



Assignment 9

Hand in this assignment until **Thursday, 6 July 2023, 12:00** at the latest.

Exam-style Exercises

Exercises marked with **E** 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: GROUP BY and HAVING

(a) Consider this schema

```
1 CREATE TABLE r(a int, b int);
```

and the following two queries¹:

```
1 SELECT r.a, COUNT(*) AS c
2 FROM   r AS r
3 WHERE  r.b = 4
4 GROUP BY r.a;
```

```
1 SELECT r.a, COUNT(*) AS c
2 FROM   r AS r
3 GROUP BY r.a
4 HAVING EVERY(r.b = 4);
```

Create an instance for `r` for which the two queries yield different results. Hand in the DML statements.

Note

The **EVERY** aggregate function is synonymous to **bool_and** which was previously discussed in the lecture.

(b) **E** Rewrite the following query without **HAVING**.

```
1 SELECT r.a, MAX(r.b) AS m
2 FROM   r AS r
3 GROUP BY r.a
4 HAVING COUNT(*) > 2;
```

Note

In the following subtasks, we will be using the university data set, i.e., the schemas and instances from Assignment 6.

(c) **E** Find all M.Sc. students who are enrolled in more than two courses. Make sure your query's output schema follows the table to the right.

name	# courses
...	...

(d) Find the names of all staff members whose courses have fewer than five students enrolled or whose courses are completely unattended. Assume that the staff names are unique. Students enrolled in more than one course should be counted as attending *each* of their enrolled courses. We propose the following **incorrect** "solution":

¹You can find explanations for various aggregate functions in the [PostgreSQL docs](#).

```

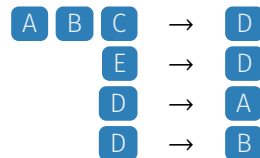
1 SELECT f.staff_name
2 FROM   enrolled AS e, class AS c, staff AS f
3 WHERE  e.enrolled_class_id = c.class_id
4 AND    c.class_staff_id    = f.staff_id
5 GROUP BY f.staff_name
6 HAVING COUNT(*) < 5;

```

Run this query and explain why the query is incorrect. Suggest a correct alternative.

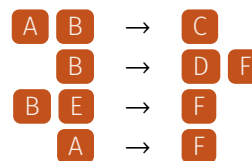
Task 2: Functional Dependencies

(a) Consider the following *functional dependencies* on relation **A B C D E**:



- Calculate the cover $\{\boxed{C}, \boxed{E}\}^+$ based on the four functional dependencies above. List the **intermediate results** of each iteration (i.e., the set X) of the cover algorithm and hand them in.
 - Is the relation in **Boyce-Codd normal form**? Explain briefly.
- (b) Consider the following relation and its functional dependencies.

A	B	C	D	E	F
1	1	?	?	1	1
2	1	2	?	3	?
3	2	3	1	1	1
1	1	1	1	2	?
4	2	2	1	1	1




- Place integer values in all the gaps (?) in the relation without violating the functional dependencies.
- Will the insertion of row

1	1	1	2	1	1
---	---	---	---	---	---

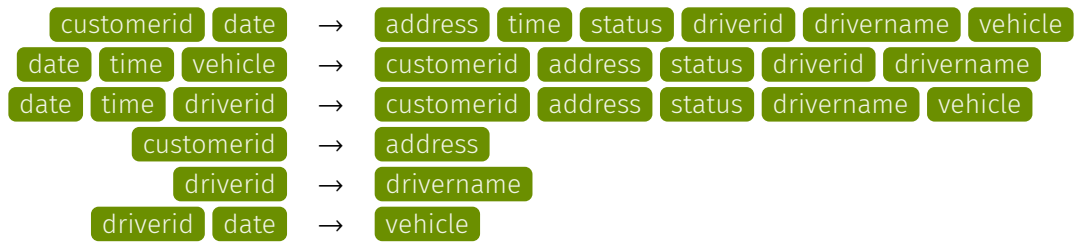
 into the above table violate the given functional dependencies? Explain briefly.

Task 3: Boyce-Codd Normal Form

The table **deliveries** stores information about a logistics company's deliveries to its customers. A delivery is made by a driver using a vehicle. Customers are assigned a unique identifier and their address is stored. For each delivery, the table stores the date, time, and delivery status, as well as the driver's ID and name of the driver and ID of the vehicle. A driver does not change vehicles during the day. A customer receives only one delivery per day, but a driver usually delivers to several customers per day.

 deliveries							
customerid	address	date	time	status	driverid	drivername	vehicle
17	Sand 13, 72076 Tü	19.08.08	17:00	ok	4711	D. Unlop	MB 104
17	Sand 13, 72076 Tü	14.10.08	09:00	no one there	1508	M. Ichelin	FT 8
17	Sand 13, 72076 Tü	29.10.08	11:00	rejected	1508	M. Ichelin	MAN 37
19	Wilhelmstr. 5, 72076 Tü	02.04.08	08:30	ok	1508	M. Ichelin	FT 8
19	Wilhelmstr. 5, 72076 Tü	10.05.08	14:30	complaint	4711	D. Unlop	MAN 37
...

The following functional dependencies hold:



This table (in 1NF) obviously contains quite a lot of redundancy, which can lead to anomalies. Construct an equivalent schema in Boyce-Codd normal form using the decomposition algorithm (Slide 25, Chapter 10 “Functional Dependencies”).

- (a) Hand in the complete split tree produced by the **split** function in plain text format. Make sure to include
- the functional dependency along which the next split is performed,
 - the two relation schemata (keys underlined) resulting from each split, and
 - the set of functional dependencies still applicable in the resulting relations.
- (b) Provide reasonable names for the BCNF relations at the leaves of the split tree.