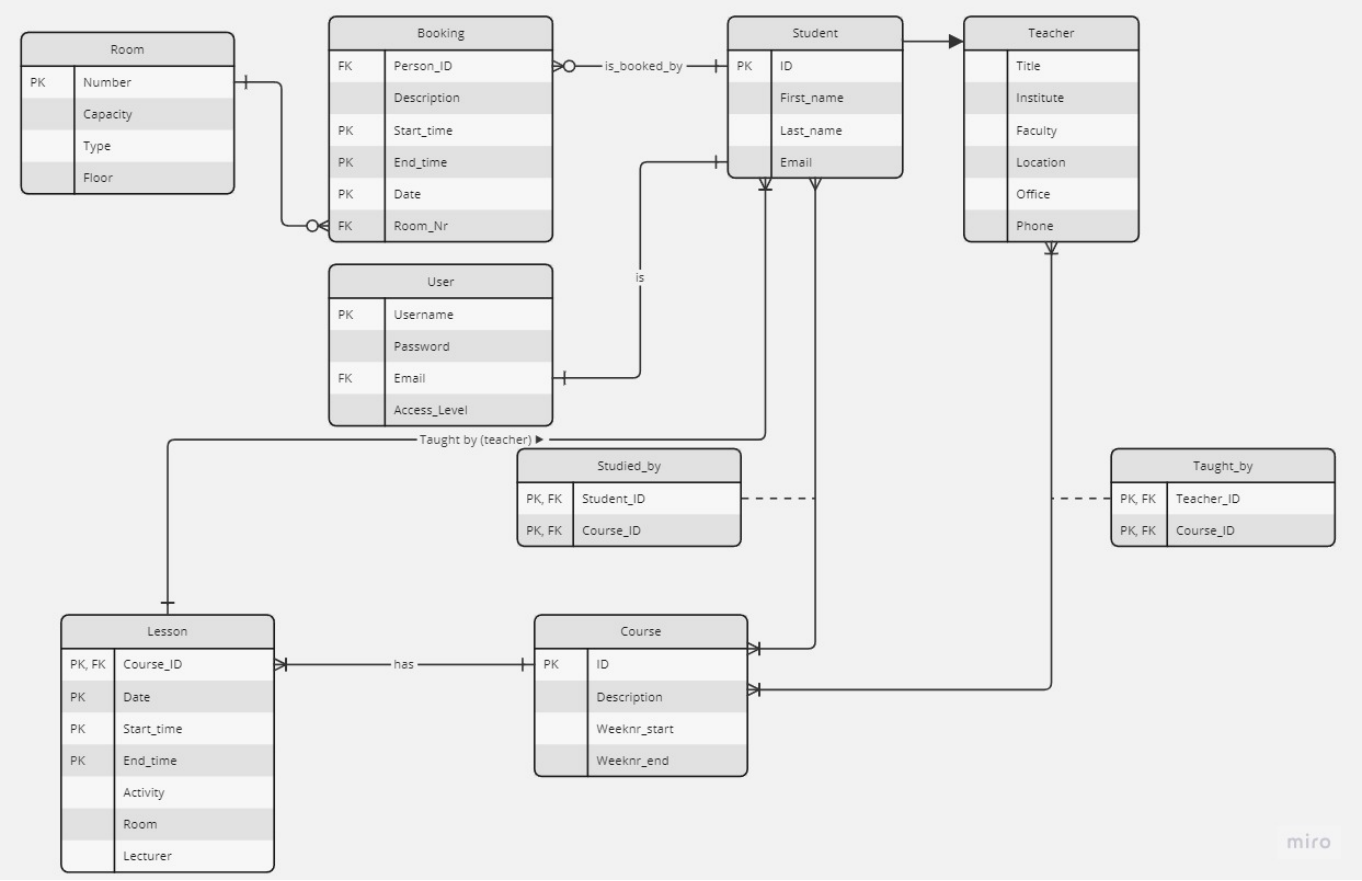
Group Members: Daniel Høyland, Ferdinand Wedén Snapa, Inga Kvam, Jan Olav Lyche Aspelund, Odin aas

ERD

The first draft of the ERD diagram (not the final):



In the first draft of the ERD everything was centered around Student, user was just a part of student and teacher was a subgroup of student, with some extra information and a different Relation to The Courses. Rooms only had a connection to booking. Lessons was a separate entity from rooms and booking with multiple relations to both Student and Teacher.

