Heinrich Heine
University Düsseldorf

Institut für Informatik
Datenbanken und Informationssysteme
Prof. Dr. Stefan Conrad
Thomas Germer

WiSe 24/25
Assignment 8
Due December 10, 8:00 AM

Relational Databases and Data Analysis

- As usual, use `zipme.py` to create the archive to upload in ILIAS.
- As in the previous assignments, the tests require that a PostgreSQL database is available.

Exercise 1 *OLAP Operators* (2 + 3 + 1 + 1 + 3 Points)

(a) In `exercise_1_a.py`, give a `GROUP BY` statement which groups by the following combinations of attributes. Use `ROLLUP`, `CUBE` and `GROUPING SETS` (all three).

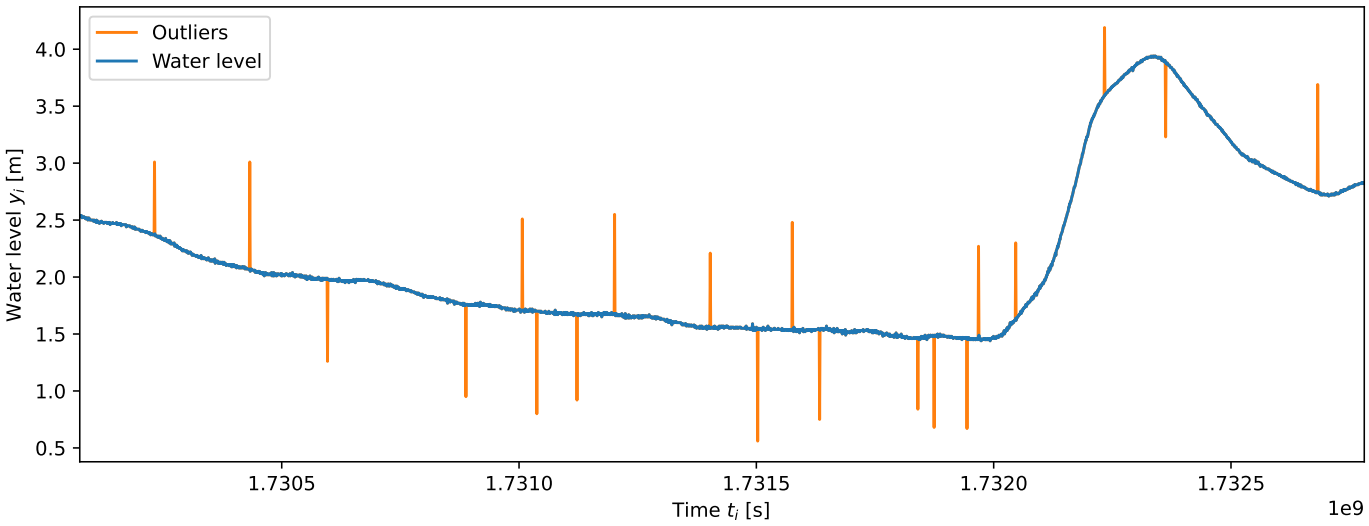
`(A,B,C,D,E)`, `(A,B,C,D,E,F)`, `(A,B,C,D,F,G)`, `(A,B,C,D,G)`, `(A,C,D,E)`, `(A,C,D,E,F)`, `(A,C,D,F,G)`, `(A,C,D,G)`, `(B,C,D,E)`, `(B,C,D,E,F)`, `(B,C,D,F,G)`, `(B,C,D,G)`, `(B,D,E)`, `(B,D,E,F)`, `(B,D,F,G)`, `(B,D,G)`, `(B,E)`, `(B,E,F)`, `(B,F,G)`, `(B,G)`, `(C,D,E)`, `(C,D,E,F)`, `(C,D,F,G)`, `(C,D,G)`, `(D,E)`, `(D,E,F)`, `(D,F,G)`, `(D,G)`, `(E)`, `(E,F)`, `(F,G)`, `(G)`

(b) You are given a table of water level measurements y_i of the river Rhine taken at different times t_i in Düsseldorf.

Measurements

Time t_i [s]	Water level y_i [m]
1730074500.0	2.54
1730075400.0	2.54
1730076300.0	2.53
...	...

The measured data contains outliers, which are undesirable for subsequent processing. Consequently, we want to detect and replace the outliers with more reasonable values **using SQL**.



In `exercise_1_b.py`, return pairs (t_i, y_i^*) where the outliers y_i have been replaced with the average of the two neighboring values $y_i^* = \frac{(y_{i-1}+y_{i+1})}{2}$. Non-outliers should stay the same. A

value is considered an outlier if it has an absolute difference of more than 50 centimeters from the average of the 21 values around it (including itself). For simplicity, we will assume outliers are not within a distance of 10 of each other. You must use the `OVER` keyword at least twice.

(c) The relation `snowfall(sid, snow, date)` records the amount of snow that has fallen for each day. Write a query that returns a list of (`date`, `avg_week`, `avg_month`) tuples where the attribute `avg_week` represents the average snow per week and `avg_month` is the average amount of snow per month for that date. For example, if it only snowed in December, then `avg_month` should be 1 for every day of the month.

(d) Come up with a list of values for the attribute `x` of type `INT` so that the following two queries do *not* return the same result.

```
SELECT       RANK() OVER (ORDER BY x) FROM x_values ORDER BY x;
SELECT ROW_NUMBER() OVER (ORDER BY x) FROM x_values ORDER BY x;
```

(e) In `exercise_1_e.py`, create an example which demonstrates that early grouping can have superior performance. You need to populate a database with appropriate data and then create two queries, one without and one with early grouping. The second query should be faster due to early grouping. Both queries must always return the same result, even if the values in the underlying tables were modified.