UNIVERSITY OF LEEDS

SQL

Outer Joins and SQL with Python

COMP1850 Semester 2

Week 2 Session 1

UNIVERSITY OF LEEDS

Last Time…

You saw how to use joins to connect tables

- To enable queries across multiple tables
- To construct SQL code for those queries

However, we only used **inner joins.**

UNIVERSITY OF LEEDS

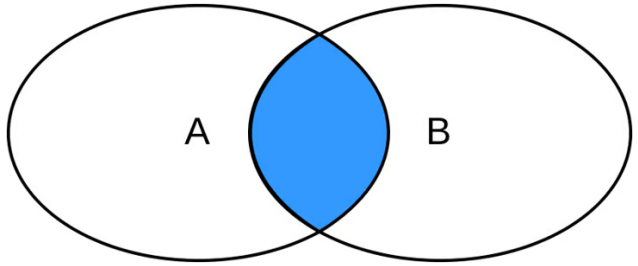
This session…

We will look at **outer joins** and how these differ from inner joins, as well as how we can integrate SQL code into Python to make functional programs.

By the end of the session, you should be able to:

- Explain the difference between an inner, a full outer, and left/right outer joins
- Choose the correct join for a given scenario
- Be confident using all four types of join in your code
- Be able to write a simple python program which uses SQLite.

Recap – Inner Joins

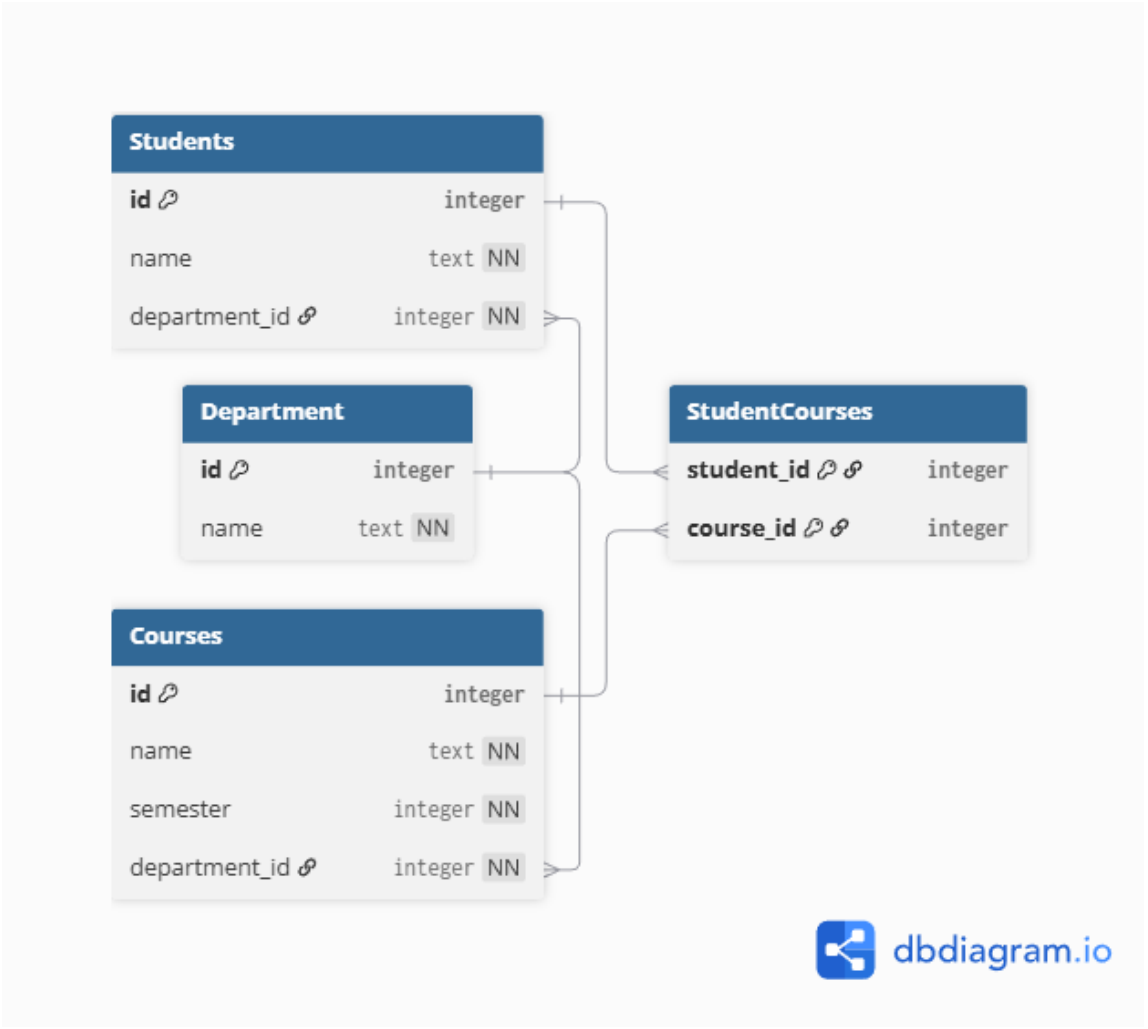


UNIVERSITY OF LEEDS

A university is using the database described in the diagram to store their student course data.

They want to get a list of students' names and the names of their departments.

