NORMALIZATION AND DENORMALIZATION

◇ Normalization

What it means:

Breaking data into **smaller, related tables** to reduce duplication and improve accuracy.

Goal:

✓ Avoid repeated data

✓ Maintain data consistency

✓ Make updates safer and easier

Example (Before normalization):

StudentName	CourseName	Instructor
Ali	SQL	Sara
Ali	Python	Sara

Instructor name is repeated.

After normalization:

- **Students** table
- **Courses** table
- **Instructors** table
- Linked using **foreign keys**

Advantages:

- Less data duplication
- Data is more accurate
- Easier to update data

Disadvantages:

- More tables
- Queries may need **JOINs**, which can be slower

◇ Denormalization

What it means:
Combining tables or **adding repeated data** to improve query speed.

Goal:

✓ Faster data retrieval

✓ Fewer JOINs

Example (Denormalized):

StudentName	CourseName	Instructor
Ali	SQL	Sara
Ali	Python	Sara

Instructor name is repeated on purpose.

Advantages:

- Faster queries
- Simpler SELECT statements

Disadvantages:

- Data duplication
- Risk of inconsistent data
- Updates are harder

◇ Quick Comparison

Feature	Normalization	Denormalization
Data duplication	❌ Low	✓ High
Performance	Slower reads	Faster reads
Data accuracy	High	Lower risk
Storage	Efficient	Uses more space
Joins	Many	Few or none

◇ When to use which?

- **Use Normalization** → When designing databases, OLTP systems (e.g., student enrollment, banking).
- **Use Denormalization** → For reporting, analytics, and read-heavy systems.

🔷 خلاصة للحفظ 📝

- **Norm.** = تنظيم + جداول أكثر + تكرار أقل
- **Denorm.** = سرعة + جداول أقل + تكرار أكثر