Lecture 2

COMP207

How does the website work?

- I will demonstrate how I have intended for you to use the website
  - Let me know how you would like to have access to the information if it is not currently convenient for you!
- E.g:
  - How to access textbooks
    - you can read them if you like reading textbooks, but they are not directly required
    - They should match what is going on in the course otherwise, with a few exceptions that I will mention directly when we get to them.
  - How to access weekly assessments
    - And see which are assessments that give some part of your grade and which are just for fun quizzes that does not

Brief overview over videos

- The videos for this week are about most everything to do with SQL, except how to query them
  - Query = ask questions/get output
  - (I will use 1 query here in these lectures though, namely the simplest possible one)

More detail

- Specifically, the videos cover:
  - The history of SQL
  - How to create a database
  - How to create tables
  - How to create constraints (UNIQUE, FOREIGN KEY, PRIMARY KEY)
  - How to insert, remove and add information to the tables
  - SQL Injections (not required for the exam)

Together called data definition language

This + queries = formally: data manipulation language
(informally: CRUD)

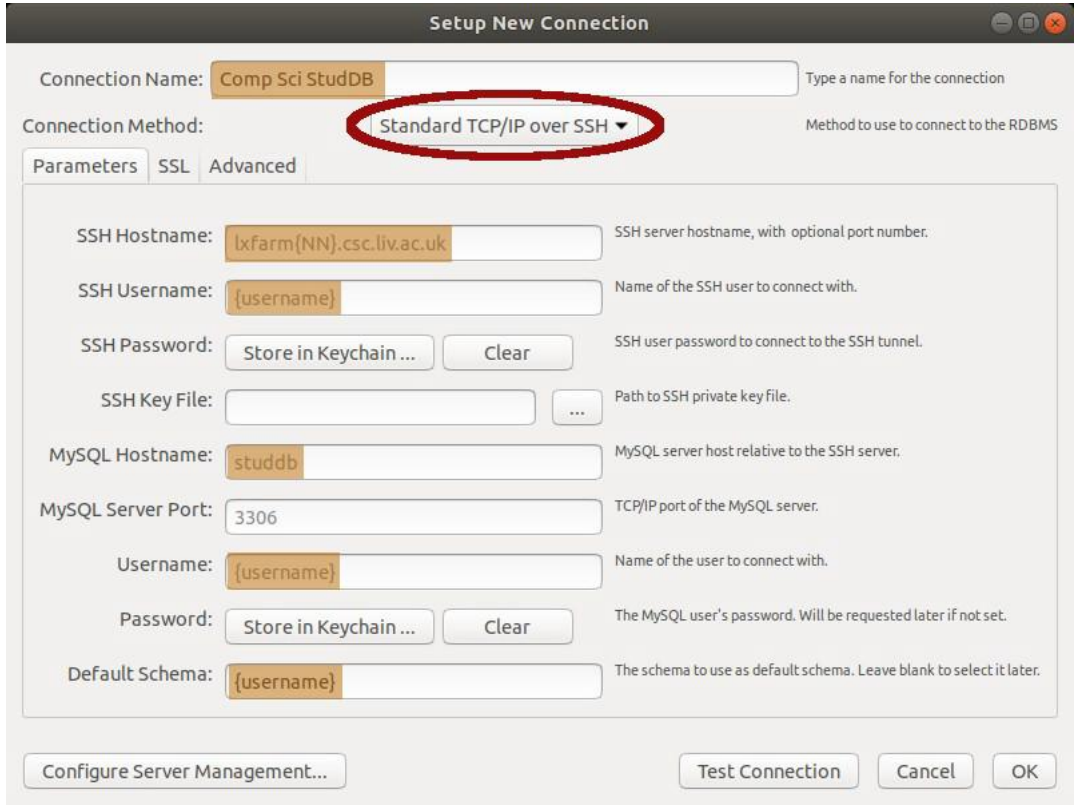
Examples covered today

- How to create a database
- How to access the university server in MySQL
- How to create a table
- Persistency
- How to create constraints
- How to insert data
- How to update data (with some comments on WHERE)
- How to delete data
- SQL Injections

