TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Exercises for *Foundations in Data Engineering*, WiSe 23/24

Alexander Beischl, Maximilian Reif (i3fde@in.tum.de)
http://db.in.tum.de/teaching/ws2324/foundationsde

Sheet Nr. 06

Exercise 1 Decorrelate the following SQL query, you can use our WebInterface to test your query:

```
SELECT sum(l1.l_extendedprice)
FROM lineitem l1
WHERE l_extendedprice > (
      SELECT avg(l2.l_extendedprice)
      FROM lineitem l2
      WHERE l2.l_orderkey = l1.l_orderkey);
```

Solution:

```
SELECT sum(l.l_extendedprice)
FROM lineitem l,
     (SELECT avg(l_extendedprice) as avgPrice, l_orderkey
      FROM lineitem
      GROUP BY l_orderkey
     ) precompute
WHERE l.l_extendedprice > precompute.avgPrice
  and l.l_orderkey = precompute.l_orderkey;
```

Exercise 2 Please transform this query into an equivalent query that does not contain correlated subqueries:

```
SELECT o1.o_orderkey
FROM orders o1
WHERE o1.o_totalprice < (
      SELECT avg(o2.o_totalprice)
      FROM orders o2
      WHERE o2.o_shippriority = o1.o_shippriority
         or o2.o_orderstatus = o1.o_orderstatus);
```

Solution:

```
SELECT o1.o_orderkey
FROM orders o1,
     (SELECT avg(o2.o_totalprice) avgprice, o3.o_shippriority,
        o3.o_orderstatus
      FROM orders o2,
           (SELECT distinct o_shippriority, o_orderstatus
            FROM orders
           ) o3
     WHERE o2.o_shippriority = o3.o_shippriority
        or o2.o_orderstatus = o3.o_orderstatus
     GROUP BY o3.o_shippriority, o3.o_orderstatus) precomputed
WHERE o1.o_totalprice < precomputed.avgprice
  and o1.o_shippriority = precomputed.o_shippriority
  and o1.o_orderstatus = precomputed.o_orderstatus;
```

Exercise 3 Solve the queries using SQL based on the university schema. Use this WebInterface for it. By clicking on the button UniSchema you can see the different relations. Use the expanded *examination* relation:

```
WITH examination(MatrNr,CourseNr,PersNr,Grade) as (
    SELECT * FROM pruefen
    UNION
    VALUES (29120,0,0,3.0), (29555,0,0,2.0),
           (29555,0,0,1.3), (29555,0,0,1.0)
)
```

1. Calculate each student's average grade and return it with their name, matrnr and semester.
   Solution:

   ```
   WITH examination(MatrNr,CourseNr,PersNr,Grade) as (
       SELECT * FROM pruefen
       UNION
       VALUES (29120,0,0,3.0), (29555,0,0,2.0),
              (29555,0,0,1.3), (29555,0,0,1.0)
   )
   SELECT s.name, s.matrnr, semester, avg(Grade)
   FROM studenten s, examination e
   WHERE s.matrnr = e.matrnr
   GROUP BY s.name, s.matrnr, semester;
   ```

2. Based on the individual average grade, determine each student's rank within their cohort (students in the same semester).

Solution:

```
WITH examination(MatrNr,CourseNr,PersNr,Grade) as (
    SELECT * FROM pruefen
    UNION
    VALUES (29120,0,0,3.0), (29555,0,0,2.0),
           (29555,0,0,1.3), (29555,0,0,1.0)
),
grades(Name,MatrNr,Semester,Grade) as (
    SELECT s.name, s.matrnr, semester, avg(Grade)
    FROM studenten s, examination e
    WHERE s.matrnr = e.matrnr
    GROUP BY s.name, s.matrnr, semester
)
SELECT *,
    (SELECT count(*) + 1
     FROM grades x
     WHERE x.Semester = n.Semester
       and x.Grade < n.Grade
    ) as Rank_S
FROM grades n
ORDER BY n.Semester, Rank_S;
```

3. Additionally, for each student calculate the difference between their average grade and the cohort's average. (The cohort's average is the average of individual averages.)
Solution:

```
WITH examination(MatrNr,CourseNr,PersNr,Grade) as (
    SELECT * FROM pruefen
    UNION
    VALUES (29120,0,0,3.0), (29555,0,0,2.0),
           (29555,0,0,1.3), (29555,0,0,1.0)
),
grades(Name,MatrNr,Semester,Grade) as (
    SELECT s.name, s.matrnr, semester, avg(Grade)
    FROM studenten s, examination e
    WHERE s.matrnr = e.matrnr
    GROUP BY s.name, s.matrnr, semester
)
SELECT *,
    (SELECT count(*) + 1
     FROM grades x
     WHERE x.Semester = n.Semester
       and x.Grade < n.Grade
    ) as Rank_S,
    (SELECT avg(x.Grade)
     FROM Grades x
     WHERE x.Semester = n.Semester) as GPA,
    (SELECT avg(x.Grade)
     FROM Grades x
     WHERE x.Semester = n.Semester) - Grade
    as difference
FROM grades n
ORDER BY n.Semester, Rank_S;
```

Exercise 4 We will use Postgres for some exercises of the upcoming exercise sheets. Therefore, install the Postgres database, explore it and play around. Here is a nice tutorial explaining everything: Tutorial Postgres.

The Ubuntu way:

```
sudo apt-get install postgresql
```

All other ways are explained on the Postgres installation website.