



# Lecture – Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
  - ❑ Please review Code of Conduct (in Student Undertaking Agreement) if unsure
- ❑ No question is daft or silly - **ask them!**
- ❑ There are Q&A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❑ Should you have any questions after the lecture, please schedule a mentor session.
- ❑ For all non-academic questions, please submit a query: [www.hyperiondev.com/support](https://www.hyperiondev.com/support)

# Lecture Objectives

1. Introduction to Database Normalisation
2. An overview of how we apply normal forms

# Database Normalisation

Database normalization is a systematic process used in database design to organize and structure data in a relational database management system (RDBMS).

It involves breaking down large, complex tables into smaller, related tables and ensuring that data is stored efficiently and accurately.


Why is it important?

1. Minimize data redundancy
2. Reduce the likelihood of data anomalies (errors)
3. Improve data integrity while maintaining data relationships.

# Database Normalisation



There are six normal forms, but we will only look at three:

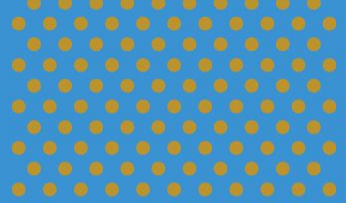
1. First normal form 1NF
  2. Second normal form 2NF
  3. Third normal form 3NF
- 

# Database example

We have a Student\_Grade\_Report table from a School database.

**Student\_Grade\_Report** (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

# First Normal Form


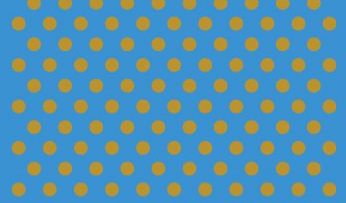


For a table to be in first normal form, the following must be satisfied:



- a single cell must not hold more than one value (atomicity)
- there must be a primary key for identification
- no duplicated rows or columns
- each column must have only one value for each row in the table

# First Normal Form



Our Student\_Grade\_Report table is not in 1NF because we have a repeating group – course information. A student can take many courses.

Solution: remove the repeating group. In this case, we are going to remove the course information for each student to create a new table with its own primary key that will uniquely identify the attribute value.



# First Normal Form

The Student table is now in first normal form with the repeating group removed, and we have a new table called StudentCourse.

**Student** (StudentNo, StudentName, Major)

**StudentCourse** (StudentNo, CourseNo, CourseName, InstructorNo,  
InstructorName, InstructorLocation, Grade)

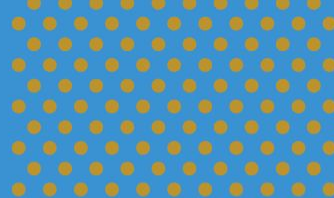
# First Normal Form

1NF does not take care of all potential issues. Let's consider the case where you want to delete a student from the StudentCourse table:

**StudentCourse** (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

Deleting a *student* might also delete critical information about a course.

# Second Normal Form



For a table to be in 2NF, it must satisfy the following:

- be in 1NF already
- has no partial dependency (all non-key attributes are fully dependent on a primary key)



# Second Normal Form

The Student table is already in 2NF because it has a single-column primary key.

When examining the Student Course table, we see that not all the attributes are fully dependent on the primary key. To be specific, all course information is not dependent on the composite primary key, and only grade is fully dependent on it.

**StudentCourse** (StudentNo, CourseNo, CourseName, InstructorNo,  
InstructorName, InstructorLocation, Grade)

# Second Normal Form

This partial dependency violates the 2NF rule, which states that all non-prime attributes should depend on the entire composite primary key. To achieve 2NF, you could create separate tables for courses and instructors, each with their own primary keys, and then reference these tables in the StudentCourse table using foreign keys.

We must:

- Identify the new table that contains the course information.
- Identify the primary key for the new table.