Create databases

- CREATE DATABASE xyz;
- USE xyz;

How to access the university server in MySQL



Create table and persistency

- CREATE TABLE Students (
  name VARCHAR(20),
  number INT,
  programme VARCHAR(4)
  );

From Wikipedia on object permanence

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702
Danny	20241112	G702
John	*null*	G702

Create constraints

```
CREATE TABLE Students (
    name VARCHAR(20),
    number INT,
    programme VARCHAR(4),
    CONSTRAINT pk_students PRIMARY KEY (number)
);
```

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702
~~Danny~~	~~20241112~~	~~G702~~
~~John~~	*null*	~~G702~~

Create constraints

```
CREATE TABLE Students (
    name VARCHAR(20),
    number INT,
    programme VARCHAR(4),
    CONSTRAINT pk_students PRIMARY KEY (number)
);
```

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

Create constraints

```
CREATE TABLE Students (
    name VARCHAR(20),
    number INT,
    programme VARCHAR(4),
    CONSTRAINT pk_students PRIMARY KEY (number)
);

CREATE TABLE Enrolment (
    student_number INT,
    course_id VARCHAR(20),
    CONSTRAINT pk_students PRIMARY KEY(student_number,course_id)
);
```

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

Enrolment

student_number	course_id
20241989	COMP207
20241112	COMP105
20241989	COMP105

Create constraints

```
CREATE TABLE Students (
    name VARCHAR(20),
    number INT,
    programme VARCHAR(4),
    CONSTRAINT pk_students PRIMARY KEY (number)
);

CREATE TABLE Enrolment (
    student_number INT,
    course_id VARCHAR(20),
    CONSTRAINT pk_students PRIMARY KEY(student_number,course_id),
    CONSTRAINT fk_enrolment_students FOREIGN KEY (student_number)
    REFERENCES Students(number)
);
```

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

Enrolment

student_number	course_id
20241989	COMP207
20241112	COMP105
20241989	COMP105

Insert data

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

- INSERT INTO Students(name,number,programme) VALUES ('Anna', 20241989, 'G402’);
- INSERT INTO Students VALUES ('Oliver', 20241112, 'G702');
- INSERT INTO Students(programme,number) VALUES (20240000, 'G402');
- INSERT INTO Students(name,number,programme) VALUES ('Anna', 20241989, 'G402'), ('Oliver', 20241112, 'G702');

Update values

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

- UPDATE Students
  SET name = 'Danny', programme = 'G700'
  WHERE number = 20241112;
- UPDATE Students
  SET name = ‘Oliver'
  WHERE number = 'G700';

Conditions in WHERE clauses

- Comparisons: =,<,<=,>=,>,<> (or != for the last)
  - Used for equals, strictly less than, less than or equal, greater than or equal, strictly greater and not equal of e.g. numbers
- Conditions can contain:
  - AND
    - E.g. if you want both that the name is Oliver and the programme is G402, you write WHERE name = ‘Oliver’ AND programme = ‘G402’
  - OR
    - Similar to AND, but used if you want or…
  - NOT
    - If you want everything but something in particular
  - BETWEEN
    - E.g. “Price BETWEEN 10 AND 20” if you want the price to be between 10 and 20
  - LIKE
    - For string matching
    - _ matches any 1 letter and % any number of letters
    - E.g. “Name LIKE ‘O%r’” and “Name LIKE ‘O____r’” matches Oliver

Conditions in WHERE clauses cont.: IN

```
UPDATE Students
SET name = 'Victor'
WHERE name IN ('John', 'Sebastian');
```

Special version using queries – see the video on queries – the optional part

Students

name	number	programme
Anna	20171989	G402
Oliver	20171112	G702
Danny	*null*	G702
John	*null*	G702

Delete values

Students

name	number	programme
Anna	20241989	G402
Oliver	20241112	G702

- UPDATE Students
  SET name = 'Danny', programme = 'G700'
  WHERE number = 20241112;
- UPDATE Students
  SET name = ‘Oliver'
  WHERE number = 'G700’;
- DELETE FROM Students
  WHERE number = 20241112;
- DELETE FROM Students
  WHERE number = 'G700’;