Database Systems – Autumn 2022
Teacher: Matteo Lissandrini – Exercise Sheet: Advanced SQL
Thanks to: Katja Hose, Gabriela Montoya

AALBORG UNIVERSITET

1 Advanced SQL Queries

Consider the following relational schema, which obviously does not describe the standard situation at Aalborg University.

We assume that tutors are responsible for one or multiple study groups, students individually (not per group) hand in solutions for exercise sheets and receive individual grades in terms of the number of achieved points per sheet. Some of the tutors are more experienced (senior) than others.

student: {[<u>sid: int</u>, firstname: string, lastname: string, semester: int, birthdate: date]}

tutor: {[<u>tid: int</u>, firstname: string, lastname: string, issenior: boolean]}

studygroup: {[<u>gid: int</u>, tid → tutor, weekday: string, room: string, starttime: time]}

exercisesheet: {[<u>eid: int</u>, maxpoints: int]}

handsin: {[<u>sid → student, eid → exercisesheet</u>, achievedpoints: int]}

member: {[<u>sid → student, gid → studygroup</u>]}

1. Determine the ID and the last name of all the different students who have study group more than two days per week. Notice that even if a study group is associated with one week day, a student might have study group more than one day per week if the student is member of more than one study group

Solution.

```
SELECT DISTINCT s.sid, s.lastname
FROM student s
    JOIN member m1 ON s.sid = m1.sid
    JOIN studygroup g1 ON g1.gid = m1.gid
GROUP BY s.sid, s.lastname
HAVING (COUNT(DISTINCT g1.weekday)>2);
```

2. Determine the ID of the tutor who supervises the most students. Notice that if there is more than one tutor that supervises the most students, the query should determine the IDs of all those tutors.

Solution.

```
WITH tutorLoad AS
(SELECT g.tid, COUNT(n.sid) as load
 FROM studygroup g
```

```
    JOIN member n on n.gid = g.gid
  GROUP BY g.tid
)
SELECT tid
FROM tutorLoad
WHERE load = (SELECT MAX(load)
              FROM tutorLoad);
```

3. Find **all the exercise sheet IDs**. For the exercise sheet IDs with at least one handsin, include the on average achieved number of points.

Solution.

```
SELECT e.eid, a
FROM exercisesheet e LEFT OUTER JOIN (
    SELECT eid, AVG(achievedpoints) AS a
    FROM handsin
    GROUP BY eid
  ) AS averages
  ON e.eid = averages.eid;
```

4. Determine the IDs of all students who achieved between 1 and 5 points ($1 \leq achievedpoints \leq 5$) for more than 3 exercise sheets.

Solution.

```
SELECT sid
FROM handsin
WHERE achievedpoints BETWEEN 1 AND 5
GROUP BY sid
HAVING COUNT(*) > 3;
```

5. Determine the IDs and the last names of all students who achieved the maximum number of points for an exercise sheet and the exercise sheet ID.

Solution.

```
SELECT w.eid, s.sid, s.lastname
FROM student s
```

```
    JOIN handsin w ON w.sid = s.sid
    JOIN exercisesheet b ON b.eid = w.eid
WHERE w.achievedpoints = b.maxpoints;
```

2 Recursion

Consider the following tables

part: {[partID, name, cost]}
subpart: {[partID → part, subpartID → part, count]}

A tuple (p1, p2, 3) in the subpart relation denotes that the part with partID p2 is a direct subpart of the part with partID p1, and p1 has 3 copies of p2.

Note that p2 may itself have further subparts.

Please write a recursive SQL query that outputs the names of all subparts of the part with part-id "P-100".

Solution.

```
WITH RECURSIVE totalPart(partid)
AS (
   SELECT subpartID
   FROM subpart
   WHERE partID='P-100'
UNION
   SELECT DISTINCT s.subpartid
   FROM totalPart t, subpart s
   WHERE t.partid = s.partid
)
SELECT p.name
FROM totalPart t, part p
WHERE t.partID = p.partID;
```

or if we want to have P-100 in our list:

```
WITH RECURSIVE totalPart(partid)
AS (
   VALUES('P-100')
UNION
   SELECT DISTINCT s.subpartid
   FROM totalPart t, subpart s
   WHERE t.partid = s.partid
)
SELECT p.name
FROM totalPart t, part p
WHERE t.partID = p.partID;
```

3 SQL – Views

1. What is the difference between (dynamic) views and materialized views?

Solution.

Dynamic view

- *Corresponds to a macro for a query*
- *Result of the query is not precomputed, it is computed whenever the view is used by a query*
- *Computational load during query time*

Materialized view

- *Result is precomputed*
- *If a query uses the view, the result of the view is already computed*
- *Computational load before (e.g., updating time, maintainance time)*

2. Please create a view named "admittedstudents" based on the schema stated above for exercise 1. Please find an appropriate CREATE VIEW statement in SQL. Please include the following points:

   - The view should contain the IDs of all students who achieved at least 50% of all achievable points over all (not only the ones that the student has handed in) exercise sheets (total sums, not each sheet in separate).
   - In addition to the student ID, the view should also have a column showing the percentage of achieved points.
   - The column containing the student IDs should be named "studentID".
   - The column with the percentages should be named "achievedPercentage".

Solution.

```
CREATE OR REPLACE VIEW admittedstudents AS
SELECT s.sid AS studentID, ((SUM(s.achievedPoints)
   * 100) / e.achievablePoints) AS
   achievedPercentage
FROM handsin s, (SELECT SUM(maxpoints) AS
   achievablePoints FROM exercisesheet) AS e
GROUP BY s.sid, e.achievablePoints
HAVING SUM(s.achievedPoints) >=
     0.5 * e.achievablePoints;
```

4 SQL – Spot the Errors

Identify 4 errors that would occur when executing this query (refer to the schema in Exercise 1).

```
SELECT s.sid, s.lastname, AVG(h.achievedpoints),
     (5-5=NULL) AS abcd
FROM student s
    JOIN handsin h ON h.sid = s.sid
HAVING w.achievedpoints >= 10
GROUP BY sid;
```

Solution.

1. *Attribute sid listed in the GROUP BY clause does not unambiguously reference a particular column*
2. *The HAVING clause must not occur before the GROUP BY clause*
3. *The SELECT clause may only list attributes that also occur in the GROUP BY clause or aggregate functions:*
   s.lastname must not be listed here
4. *The HAVING clause references a column that is not listed in the GROUP BY clause nor involved in an aggregate function, correction could be: SUM(w.achievedPoints) – plus "w" is no valid variable, should be "h" instead*

(5-5=NULL) AS abcd is useless but not an error

5 Test your solutions using PostgreSQL

1. Test your solutions to exercise 1 using the database schema and instance available at `https://www.moodle.aau.dk/pluginfile.php/2752733/mod_folder/content/0/ex1-university.sql?forcedownload=1`.
2. Test your solutions to exercise 2 using the database schema and instance available at `https://www.moodle.aau.dk/pluginfile.php/2752733/mod_folder/content/0/ex2-part.sql?forcedownload=1`.