The idea behind the first iteration of the ERD was that the student was the main location to store values for each user and that Teachers were mostly just student with higher access level, Teachers had to book rooms for each lesson, the same way that a student would book a room. We also have the Junctions tables from student to Course called Studied_by and from Teacher to course called Taught_by in order to connect the two again, and to represent that a teacher may have multiple courses and that a course might have multiple teachers, same with the Student

We all started working together on the ER diagram. We all had meetings at school where all shared our thoughts on how the ERD should look and function, imagining how the database would look for the time plan app we use at NTNU

Ferdinand was the main contributer to the ERD in miro and the main person updating it based on the Peer reviews, with comments from other people in the group. He edited it after we found out some more when doing the queries.

PEER REVIEW

The feedback we got from the review helped us improve out ERD by seperating the student and lacturer into two different classes, instead of it being a subclass-superclass relationship, and instead have the user itself be the actual data found in the database.

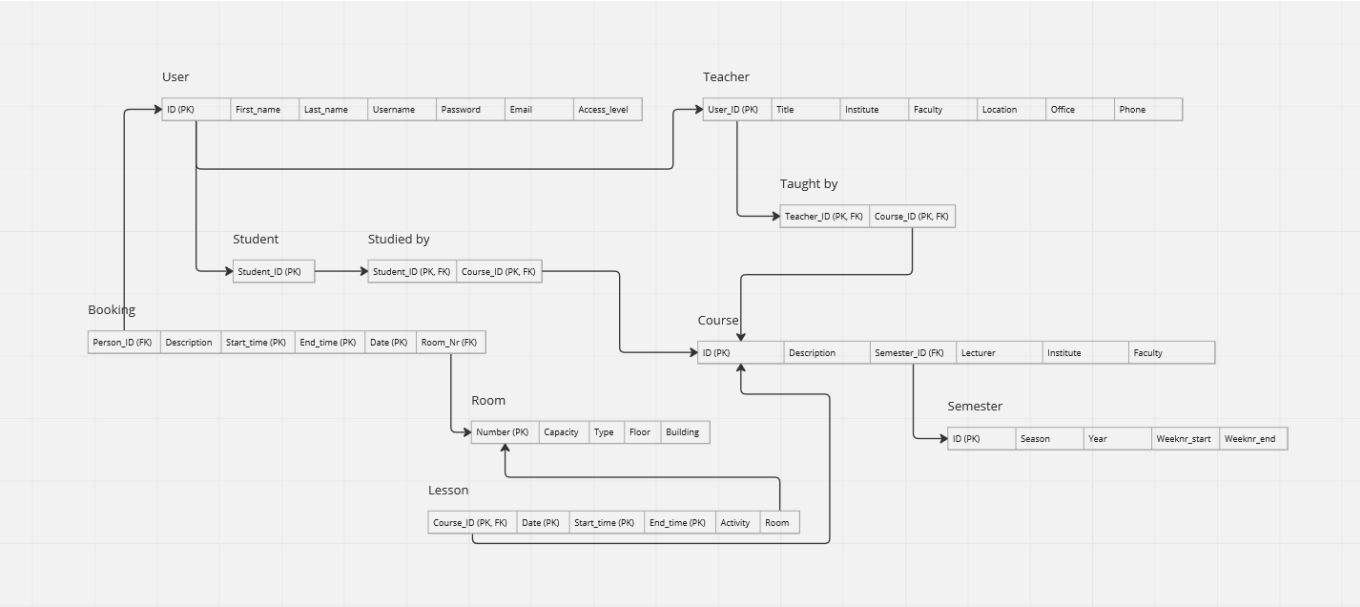
this helps by simplifying the ERD making it easier to read, and it also makes accessing the data easier.