# Second Normal Form

The three new tables are shown below:

**Student** (StudentNo, StudentName, Major)

**CourseGrade** (StudentNo, CourseNo, Grade)

**CourseInstructor** (CourseNo, CourseName, InstructorNo,  
InstructorName, InstructorLocation)

# Third Normal Form

For a table to be in 3NF it must:

- be in 2NF
- have no transitive partial dependency.

Transitive partial dependency: simply put, if A is related to B, and B is related to C, then there's a transitive relationship between A and C, even though they are not directly connected.

# Transitive Partial Dependency

Students (via StudentNo) are indirectly connected to instructors through the CourseGrade table and the CourseInstructor table.

Consider the case where you want to find out who the instructors are for the courses a specific student has taken. You would need to go through the CourseGrade table (to find the courses the student took) and then the CourseInstructor table (to find the instructors for those courses). This relationship involves traversing two tables indirectly, making it a transitive relationship.

**Student** (StudentNo, StudentName, Major)

**CourseGrade** (StudentNo, CourseNo, Grade)



**CourseInstructor** (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)



# Transitive Partial Dependency



Why is this dependency an issue? In our case:

- 
- Updating course information could lead to inconsistencies for instructor information.
  - Deleting a course may also delete instructor information
- 

# Third Normal Form

1. Eliminate all dependent attributes in transitive relationships from each of the tables that have a transitive relationship.
2. Create new tables with removed dependency.
3. Check the new tables as well as tables modified to make sure that each table has a determinant and that no table contains inappropriate dependencies.

**Student** (StudentNo, StudentName, Major)

**CourseGrade** (StudentNo, CourseNo, Grade)

**CourseInstructor** (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)

# Third Normal Form Result

**Student** (StudentNo, StudentName, Major)


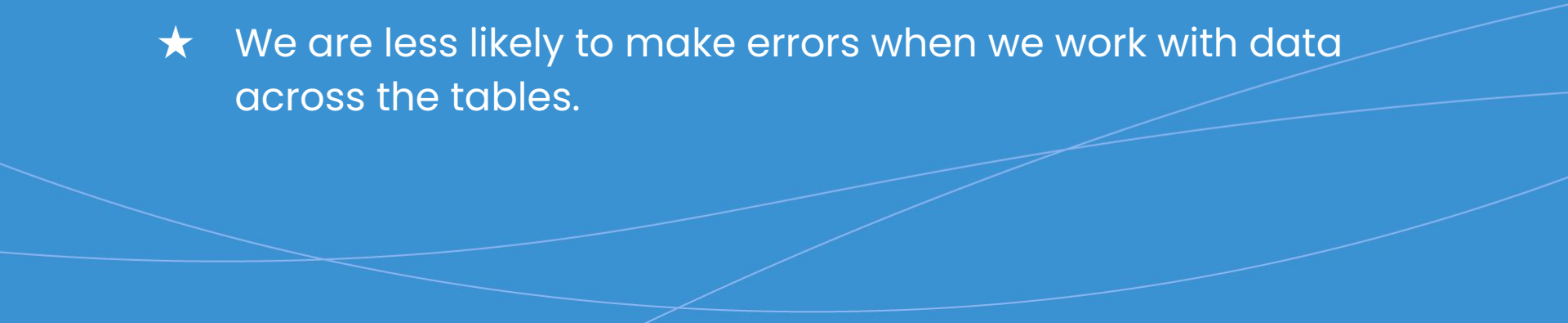
**CourseGrade** (StudentNo, CourseNo, Grade)

**Course** (CourseNo, CourseName, InstructorNo)

**Instructor** (InstructorNo, InstructorName, InstructorLocation)

# Database Normalisation



- 
- ★ And so, we have successfully broken down a large, complex table into 4 smaller, related tables.
  - ★ We have also ensured that data is stored efficiently and accurately.
  - ★ We are less likely to make errors when we work with data across the tables.
- 

# Resources



A Database Normalisation Step-By-Step Guide with useful examples:

<https://www.databasestar.com/database-normalization/>





# Questions and Answers

Questions around Database Normalisation

