EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

# Assignment 6

Hand in this assignment until **Thursday, 15 June 2023, 12:00** at the latest.

> ⚠ **Lecture Evaluation**
>
> In the near future, you will have the opportunity to **evaluate lectures** you are attending. With this in mind, we kindly ask you to keep an eye on your inbox and to provide us with **your** valuable feedback. Thank you!

> Ⓔ **xam-style Exercises**
>
> Exercises marked with Ⓔ are similar in style to those you will find in the exam. You can use these to hone your expectations and gauge your skills.

> **Running out of ideas?**
>
> Are you hitting a roadblock? Are some of the exercises unclear? Do you just need that one hint to get the ball rolling? Refer to the #forum channel on our Discord server and check the tag for this assignment —maybe you'll find just the help you need.

## Task 1: About *SQL*
<div align="right">(1 credit)</div>

Please answer the following questions **briefly**:

(a) Ⓔ Consider the following schemata and query:

```sql
1  CREATE TABLE r(d real, e int, f int);
2  CREATE TABLE s(x int, y int);
3  CREATE TABLE t(a real, b text, c text);
4
5  SELECT *
6  FROM   (SELECT r.*, t.a, t.b
7          FROM   r AS r, t AS t
8          WHERE  r.d < t.a) AS r1,
9          s AS r2
10 WHERE   r2.x <> r1.f;
```

What is the *row type* of row variable r1? What is the *row type* of the overall query result?

(b) Explain briefly, why the following queries are invalid.

  i.

```sql
1  CREATE TABLE r(a int, b int, c int);
2
3  SELECT t.b, r.c, t.d
4  FROM   (SELECT r.*, r.a + r.b AS d FROM r AS r) AS t;
```

  ii.

```sql
1  CREATE TABLE r(a int, b int, c int);
2  CREATE TABLE s(x int, y int);
3
4  SELECT r.*, t.y
5  FROM   r AS r, (SELECT s.y
6                  FROM   s AS s
7                  WHERE  s.x = r.a) AS t;
```

iii.

```
1  CREATE TABLE r(a int, b int, c int);
2  CREATE TABLE s(x int, y int);
3  ALTER TABLE s ADD PRIMARY KEY (x);
4
5  SELECT s.a, (SELECT s.y
6                 FROM   s AS s
7                 WHERE  s.x = s.a) AS c
8  FROM    (SELECT r.a, r.b FROM r AS r) AS s;
```

(c) **E** Simplify the following *SQL* queries as far as possible.

i.

```
1  CREATE TABLE r(a int, b int, c int, d int);
2
3  SELECT r1.a, r1.b, r1.c
4  FROM    (TABLE r) AS r1
5  WHERE   true;
```

ii.

```
1  CREATE TABLE r(a int, b int, c int, d int);
2
3  SELECT r1.a, r1.b, r1.c, r1.d
4  FROM    (SELECT t.* FROM r AS t) AS r1;
```
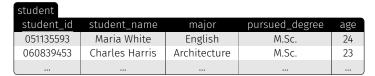
iii.

```
1  SELECT ROW(v.*) :: t FROM t AS v;
```

(d) **E** Consider the following query:

```
1  SELECT r.*,s.* FROM r AS r, s AS s WHERE r.a = s.x;
```

Tables r and s contain |r| and |s| rows, respectively.

i. Without further knowledge, what can you say about the number of rows in the join result?

ii. Now, assume that x is the primary key in s. What can you now say about the size of the join result?

## Task 2: *SQL* University

(1 credit)

In this task you will take a look at a database representing a fictional university and write queries over it to answer some questions.

**student**

| student_id | student_name | major | pursued_degree | age |
|---|---|---|---|---|
| 051135593 | Maria White | English | M.Sc. | 24 |
| 060839453 | Charles Harris | Architecture | M.Sc. | 23 |
| … | … | … | … | … |

**class**

| class_id | class_name | meets_at | room | class_staff_id |
|---|---|---|---|---|
| 1 | Data Structures | MWF 10 | R128 | 489456522 |
| 2 | Database Systems | MWF 12:30-1:45 | 1320 DCL | 142519864 |
| … | … | … | … | … |

**enrolled**

| enrolled_student_id | enrolled_class_id |
|---|---|
| 112348546 | 2 |
| 115987938 | 2 |
| … | … |

**department**

| department_id | department_name |
|---|---|
| 20 | Computer Science |
| 68 | Engineering |
| … | … |

**staff**

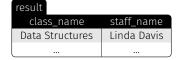| staff_id | staff_name | staff_deptartment | age |
|---|---|---|---|
| 142519864 | Ivana Teach | 20 | 54 |
| 242518965 | James Smith | 68 | 45 |
| … | … | … | … |

To get started, import the schema from the supplied `uni-schema.sql` and then load the data into the tables by running

$$\text{\textbackslash copy } \langle\text{table}\rangle \text{ FROM } \langle\text{file}\rangle \text{ CSV}$$
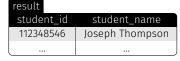
for each `.csv` file. Write the following *SQL* queries using only the *SQL* constructs discussed up to Chapter 6 (*A Diversion into SQL*) of the lecture. The resulting schemas of your *SQL* queries should follow the examples on the right.

(a) List names of all BSc students whose name begins with "Mar". Use **LIKE**[1] to formulate this predicate.

**result**

| staff_name |
|---|
| Maria White |
| … |

(b) For each class, its class name and the name of the teaching staff member.

**result**

| class_name | staff_name |
|---|---|
| Data Structures | Linda Davis |
| … | … |

(c) List all students enrolled in classes of the "Computer Science" department. The result should not contain any duplicates. Is **DISTINCT** necessary to ensure this? Explain.

**result**

| student_id | student_name |
|---|---|
| 112348546 | Joseph Thompson |
| … | … |

(d) Find names of B.Sc. students which are enrolled in classes taught by "Ivana Teach". Draw the join graph for your query as well.

**result**

| student_name |
|---|
| Christopher Garcia |
| … |

(e) Ⓔ List names of staff members who are at least twice as old as some student enrolled in one of their classes.

**result**

| staff_name |
|---|
| Ivana Teach |
| … |

(f) Which classes have both B.Sc. and M.Sc. students enrolled? Draw the join graph for your query as well.

**result**

| class_name |
|---|
| Database Systems |
| … |

> **Note**
> Do not list classes for which only one category of students is enrolled.

---

[1] https://www.postgresql.org/docs/current/functions-matching.html#FUNCTIONS-LIKE

(g) Ⓔ For each class, list its class name, the name of the teaching staff member and the name of their department. Do not forget to list teaching staff members without a designated department.

| result | | |
| --- | --- | --- |
| class_name | staff_name | department_name |
| Data Structures | Linda Davis | Computer Science |
| … | … | … |

> **Note**
> Missing department information shall be represented by **NULL**. Think about using a correlated subquery to determine the department name.