We also got the idea of adding a Semester to the course and connecting the Lessons up to Rooms for better representation of where the Lesson takes place, We made sure that Lessons have Zero to many as some Lessons may be online, and other have multiple classrooms booked. Lecturers were implemented as a user that may moderate Courses, they can also be teachers and have a field dedicated to the Lecturer in charge of each Course.

We all participated equally in the peer review, we discussed together what their strengths and weaknesses were, and discussed their response to our ERD.

Relations logical model

Logical relationship model:



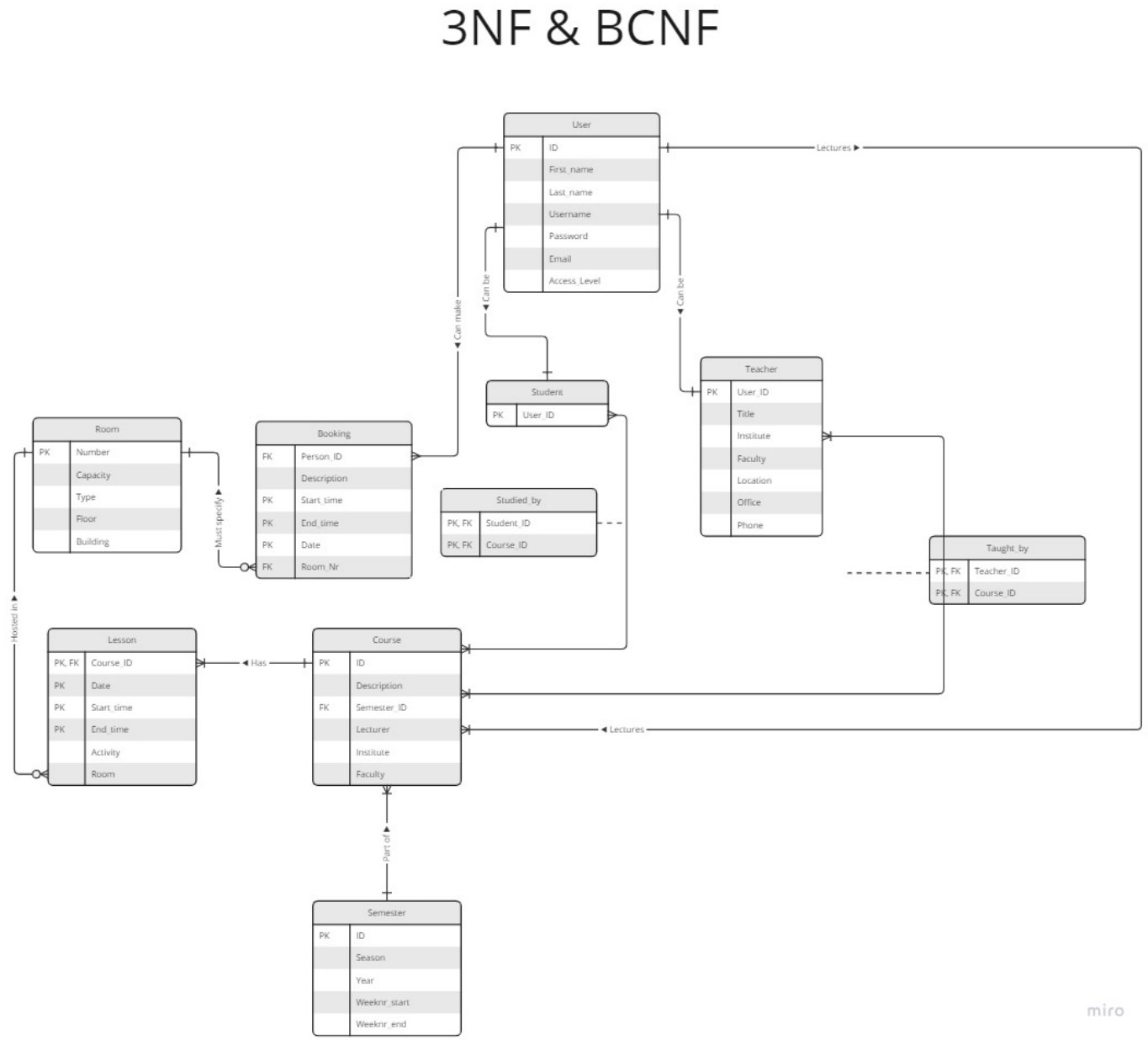
The logical relationship model was not hard to make, as we saw quite clearly what needed to be connected already when designing the entities in the entity relationship diagram. The most difficult part was figuring out which shapes to use for the model, because Miro, the program used to make this, does not have readily the shapes most suitable for this kind of model.