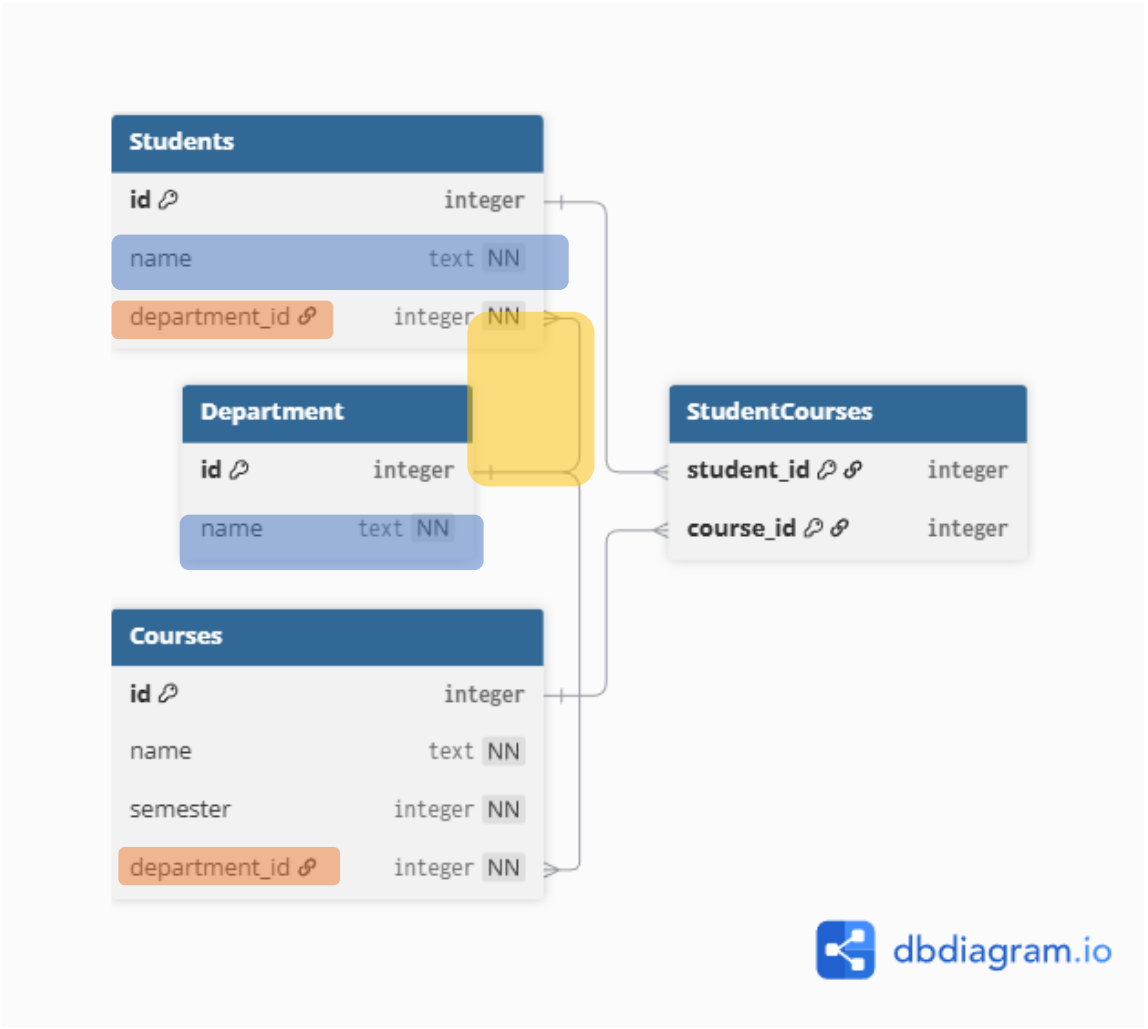
How would they do this?



Recap – Inner Joins – How?

They want to get a list of students' names and the names of their departments.

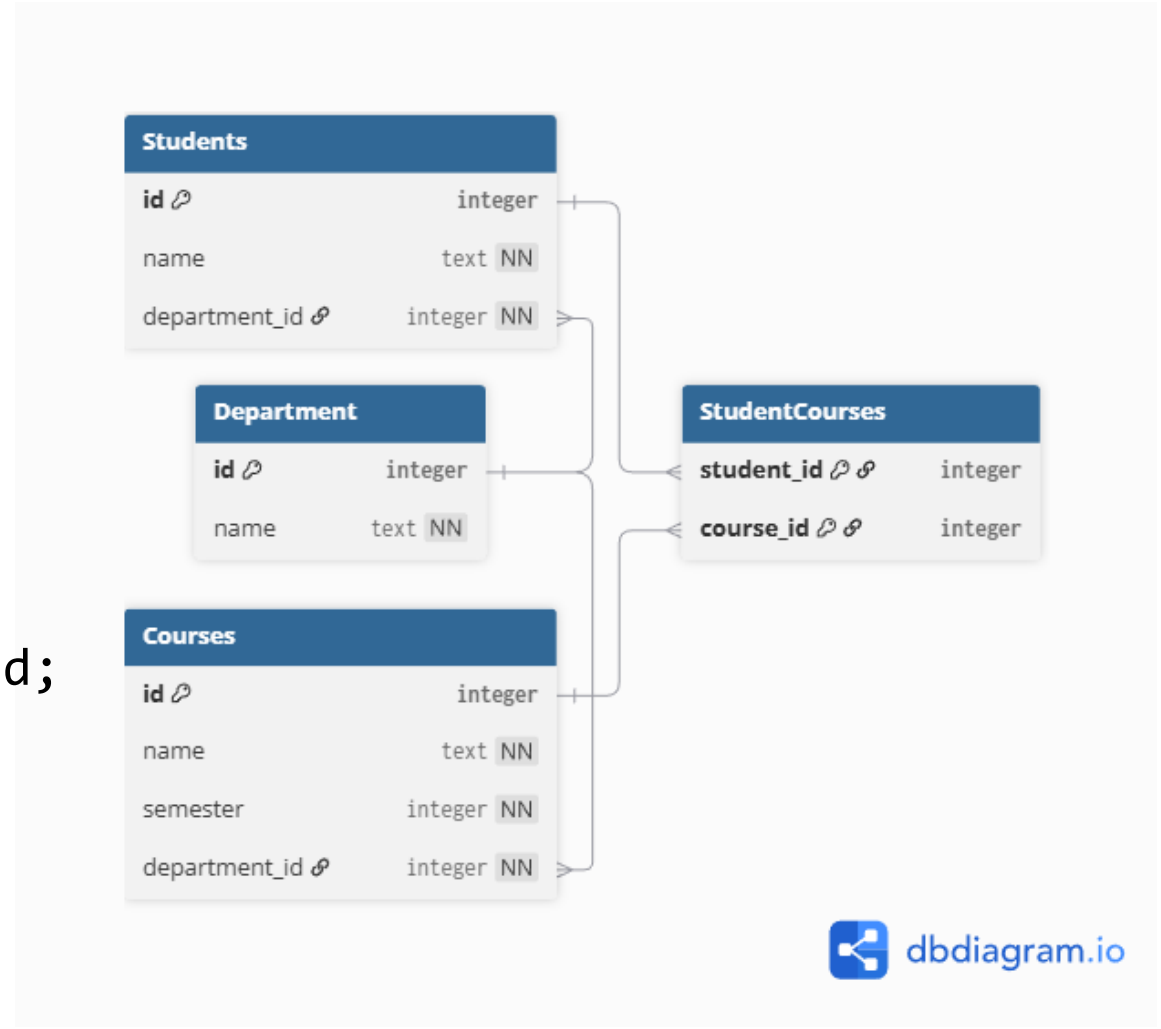
1. Locate the data we want in the database and the relevant tables
2. Identify the link between the tables
3. Identify how we create the join (which attributes from each table?)



