PART 1- RDBMS Concepts (Conceptual)

- What is RDBMS?
- Table, Row, Column
- Primary Key
- Foreign Key
- Relationship between tables
- Use simple student + course example

PART 2 - Referential Integrity (Very Important)

Explain using:

- Parent table
- Child table
- What happens if parent record is deleted
- Why DB throws error

Then connect to:

- @ManyToOne
- @JoinColumn
- cascade

PART 3 – Normalization (Core Focus)

Teach step-by-step:

- Unnormalized form
- 1NF
- 2NF
- 3NF
- Stop at 3NF — perfect for 2nd year.

Use:

- Student table
- Course table
- Address table

PART 4 – Mapping Normalized Tables to JPA

Show:

Normalized tables → Entities

Relationships → Annotations

CRUD still works the same

This is where Spring Boot makes sense.

Part 1:

- Table → Stores data
- Row → One record
- Column → One property
- Primary Key → Unique identity
- Foreign Key → Connection between tables
- Relationships → Real-world connections

Part 2:

Referential Integrity ensures that relationships between tables remain valid and meaningful.

In simple words:

- A child table must always refer to a valid parent table record
- Invalid references are not allowed

Referential Actions - These define what happens when parent data changes.

1.RESTRICT / NO ACTION (Default)

- Parent cannot be deleted if child exists.

Example:

- Student has marks
- Try to delete student ⍰

Result:

- Database throws error
- Safe but strict

2. CASCADE

- Parent change affects child automatically.

Example:

- Delete student
- All marks deleted automatically
- Clean delete
- Used carefully

3. SET NULL

- Child foreign key becomes NULL.

Example:

- Course deleted
- Student.course_id becomes NULL
- Child exists, relation removed

4. SET DEFAULT

- Child foreign key set to default value.
- (Used rarely)

Simple Comparison Table (Student Friendly)

Action	**What Happens**
RESTRICT	Prevent delete
CASCADE	Auto delete child
SET NULL	Break relation
SET DEFAULT	Assign default

There are 5 cascade types in JPA:

- CascadeType.PERSIST
- CascadeType.MERGE
- CascadeType.REMOVE
- CascadeType.REFRESH
- CascadeType.DETACH

And one shortcut:

- CascadeType.ALL

Explain Each CASCADE Type (CLEAR + REAL TIME)

1. CascadeType.PERSIST

- When parent is saved, child is saved automatically.

Example

- Save Course → Students auto saved
- @OneToMany(cascade = CascadeType.PERSIST)
- private List<Student> students;

Use when:

- Creating parent + child together

2. CascadeType.MERGE

- When parent is updated, child is updated automatically.
- @OneToMany(cascade = CascadeType.MERGE)
- private List<Student> students;

Use when:

- Updating parent & child together

3. CascadeType.REMOVE

- When parent is deleted, child is deleted automatically.
- @OneToMany(cascade = CascadeType.REMOVE)
- private List<Student> students;

Real-life:

- Delete Course → All Students removed
- Dangerous if misused

4. CascadeType.REFRESH

- Refresh parent → child refreshed from DB
- Rarely used in basic projects.

5. CascadeType.DETACH

- Detach parent → child detached from persistence context
- Advanced concept (you can just mention).

What is CascadeType.ALL?

cascade = CascadeType.ALL

Means:

Apply ALL cascade operations:

- PERSIST
- MERGE
- REMOVE
- REFRESH
- DETACH

Powerful

Use only when child completely depends on parent