The logical relationship model was made after the final iteration of the entity relationship diagram was made. The model was made by Jan Olav with the assistance of Inga.

Normalization

When making our ERD, we first had problems getting started and visulizing what we wanted, but when we got the first draft, it was already looking really good, and after the peer review, After a careful deliberation, we figured that it was already in BCNF notation, at least according to our research.

This is the normalized (not changed) ERD:



The normalization was done by Inga, while Odin helped.

SQL

The way we handled the SQL part was to use SQL commands to make the whole database with it. We saved all the commands in a file and tested them all togheter in case if the database got messed up in some way by us and for easy recovery. We used alter table after the creation of the tables to make the foreignkeys and used insert to insert data into the diffrent tables. By making the SQL and looking at the 16 SQLQueries that we needed to test on the database we found something we were missing. So after we noticed that we got the Semester table, lecturer in course, connection between lesson and room and some more small things. After wards we began on the Queries when we had insert some data.

The person that worked most on this stage was Daniel where he had some help by Jan Olav. Daniel did the whole sql database and made the the queries. He inserted some data into the database and Jan Olav did that too. Odin was the person who did almost the whole Python file where he got help about the queries errors by Daniel.

Some things we noticed before i begin talking about the 16 Queries. In querie 3, 4 and 10 values that is type "time" is getting displayed atleast in the python got error messages, so number 3,4 and 10 is not excatly as the ones in the python but lik 90% ++ alike. All code displayed under is the 16 SQL queries before implementaion to python endmpoints.

Query 1

A simple query that gets a course, its rooms and buildings if there isn't a lecturer assigned to the course.

```
SELECT Course.ID, Lesson.Room, Room.Building
FROM Course
INNER JOIN Lesson ON Course.ID = Lesson.Course_ID
INNER JOIN Room ON Lesson.Room = Room.Number
WHERE Course.Lecturer IS NULL;
```

```
[
    [
        1,
        101,
        "Main"
    ],
```

This is the result of that query when doing a get request on postman. First value is the unique ID of Course, then room number, then what building.

Query 2

A query that gets the description and season of all courses taught by a specific teacher in a given year and semester.

```
SELECT  c.ID, c.Description, s.Season, s.Year
FROM Course c
JOIN Taught_by tb ON c.ID = tb.Course_ID
JOIN Teacher t ON t.User_ID = tb.Teacher_ID
JOIN Semester s ON c.Semester_ID = s.ID
WHERE t.User_ID = 5
AND s.Season = 'Fall'
AND s.Year = 2022;
```

```
[
    [
        3,
        "Web Development",
        "Fall",
        2022
    ]
]
```


This is the result of that query when doing a get request on postman. The first value is the course unique ID, then Description which is more like the title of the course and then the semester info like semester fall 2022.

Query 3

A query that gets the details of all lessons held in a specific room on a specific date between specific times.

```
SELECT Lesson.Room, Course.ID, Course.Description, Lesson.Lesson_Date,
Lesson.Start_time, Lesson.End_time
FROM Course
INNER JOIN Lesson ON Course.ID = Lesson.Course_ID
WHERE Lesson.Room = 102
  AND Lesson.Lesson_Date = '2023-02-13'
  AND Lesson.Start_time >= '04:00:00'
  AND Lesson.End_time <= '18:00:00';
```


```
[
    [
        102,
        2,
        "Database Systems",
        "Sun, 12 Feb 2023 23:00:00 GMT",
        "14:00:00",
        "16:00:00"
    ]
]
```


This is the result of that query when doing a get request on postman. First value displays room number that is choosen, then which course unique ID and descriptor and then when the lessons that is displayed between the times and on date.

