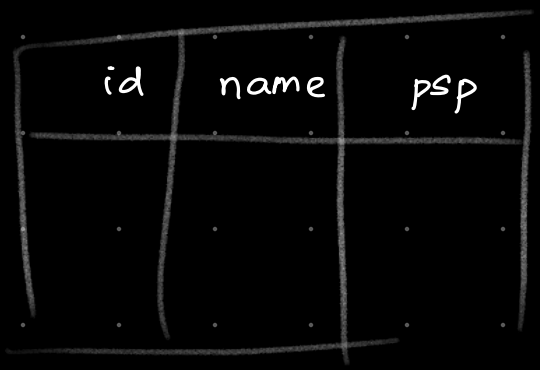
Understanding ORM

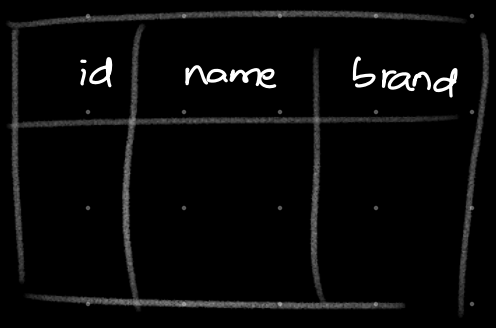
```
@Entity
class Student
    @Id
    long id
    String name
    String psp
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
```

Tables created -_-



Student



laptop.

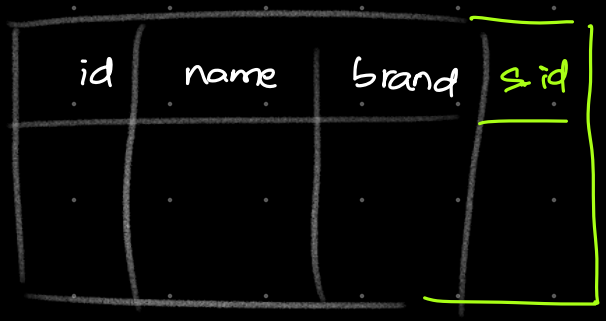
Add Rel^:

1 Student can have 1 laptop.

1:1.

How tables should be created

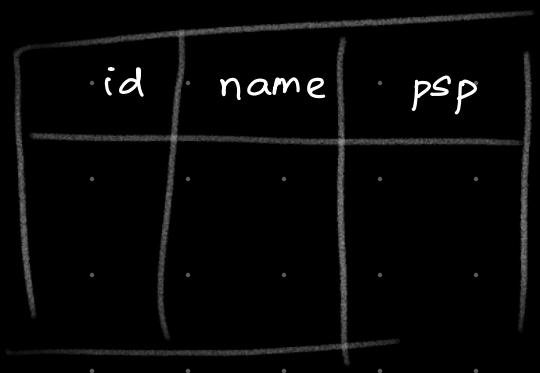
Student



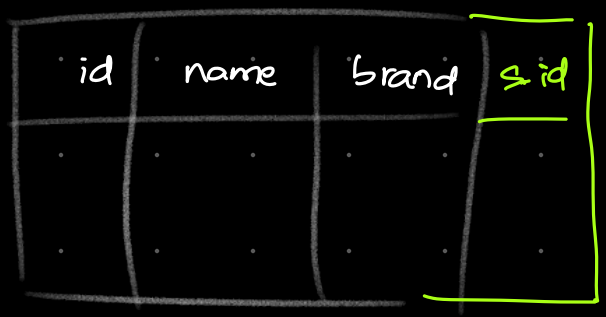
laptop.

```
@Entity
class Student
    @Id
    long id
    String name
    String psp
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
    @OneToOne
    Student student;
```



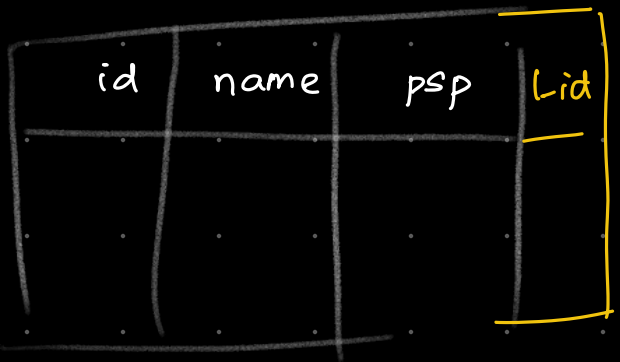
Student

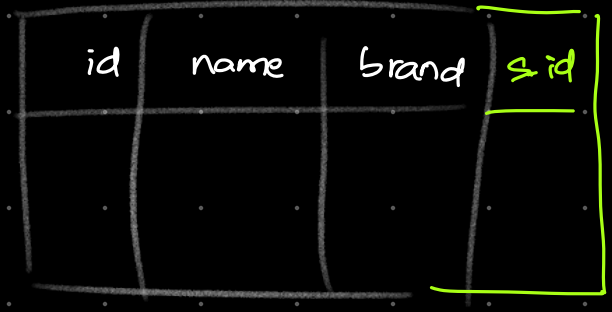


laptop.

```
@Entity
class Student
    @Id
    long id
    String name
    String psp
    @OneToOne
    laptop laptop
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
    @OneToOne
    Student student;
```





Student

laptop.

Fix

```
@Entity
class Student
    @Id
    long id
    String name
    String psp
    @OneToOne(mappedBy = "student")
    laptop laptop
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
    @OneToOne
    Student student;
```

id	name	psp

Student

id	name	brand	s_id

laptop.

