Grouping the aggregated rows and using sub queries

- The learning objectives for this week are:
  - Knowing how the `GROUP BY` clause operates and how it is related to the aggregate functions
  - Knowing what are sub queries and how to use them with the `SELECT` clause

Grouping the aggregated rows

- So, an aggregate function performs a calculation for multiple rows so that the end result is a single value
- If the result table always contains just a single row, how can we write a query such as, "*what's the average grade from each course?*"
- To achieve this, we need to *group* the rows based on a specific column and perform the aggregate function for each group separately
- This can be done using the `GROUP BY` clause

The GROUP BY clause

```
GROUP BY column_list [ HAVING group_filtering_condition ]
```

- The `GROUP BY` clause uses a column or a group of columns in a `SELECT` statement to form groups of rows which the aggregate function operators on:

```
-- what's the average grade from each course?
SELECT course_code, AVG(grade) as average_grade FROM CourseGrade
-- form the groups based on the course_code
GROUP BY course_code
```

The GROUP BY clause

- The result table will have a row for *each distinct column value* of the `GROUP BY` column
- Each row has the corresponding aggregate function result for that group
- In the example's case the result table would contain the average grade for each distinct `course_code`:

course_code	average_grade
a290	2.5
a450	3.0
...	...

The GROUP BY clause

- As mentioned, the `GROUP BY` clause can have multiple columns
- In this case the result table will have a row for *each distinct combination of column values* of the `GROUP BY` columns

```
-- what's the average grade from each course instance?
SELECT course_code, instance_number, AVG(grade) as average_grade FROM CourseGrade
-- form the groups based on the course_code and instance_number
GROUP BY course_code, instance_number
```

The GROUP BY clause

- In the example's case the result table would contain the average grade for each distinct combination of `course_code` and `instance_number`:

```
SELECT course_code, instance_number, AVG(grade) as average_grade FROM CourseGrade
GROUP BY course_code, instance_number
```

course_code	instance_number	average_grade
a290	1	4.5
a290	2	3.0
a450	1	2.9
...	...	...

The GROUP BY clause

- It is worth noting that in the `SELECT` statement we can only select columns that are either aggregate functions or columns used in the `GROUP BY` clause:

```
-- ❌ student_number is not an aggreagate function, nor it is in the GROUP BY clause.
-- This will lead into an error
SELECT course_code, student_number, AVG(grade) as average_grade FROM CourseGrade
GROUP BY course_code
```

- This causes the following error:

> *"Column 'CourseGrade.student_number' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause"*

Combining with the WHERE clause

- We can use the `WHERE` clause to apply filtering *before the grouping is done* by the `GROUP BY` clause:

```
-- how many employees whose salary is above 10000 there are in each department?
SELECT deptno, COUNT(*) AS number_of_employees
FROM Employee
WHERE salary > 10000 -- The WHERE clause is applied before grouping is done
GROUP BY deptno
ORDER BY deptno
```

Using aggregate functions in filtering

- The `WHERE` clause can't use aggregate functions because it is *applied before* the `GROUP BY` clause while the query is executed
- Instead, we can use the `HAVING` clause to filter based on aggregate functions:

```
-- which departments have more than 10 employees?
-- ❌ can't use aggregate functions with the WHERE clause, this won't work
SELECT deptno, COUNT(*) AS "number of employees"
FROM Employee
WHERE COUNT(*) > 10
GROUP BY deptno

-- ✅ we should use the HAVING clause instead
SELECT deptno, COUNT(*) AS "number of employees"
FROM Employee
GROUP BY deptno HAVING COUNT(*) > 10
```