Query 4

A query that gets the details of all lessons held in a specific room on a specific date that overlap with a specific time.

```
SELECT Course.ID, Course.Description, Lesson.Lesson_Date, Lesson.Start_time,
Lesson.End_time
FROM Course
INNER JOIN Lesson ON Course.ID = Lesson.Course_ID
WHERE Lesson.Room = 102
  AND Lesson.Lesson_Date = '2023-02-13'
  AND Lesson.Start_time <= '15:00:00'
  AND Lesson.End_time >= '15:00:00';
```


```
[
    [
        2,
        "Database Systems",
        "Sun, 12 Feb 2023 23:00:00 GMT",
        "14:00:00",
        "16:00:00"
    ]
]
```


This is the result of that query when doing a get request on postman. The firs value is course ID second is The descriptor, and the last info is about when the lesson is.

Query 5

A query that gets the course description, first name, last name, and email of all teachers who teach a course.

```
SELECT Course.Description, Users.First_name, Users.Last_name, Users.Email
FROM Course
JOIN Taught_by ON Course.ID = Taught_by.Course_ID
JOIN Teacher ON Taught_by.Teacher_ID = Teacher.User_ID
JOIN Users ON Teacher.User_ID = Users.ID
```

```
[
    [
        "Web Development",
        "Michael",
        "Brown",
        "michaelbrown@example.com"
    ]
]
```


This is the result of that query when doing a get request on postman. First is the course descriptor then its who the Teacher is with their email last.

Query 6

A query that gets the course description, institute, and faculty of all courses taught by a specific teacher.

```
SELECT Course.Description, Course.Institute, Course.Faculty
FROM Course
JOIN Taught_by ON Course.ID = Taught_by.Course_ID
JOIN Teacher ON Taught_by.Teacher_ID = Teacher.User_ID
WHERE Teacher.User_ID = 4;
```


```
[
    [
        "Web Development",
        "Department of Computer Science",
        "School of Engineering"
    ]
]
```


This is the result of that query when doing a get request on postman. This firstly display the couse descriptor then the institute adn faculty for the course

Query 7

A query that gets the course description, room number, and building of all courses taught by a specific teacher in a specific building.

```
SELECT Course.Description, Room.Number AS Room_Number, Room.Building
FROM Course
JOIN Lesson ON Course.ID = Lesson.Course_ID
JOIN Room ON Lesson.Room = Room.Number
```

```
JOIN Users ON Course.Lecturer = Users.ID
WHERE Users.First_name = 'Emily' AND Users.Last_name = 'Davis';
```


```
[
    [
        "Web Development",
        103,
        "Main"
    ],
```


This is the result of that query when doing a get request on postman. This shows the course room and buildings the a spesified course lecturer has their classes and which course.

Query 8

A query that gets all room numbers and buildings where there are no bookings during a specific time and date range.

```
SELECT Room.Number, Room.Building
FROM Room
WHERE Room.Number NOT IN (
    SELECT Booking.Room_Nr
    FROM Booking
    WHERE Booking.Start_time <= '15:00:00'
    AND Booking.End_time >= '14:00:00'
    AND Booking.Booking_Date = '2023
)
```

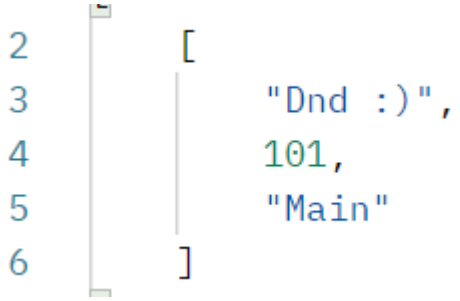

```
[
    [
        101,
        "Main"
    ],
```


This is the result of that query when doing a get request on postman. First value is room number and the scond one is which building it is in.