Recap – Inner Joins - SQL

They want to get a list of students' names and the names of their departments.

```
SELECT Students.name, Department.name
FROM
Students JOIN Department
ON Students.department_id = Department.id;
```



Recap – Inner Joins - Limitations

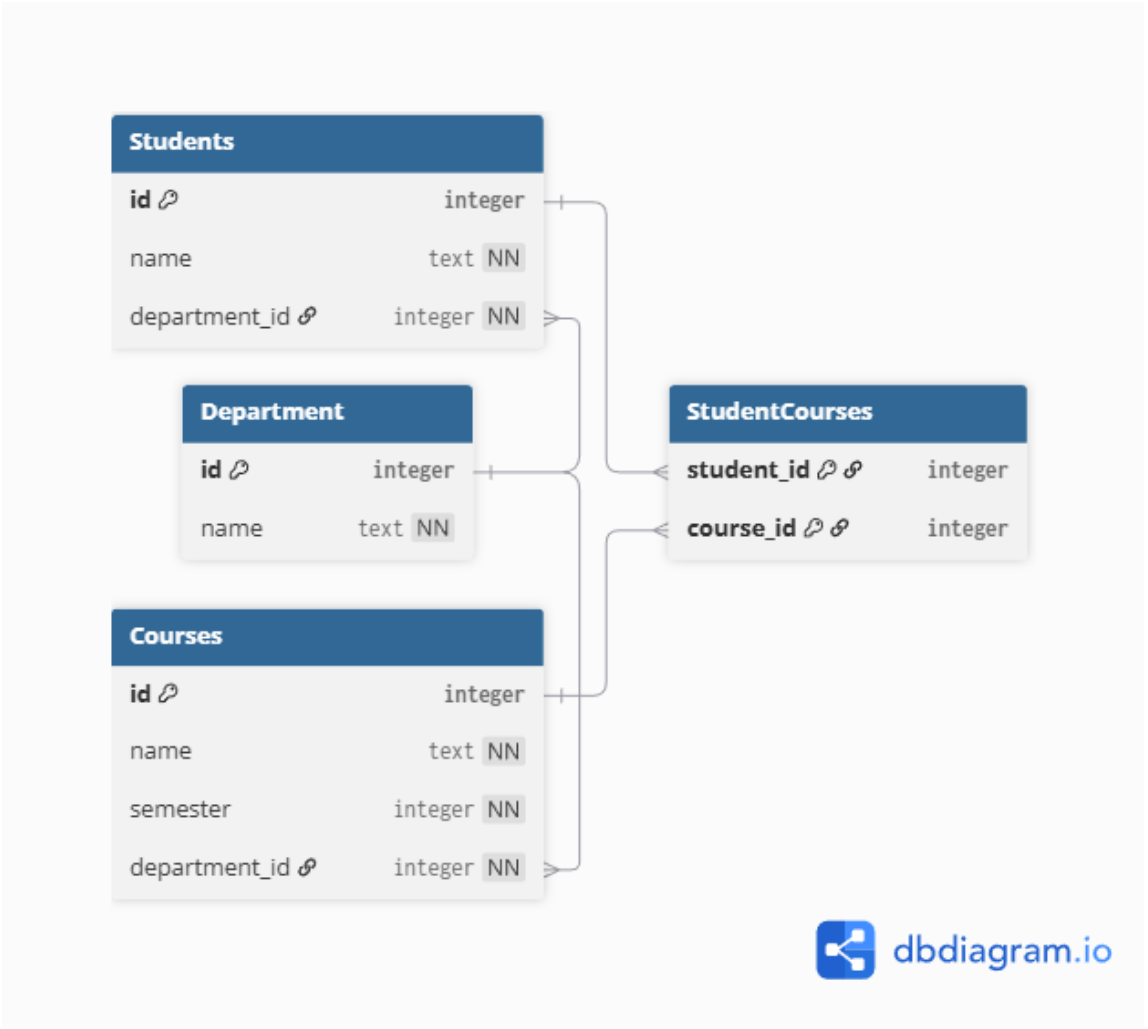
There are limitations to inner joins.