One student can have multiple laptops.. [1:M].

tables you want to create.

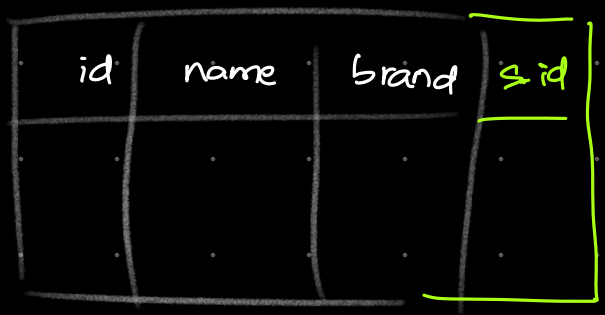
Put id of student in laptop table.

The code that we've...

```
@Entity
class Student
    @Id
    long id
    String name
    String psp
    @OneToMany
    List<laptop> laptop
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
    @Manytoone
    Student student;
```

Student



laptop.

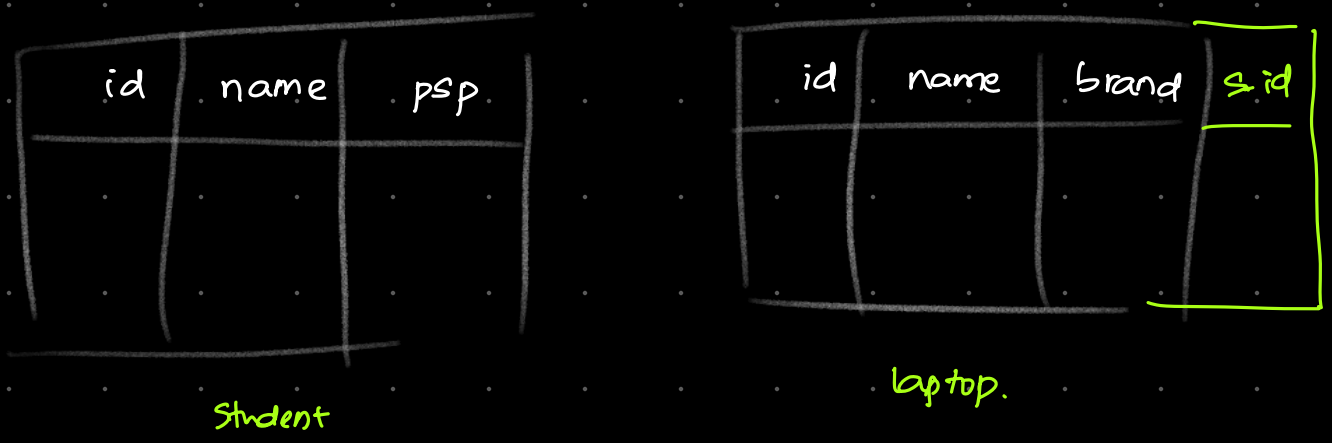
s-id	l-id
1	1
1	2
1	3

Student-laptop

Try

```
@Entity
class Student
    @Id
    long id
    String name
    String psp
    @OneToMany(mappedBy="student")
    List<laptop> laptop
```

```
@Entity
class laptop
    @Id
    long id
    String brand
    String name
    @Manytoone
    Student student;
```



Student

laptop.

// Try doing M:M and verify & write the corresponding annotations.

Optional

```
User user = UserUtil.findUser(userId)
if(user!=null)
    user.name → print
```

```
Optional<User> user = ____
user → Optional.
```


```
user.isPresent() → boolean
user.get() // get actual data
```

Student

id	name	psp
1	Keerthi	90

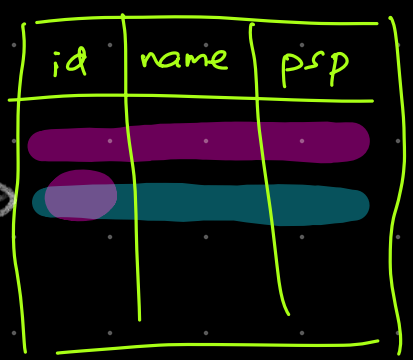
1. establish db connection
2. write query
3. execute query
4. map to your objects
5. get the data

Repositories

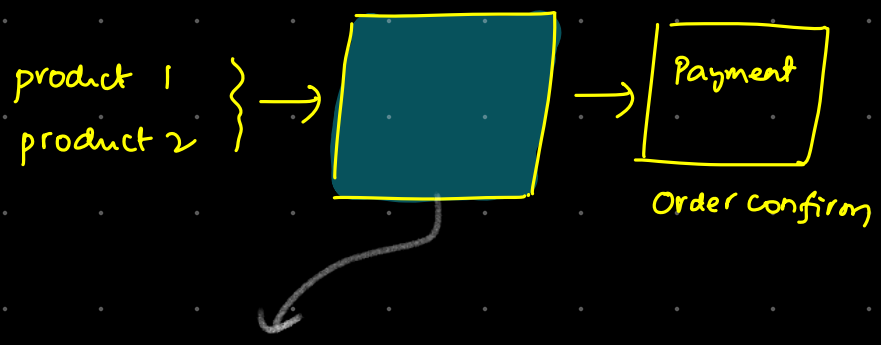
```
interface StudentRepository extends JpaRepository<Student, Long>
```

StudentController

```
StudentRepository. ____ .save(Student)
                   ____ .findById(123)
                   ____ .count()
                   ____ .findAll()
                   ____
```



Amazon.in



Add address / confirm products / more details.