example : queryByAgeGreaterThan(int age)
```


*C:\Users\Deepesh Panwar\Downloads\Notes - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Spring with JPA.txt
Spring Data JPA Introduction.txt
Spring Data JPA Program - 1.txt
new 1
Notes
27. countDistinctBy{Property}(Type property):
= For example : countDistinctByCity(String city)
= Generated SQL Query : SELECT COUNT(DISTINCT city) FROM tablename WHERE city = ?
28. countBy{Property}(Type property)
= For example : countByActive(boolean active)
= Generated SQL Query : SELECT COUNT(*) FROM tablename WHERE active = ?
29. deleteDistinctBy{Property}(Type property):
= For example : deleteDistinctByStatus(String status)
= Generated SQL Query : DELETE DISTINCT FROM tablename WHERE status = ?
30. deleteBy{Property}(Type property)
= For example : deleteByStatus(String status)
= Generated SQL Query : DELETE FROM tablename WHERE status = ?
31. deleteBy{Property1}And{Property2}(Type property1, Type property2):
= For example : deleteByStatusAndCategory(String status, String category)
= Generated SQL Query : DELETE FROM tablename WHERE status = ? AND category = ?
32. removeBy{Property}(Type property)
= For example : removeByLastLoginBefore(Date date)
= Generated SQL Query : DELETE FROM tablename WHERE last_login < ?
33. removeBy{Property1}Or{Property2}(Type property1, Type property2):
= For example : removeByStartDateAfterOrEndDateBefore(Date startDate, Date endDate)
= Generated SQL Query : DELETE FROM tablename WHERE start_date > ? OR end_date < ?
Normal text file
length : 7,040 lines : 153
Ln : 6 Col : 1 Pos : 224
Windows (CR LF) UTF-8 INS
ENG

=> Query Methods :-

-> Query methods are the methods which are used to define database queries by simply declaring method signatures in our repository interfaces. These methods are used to automatically generate SQL queires based on the method names and parameters, reducing the need for writing custom SQL queries.

-> Query methods are defined in repository interfaces

-> Query methods are used to define custom queries using :-

= method naming conventions

= custom JPQL queries

= custom SQL queires

-> For eg :-

= Query Method : findByEmail(String email)

= Generated SQL query : SELECT * FROM table_name WHERE email=?;

==========

```java
@Query("SELECT s FROM Student s WHERE s.email = :stdEmail")
public Student getStdDetailsByEmail(@Param("stdEmail") String email);

@Query("SELECT s FROM Student s WHERE s.marks > :stdMarks")
public List<Student> getByMarksGreaterThan(@Param("stdMarks") float marks);
```