They only include rows which have a **match** on the given attributes – the **intersection** of the two tables.

But sometimes you want to also see rows where there **are not** matches.

- Students with no assigned department
- Departments with no students

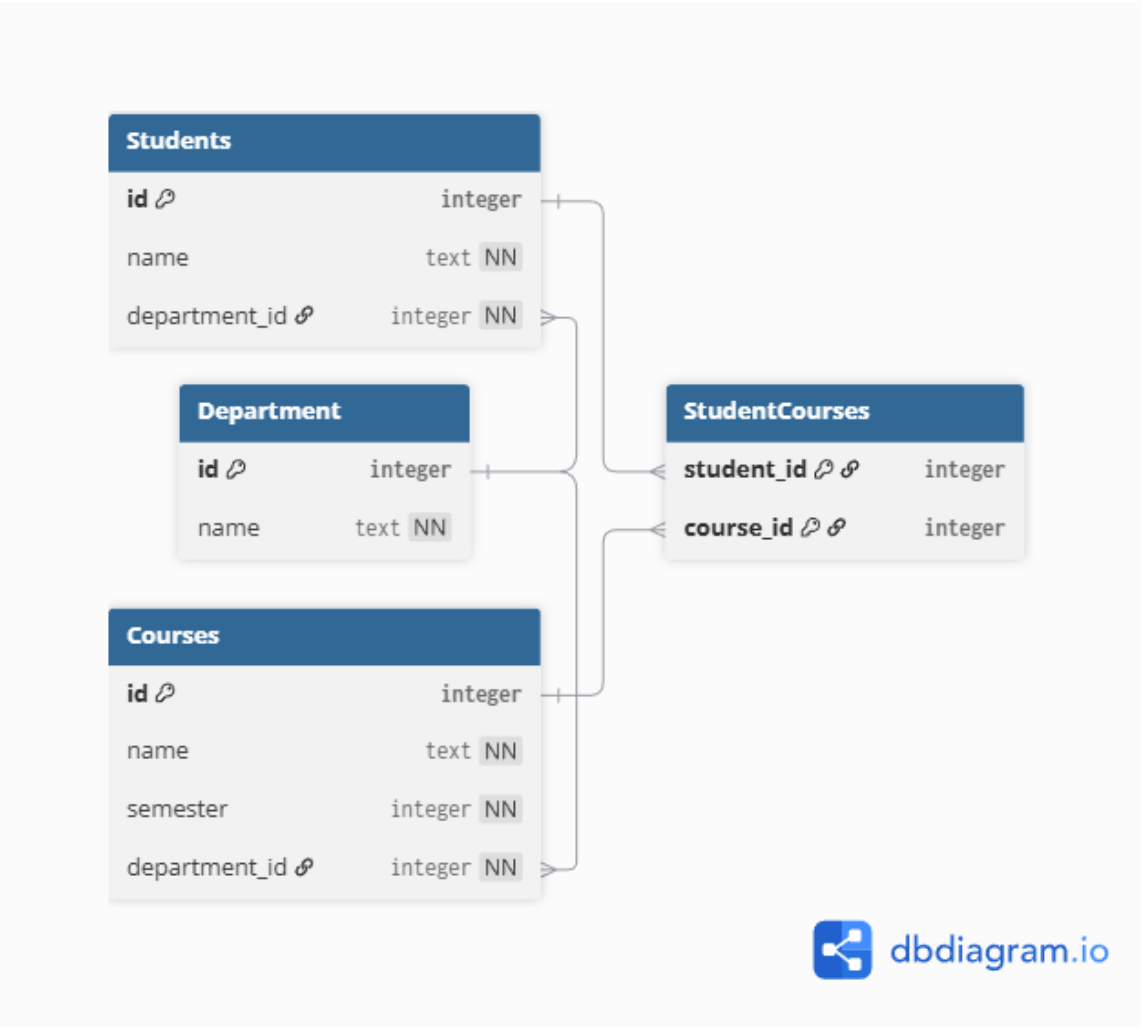
For this, we need to use a different kind of join.



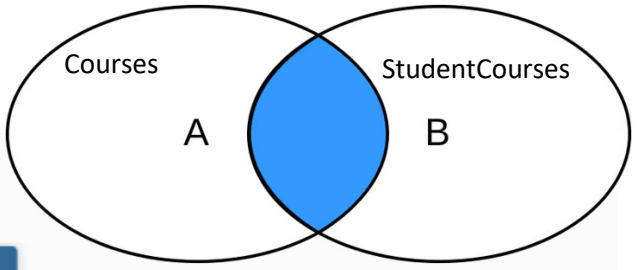
Outer Joins

The university wants to find any courses where there are fewer than 20 students registered as they are reviewing what courses they offer.