Query 9

Gets the description of a booking, as well as the room number and building where the booking is taking place, for a given person ID.

```
SELECT Booking.Description, Room.Number, Room.Building
FROM Booking
JOIN Room ON Booking.Room_Nr = Room.Number
WHERE Booking.Person_ID = 3
```

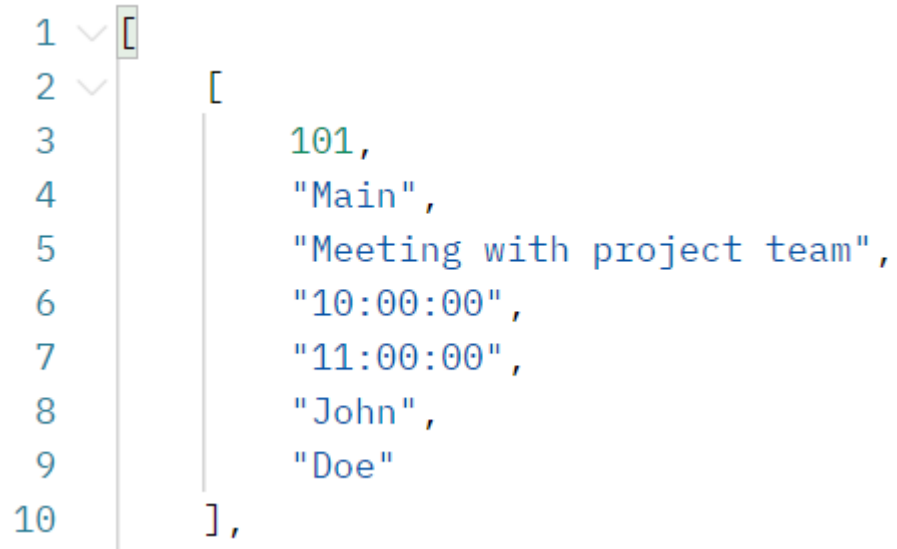


This is the result of that query when doing a get request on postman. The first value is what the "person" has said they booked the room for then which room and building.

Query 10

Gets the room number, building, booking description, start and end times, and the first and last names of the person who made the booking, for all rooms and their bookings. Results are ordered by room number and start time.

```
SELECT Room.Number, Room.Building, Booking.Description, Booking.Start_time,
Booking.End_time, Users.First_name, Users.Last_name
FROM Room
LEFT JOIN Booking ON Room.Number = Booking.Room_Nr
LEFT JOIN Users ON Booking.Person_ID = Users.ID
ORDER BY Room.Number, Booking.Start_time;
```

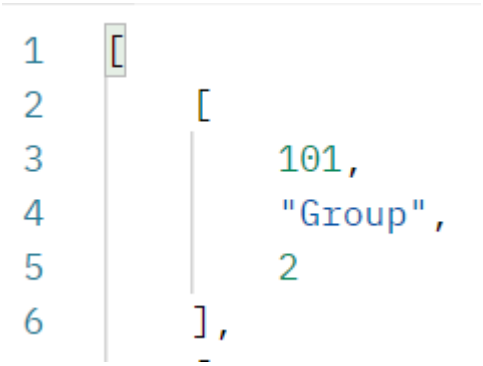


This is the result of that query when doing a get request on postman. First value is which room then which building, the booking message what time and who booked it.

Query 11

Gets the room number, type, and number of reservations for each room. Results are ordered by room number.

```
SELECT Room.Number, Room.Type, COUNT(*) AS Reservations
FROM Room
LEFT JOIN Booking ON Room.Number = Booking.Room_Nr
GROUP BY Room.Number, Room.Type
ORDER BY Room.Number
```



This is the result of that query when doing a get request on postman. First is the Room number, then what type of room it is and the how many bookings

Query 12

Gets the user ID, first name, last name, semester ID, and number of courses taught for each teacher, for all teachers. Results are ordered by the number of courses taught in descending order.

```
SELECT Teacher.User_ID, Users.First_name, Users.Last_name, Course.Semester_ID,
COUNT(Taught_by.Course_ID) AS num_courses
FROM Teacher
INNER JOIN Users ON Teacher.User_ID = Users.ID
INNER JOIN Taught_by ON Teacher.User_ID = Taught_by.Teacher_ID
INNER JOIN Course ON Taught_by.Course_ID = Course.ID
GROUP BY Teacher.User_ID, Course.Semester_ID
ORDER BY num_courses DESC;
```

```
[
    [
        5,
        "Michael",
        "Brown",
        1,
        1
    ]
```


This is the result of that query when doing a get request on postman. First value is the userID of the teacher, then their name and then which semester ID they are having amount of courses.

