TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

***Database System Concepts for Non-Computer Scientist* - WiSe 24/25**

Alice Rey (rey@in.tum.de)
http://db.in.tum.de/teaching/ws2425/DBSandere/?lang=en

Sheet 06

Exercise 1

Write a query that determines the kind of degree a student is pursuing. In our database, we assume that this can be deduced from the student's semester in the following way: A student who has not reached her 7th semester yet is still considered a "bachelor student". Once in the 7th semester, she should be categorized as a "master student". Starting in the 11th semester, we label her as a "phd student".

Solution:

```
select s.studNr, s.name,
   case
      when s.semester < 7 then 'bachelor student'
      when s.semester < 11 then 'master student'
      else 'phd student'
   end) as degree
from Students s
```

Optional 2

Answer the following questions on our university database using SQL:

a) Find all foundation lectures (lectures that don't depend on another lecture).

b) "Lonely Students" Are there any students that are attending a lecture on their own?

c) "Industrial Students" List all students that are attending all lectures. Hint: [1] [2]

Solution:

a) Query:

```
select *
from lectures l
where not exists (
   select *
   from require r
   where l.lecturenr = r.successor)
```

[1] The task can be rephrased as: List all students, where there is no lecture with no attend record of the student.
[2] Alternatively, you could also try counting how many lectures a student is attending.

b) Solution using a sub query:

```
select distinct s.studNr, name
from Students s, attend a
where s.studNr = a.studNr
and not exists(
   select *
   from attend a2
   where a2.lecturenr = a.lecturenr
   and a2.studnr != a.studnr
)
```

Solution using `group by` and `having`:

```
select distinct min(s.studNr), min(name)
from Students s, attend a
where s.studNr = a.studNr
group by a.lectureNr
having count(*) = 1
```

c) Solution using a double negative: "Find all students where there is no lecture that they did not attend." or "Find all students where there is no lecture, for which there is no entry in attend."

```
select *
from Students s
where not exists
     (select *
      from lectures l
      where not exists(
           select *
           from attend a
           where l.lecturenr = a.lecturenr
           and a.studnr = s.studnr
      )
)
```

Alternatively, we can simply count how many lectures there are in one sub query and then use a second sub query to count how many lectures a student is attending. If the two numbers match (e.g., if there are 5 lectures at a university and a student is attending 5 lectures) then the student is attending all lectures.

```
select *
from Students s
where (select count(*)
       from lectures)
   = (select count(*)
      from attend a
      where a.studNr = s.studNr)
```

Alternatively, we can simply count how many lectures there are in one sub query and then use a second sub query to count how many lectures a student is attending. If the two numbers match (e.g., if there are 5 lectures at a university and a student is attending 5 lectures) then the student is attending all lectures.

```
select s.name, s.studNr
from Students s, attend a
where s.studNr = a.studNr
group by s.name, s.studNr
having count(*) = (select count(*) from lectures)
```

Exercise 3

Answer the following questions on our university database using SQL:

a) Calculate how many lectures each student is attending. Students who do not attend any lecture should be included in the result as well ($attend\_count = 0$) (use outer joins).

b) Figure out how many students each professor knows: A professor knows students from one of their lectures or via a test they have supervised. Include professors not knowing any students and use outer joins. Hint: [3]

[3] Remember that SQL has set operations.

Solution:

a)
```
select s.studNr, s.name, count(a.studNr)
from Students s left outer join attend a on s.studnr = a.studnr
group by s.studNr, s.name
```

b)
```
select p.persNr, p.name, count(p.studNr)
from
((
  select p.persNr, p.name, t.studNr
  from Professors p
    left outer join test t on p.persNr = t.persNr
)
union
(
  select p.persNr, p.name, a.studNr
  from Professors p
    left outer join Lectures l on p.persNr = l.given_by
    left outer join attend a on l.lectureNr = a.lectureNr
)) p
group by p.persNr, p.name
```

Uncorrelated subqueries can be easily transformed into with-statements to make the query more readable:

```
with known_from_tests as (
   select p.persNr, p.name, t.studNr
   from Professors p
      left outer join test t on p.persNr = t.persnr
),
known_from_lectures as (
   select p.persNr, p.name, a.studNr
   from Professors p
      left outer join Lectures l on p.persNr = l.given_by
      left outer join attend a on l.lectureNr = a.lectureNr
),
known as (
   select * from known_from_tests
      union
   select * from known_from_lectures
)
select persNr, name, count(distinct studNr)
from known
group by persNr, name
```

Exercise 4

Find those students who have attended all lectures that they wrote a test in.

Solution:
The requirement that students in the query result should have attended all lectures that they were tested in, can be rephrased as follow: "For a given student, there should be no test/exam, that has no entry in *attend*". This can then be translated into sql easily.

```
select s.*
from Students s
where not exists(select * from test t
                 where s.studNr = t.studNr
                   and not exists
                          (select *
                           from attend a
                           where a.studNr = s.studNr
                             and a.lectureNr = t.lectureNr));
```

This query is an example of a "for all query" where the counting-based technique can not be applied. The reason is that we can not simply count the number of attended lectures, because we need to make sure that the attended lectures match the ones that were tested.
An alternative way that only requires one "not exists" would be to connect the students with their tests and if available add the corresponding attend entry. If there is no attend available, the "left outer join" will leave the "lecture" column empty (adds a "null" value). If we find in our "not exists" subquery an entry where the lecture is null, we can remove

```
with students_tests_optLectures as (
  select s.studnr student, t.lecturenr test, a.lecturenr lecture
  from students s inner join test t on s.studnr = t.studnr
    left outer join attend a on s.studnr = a.studnr and a.lecturenr =
        t.lecturenr
)

select *
from students
where not exists (select * from students_tests_optLectures where
    studnr = student and lecture is null)
```

A second alternative without "not exists" would be to directly search for those students with a null-entry in the with-statement with an additional where clause. The resulting "students_test_woLectures" contains a list of all students that took a test without attending the lecture. Since we are interested in the opposite, we use a set operation to select all students "except" those who took a test without attending the respective lecture.

```
with students_tooktest_didnotattendlecture as (
  select distinct s.studnr
  from students s inner join test t on s.studnr = t.studnr
    left outer join attend a on s.studnr = a.studnr and a.lecturenr =
        t.lecturenr
  where a.lecturenr is null
)

select studnr from students
  except
select * from students_tooktest_didnotattendlecture
```