First, let's think about how we would do this using the process from before:



Inner Join – partial solution

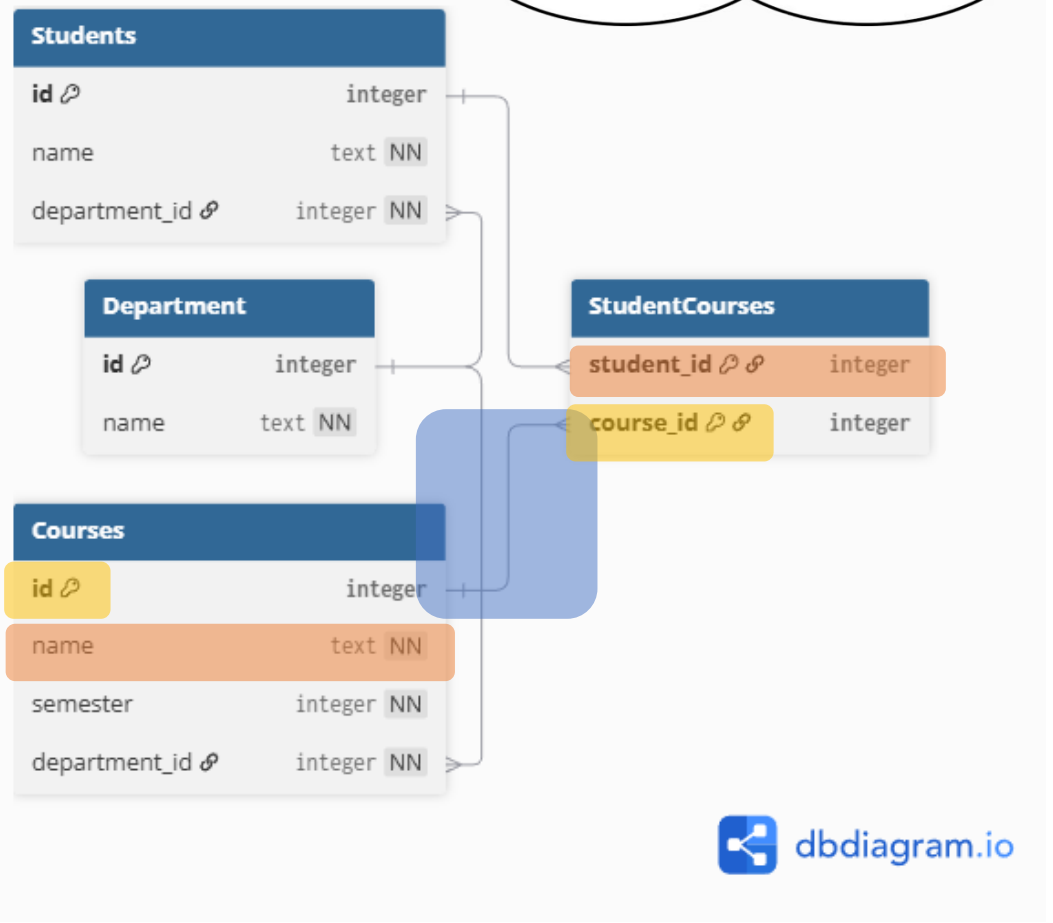


The university wants to find any courses where there are fewer than 20 students registered as they are reviewing what courses they offer.

```
SELECT name,COUNT(student_id) AS TotalStudents
FROM
Courses JOIN StudentCourses
ON Courses.id=StudentCourses.course_id
GROUP BY name HAVING TotalStudents<20;
```

But…

This only finds courses which have **at least one student**.