Query 13

Gets the first name, last name, course description, room number, and building for all courses taught by all teachers, for all teachers. Results are ordered by last name and first name in ascending order.

```
SELECT Users.First_name, Users.Last_name, Course.Description, Room.Number,
Room.Building
FROM Taught_by
JOIN Teacher ON Taught_by.Teacher_ID = Teacher.User_ID
JOIN Users ON Teacher.User_ID = Users.ID
JOIN Course ON Taught_by.Course_ID = Course.ID
JOIN Lesson ON Lesson.Course_ID = Course.ID
JOIN Room ON Room.Number = Lesson.Room
ORDER BY Users.Last_name ASC, Users.First_name ASC
```

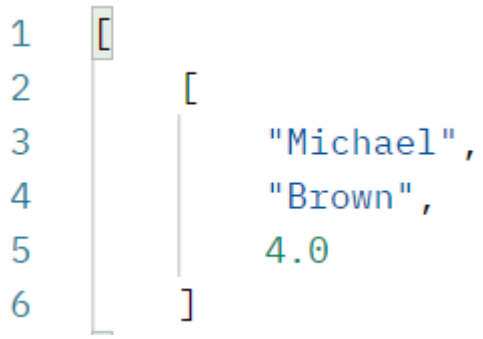

```
[
    [
        "Michael",
        "Brown",
        "Web Development",
        103,
        "Main"
    ],
```


This is the result of that query when doing a get request on postman. First the name of the teacher then the course descriptor and then the Room and number the course have. It continues again with same info diffrent room/building if the course is at multiple rooms.

Query 14

Gets the first name, last name, and total number of hours taught for each teacher, for all teachers. Results are ordered by the total number of hours taught in descending order.

```
SELECT Users.First_name, Users.Last_name,
SUM(TIME_TO_SEC(TIMEDIFF(Lesson.End_time, Lesson.Start_time))/3600) AS Total_hours
FROM Users
JOIN Teacher ON Users.ID = Teacher.User_ID
JOIN Taught_by ON Teacher.User_ID = Taught_by.Teacher_ID
JOIN Course ON Taught_by.Course_ID = Course.ID
JOIN Lesson ON Course.ID = Lesson.Course_ID
GROUP BY Teacher.User_ID
ORDER BY Total_hours DESC;
```



This is the result of that query when doing a get request on postman. Shows teacher then amount of hours spent having lessons.

Query 15

This query retrieves the description of a course, the first and last name of the teacher who taught the course, and their email, for all courses that are taught on Mondays. DAYSOFWEEK is 1 = Sunday, 2 = Monday ... 7 = Saturday

```
SELECT Course.Description, Users.First_name, Users.Last_name, Users.Email
FROM Course
JOIN Lesson ON Course.ID = Lesson.Course_ID
JOIN Taught_by ON Course.ID = Taught_by.Course_ID
JOIN Teacher ON Taught_by.Teacher_ID = Teacher.User_ID
JOIN Users ON Teacher.User_ID = Users.ID
WHERE DAYOFWEEK(Lesson.Lesson_Date) = 2
```

```
[
    [
        "Database Systems",
        "Alice",
        "Johnson",
        "alicejohnson@example.com"
    ],
```


This is the result of that query when doing a get request on postman. Shows the Course descriptor, and then name of teacher and their email

Query 16

This query retrieves the first name and last name of all teachers and their average number of students across all the courses they teach, sorted in descending order by the average number of students.

```
SELECT Users.First_name, Users.Last_name, AVG(Course.nr_students) AS avg_students
FROM Teacher
JOIN Taught_by ON Teacher.User_ID = Taught_by.Teacher_ID
JOIN Users ON Teacher.User_ID = Users.ID
JOIN Course ON Taught_by.Course_ID = Course.ID
GROUP BY Teacher.User_ID
ORDER BY avg_students DESC
```


```
[
    [
        "Michael",
        "Brown",
        35.0
    ]
```


This is the result of that query when doing a get request on postman. Shows the teacher and then the avrage amount of people togheter in the courses that teacher have.