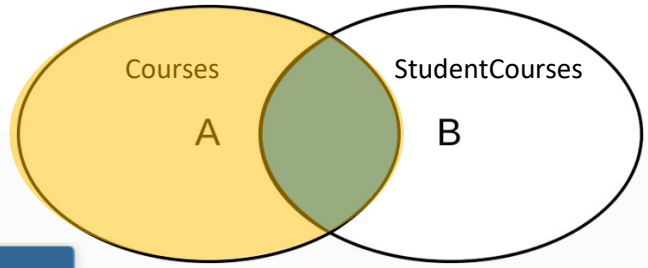


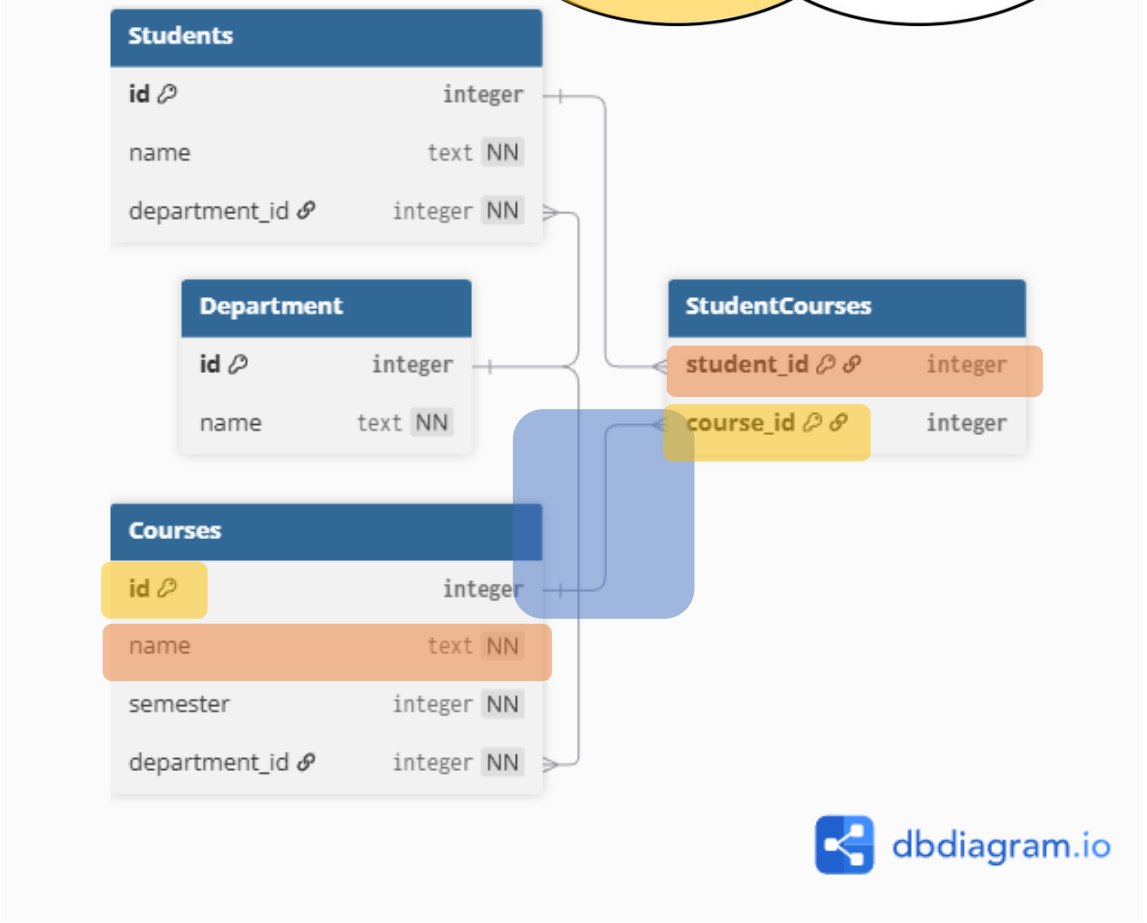
Inner Join v Outer Joins

UNIVERSITY OF LEEDS

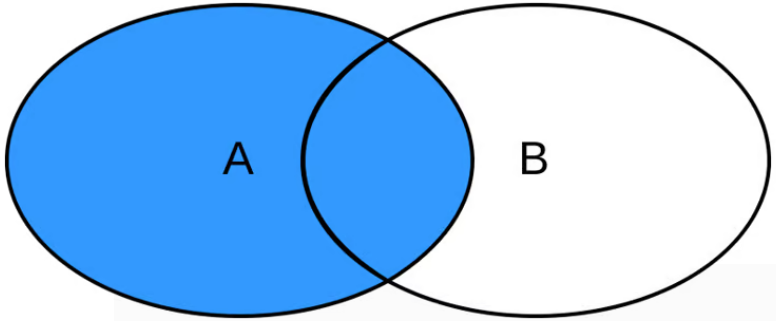
An Outer Join allows us to include NULL results

- In this case courses with no students
- We include the intersection but also the remaining (outer) part of the Courses set





Left Outer Join

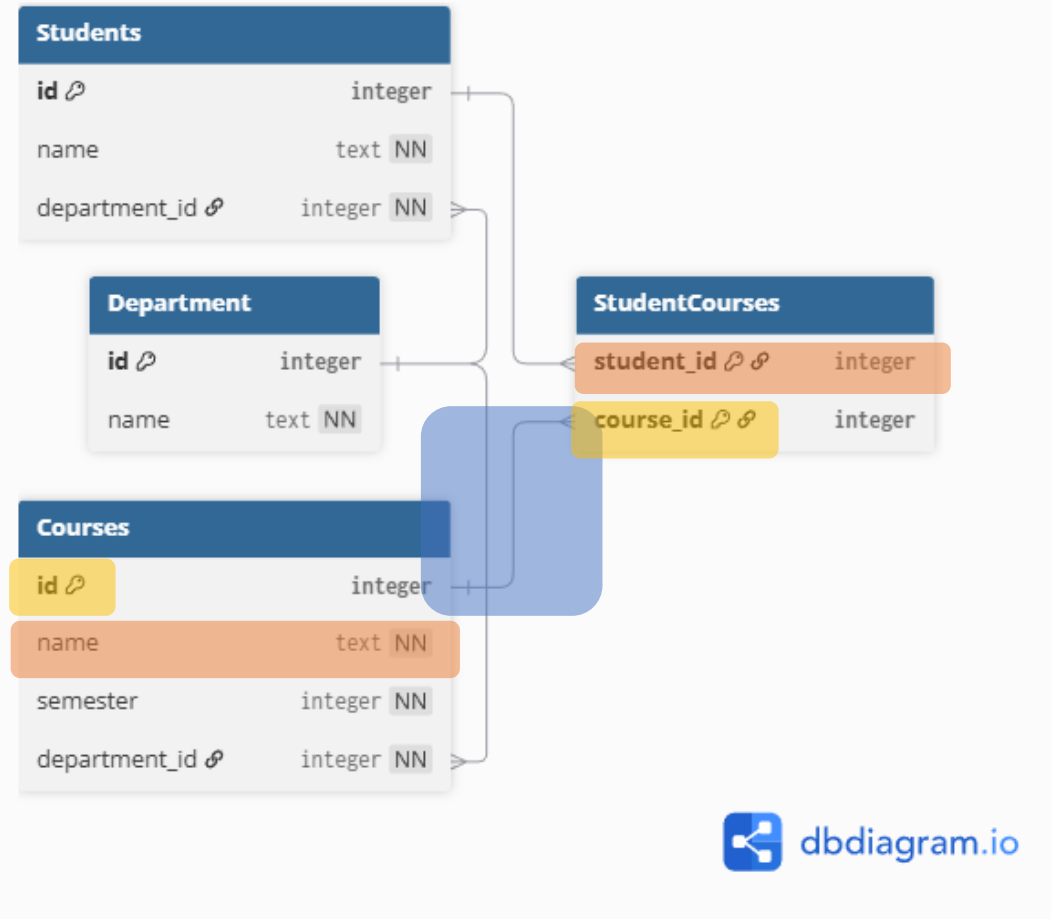


UNIVERSITY OF LEEDS

The university wants to find any courses where there are fewer than 20 students registered as they are reviewing what courses they offer.

We should use a **left join** instead of an **inner join**.

```
SELECT name, COUNT(student_id) AS TotalStudents
FROM
Courses LEFT JOIN StudentCourses
ON Courses.id=StudentCourses.course_id
GROUP BY name HAVING TotalStudents<20;
```



Left Outer Join - Explanation

UNIVERSITY OF LEEDS

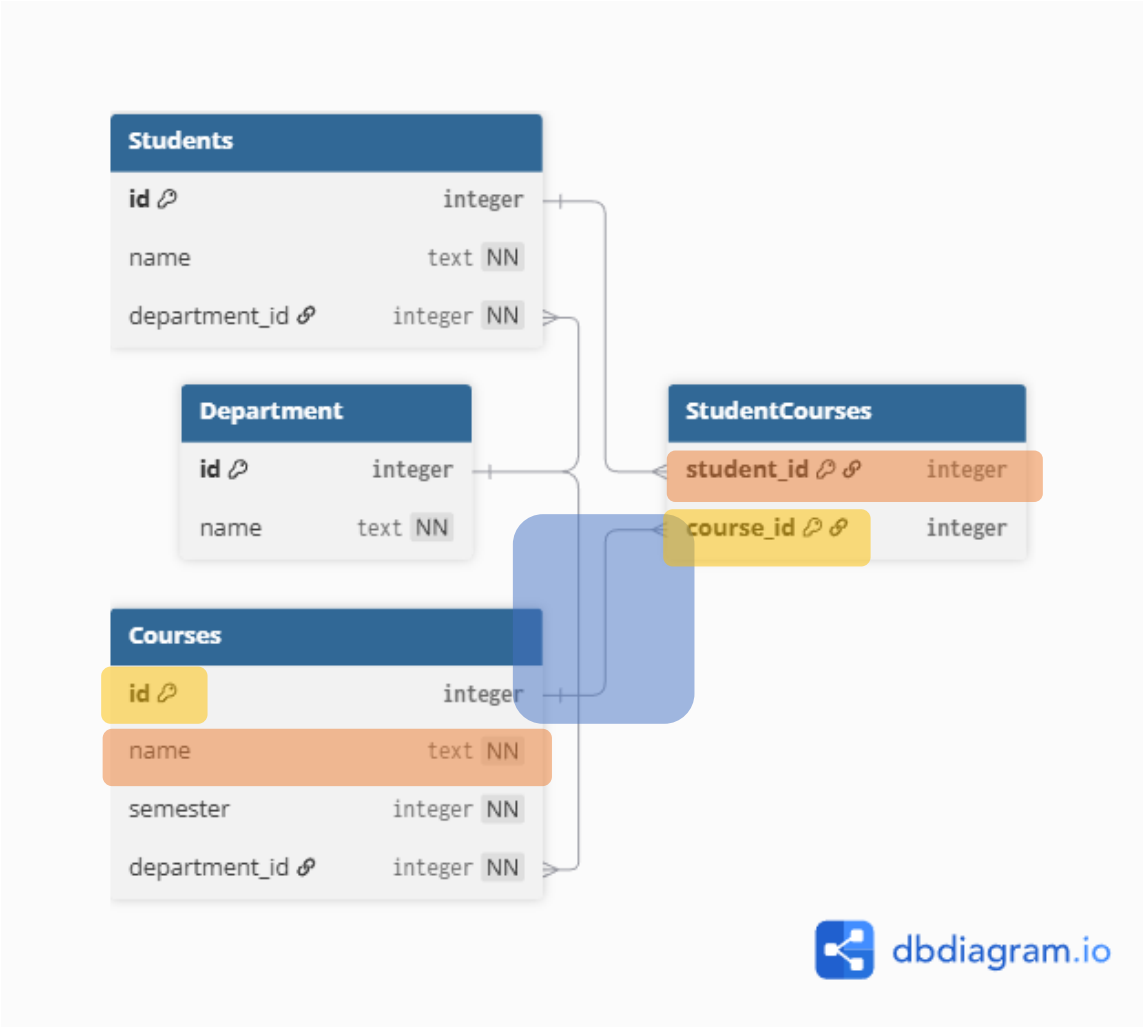
```
Courses LEFT JOIN StudentCourses
```

This is an **outer join –** a join which also includes rows of the “left table” which do not have a match.

A **left join** is one of three kinds of outer join – it takes **every row** from the table named on the left, regardless of whether it has a match in the table on the right.

Task:

Edit your code so that you can see all the columns, and see what blank columns are filled with.



UNIVERSITY OF LEEDS

Other OUTER JOINs

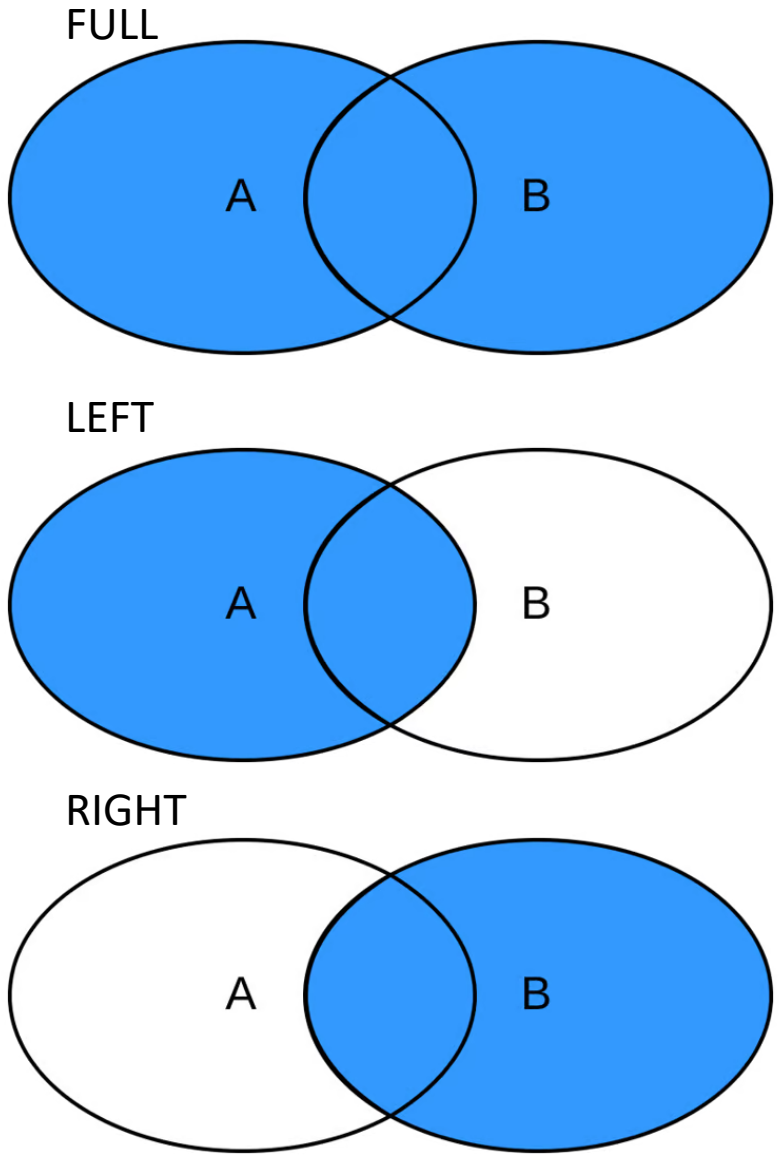
There are 3 outer joins:

LEFT and RIGHT JOIN

These are logically equivalent- anything you can do with a right join can be done with a left join - and so we tend to avoid RIGHT JOINs.

FULL OUTER JOIN

This join shows all rows from both tables – **however** sqlite does not actually have native support for full outer joins! When you run a full outer join in sqlite it will rewrite this as the UNION of LEFT and RIGHT join.



UNIVERSITY OF LEEDS

Tasks

The best way to develop your skills working with joins is to practice.

In 2_practice, you will find a database, some tasks and a database diagram.

UNIVERSITY OF LEEDS

SQLite in Python

SQL is not often used as a stand-alone language.

When you use it next year, and in future, you are likely to be using it alongside other languages to access databases within a program.

UNIVERSITY OF LEEDS

A Note On ORMs

In development, you may not even use SQL

- You are fairly likely to use an Object-Relational Mapper (ORM)
- The ORM can create and control your database.

These libraries allow us to define databases using **objects**

- as you have seen before, relations and classes follow very similar design principles
- and often allow us to write queries in simpler, language-specific ways.

The principles of SQL are still very important

Using SQL in Python

UNIVERSITY OF LEEDS

In Python, we can use the sqlite library to connect to and manipulate databases.

Documentation:

https://docs.python.org/3/library/sqlite3.html

Open 3_python/example/example.py

Connections

UNIVERSITY OF LEEDS

To connect to a database, we need to create a special object called a connection:

```
conn = sqlite3.connect(db_path)
```

- `db_path` is the external file location of the database (*.db file)
- The connection object is how we will access the database and run queries.
- It represents our external database file within the code

UNIVERSITY OF LEEDS

Queries

When we want to run a query, we can write the SQL into a string:

```
query = "SELECT s.id, s.name, d.name FROM Students s JOIN Department d ON
s.department_id=d.id WHERE s.id=?;"
```

Note that we still need to use a **semicolon** at the end!

We can add variables into our query

- In this case, the student ID we are searching for
- This is represented by a single **question mark**.

UNIVERSITY OF LEEDS

Executing the Query & Cursors

```
query = "SELECT s.id, s.name, d.name FROM Students s JOIN Department d ON
s.department_id=d.id WHERE s.id=?;"
cursor = conn.execute(query, (choice,))
```

To execute a query, we pass the query and a **tuple containing the data we want to replace the variable ? with when it runs.**

Note that

- the data **must be a tuple** and in the same order as the ?s.
- `(choice)` is **not** a tuple. `(choice,)` **is** a tuple.

The result of our query is a **cursor** object

- it points to the results of our query.

The cursor object

```
cursor = conn.execute(query, (choice,))
```

cursor points to the results of our query.

- This is because **queries may have a huge number of results**
- If it just returned them all as a list, it would be very slow
- And, potentially, cause crashes due to memory requirements

UNIVERSITY OF LEEDS

Accessing the results from the cursor

```
student = cursor.fetchone()
# or
for row in cursor:
    print(row)
```

Now we have our results, we can access them in different ways.

- In our scenario, we are searching for **one** student
  - we can use `fetchone()` which will return a **tuple** of our requested data
  - or None if no data was found.
- We could also **iterate** through the cursor
  - if you are expecting multiple results – each record is a tuple.

Writing Longer Queries

UNIVERSITY OF LEEDS

```
query = """
SELECT s.id, s.name, d.name
FROM Students s JOIN Department d
ON s.department_id=d.id
WHERE s.id=?; """
```

You may find it hard to read queries written as single long lines – you can use triple quotes to create a multi-line string if you find this easier to work with.

Quick Task

Complete the ‘`review_student_numbers()`’ function by writing the code needed to query the number of students on each course from the database, running the query, and processing the result.

UNIVERSITY OF LEEDS

Practicing SQL with Python

In leeds_eats/ you will find a new database. This is a food delivery service, similar to Uber Eats or Deliveroo, which connects drivers to orders.

A basic menu and some function stubs have been created – the task file has the full details of what you should implement.

UNIVERSITY OF LEEDS

Next Time

We will finish off the SQL topic with a larger project

- bringing together all the kinds of queries you have written
- and more extensive Python code.