EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced SQL
Winter Term 2025/2026
Prof. Torsten Grust, Tim Fischer, Björn Bamberg
WSI — Database Systems Research Group

Assignment 8

Hand in this assignment until **Friday, December 19th 2025, 12:00 pm** at the latest.

🤔 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—maybe you'll find just the help you need.

🎉 The lecture evaluation is over!
A whopping 30% of you have evaluated our Advanced SQL lecture this semester! The results are positive throughout and we are happy that the efforts we pile into this lecture seem to resonate with you. A big thank you to all of, who have taken the time and left us their feedback!

Ⓔ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.

Task 1: Advent of Code

WINDOW FUNCTIONS

Advent of Code (AoC) is an advent calendar with a collection of small programming challenges for the first 12 days in December. Though we are bit behind schedule, in this task you will be solving this year's first challenge in SQL! To get started, go to the description of the challenge 🌐 and *read through it, carefully.*

The input to this challenge consists of a single text file, where each line describes one "rotation" of a dial—`input.txt` depicts an example of such a file. As a tabluar encoding, we propose a table that comprises all rotations in the input, chopped up into semantically relevant bits. Column `stp` holds the position of the rotation in the original file, and columns `dir` and `clks` hold the sign of the rotation and how many clicks it has, respectively.

`input.txt`

L68
L30
R48
L5
R60
L55
L1
L99
R14
L82

A Ⓔ **Complete the insert statement** in `aoc-2025.sql` and make sure that you don't violate any of the `CHECK` constraints we placed on the `input` table.

`input`

stp	dir	clks
1	-1	68
2	-1	30
3	1	48
4	-1	5
5	1	60
6	-1	55
7	-1	1
8	-1	99
9	1	14
10	-1	82

🤓 Remember the lecture!
Recall how we parsed inputs to programming challenges in the lecture (on chapter 4 slide 27)! In addition to the combination of `unnest` and `WITH ORDINALITY` presented there, use the builtin function `read_text` 🌐 to read in the contents of your input file.

B **Write a SQL query** that computes the solution to the first part of the challenge. The solution is a single integer number, so you query should produce a table with exactly this one integer.

🪟 `WINDOW`s are your friends!
The schema you parsed the program input into in Subtask A lends itself perfectly to solving the challenge via a `WINDOW` function! 😉

🎄 Christmas is coming early!
You can gain up to 3 bonus points if you also solve and hand in the second part of the challenge! To gain access to the description of the second part you will need to…

1. …login into the AoC website, *e.g.*, via GitHub,…
2. …download your *personalized* input file from the challenge description,…
3. …run your query from Subtask B on your own input file, and,…
4. …finally, enter the result into the challenge description.

Once you have access to the second part's description, write a SQL query that computes the solution. If you squint your eyes just right, you might see that your solution from Subtask B might not be too far off already. 😉

Task 2: Seesaw Balance

WINDOW FUNCTIONS

A friend of yours wants to display 100 pieces of miscellaneous merch in their room. They have a board that can fit all 100 items, but no saw, no glue, and only one nail, so now they plan to balance everything like a seesaw on that one nail. They have a specific arrangement they want to place the items in, but want to know where to balance the board ahead of time as some of the items are quite fragile.

Putting aside your friends questionable taste in room decor, you offer to help them out if they give you a list of the individual weights of the items in the order they wish to place them on the board. Knowing that you're a SQL-head, they go out of their way to document their arrangement in a table `seesaw` (see the shortend example instance to the right).

`seesaw`

pos	weight
1	1
2	5
3	3
4	4
5	3
6	7
7	6
8	9
9	3

A **Write a SQL query** which finds the *most balanced* pivot position in `seesaw` to hammer the nail into the wall at. For the purposes of this task, you can consider the "most balanced" pivot to be the position in `pos` where the difference of the total weight of all of the items to the left and to right of it is minimal.

> ⚖ Balance by example
> For the shortend example instance `seesaw`, your query should compute `nail`. The most balanced pivot in the instance is `pos` = 6, since it has the minimal weight difference of |16 − 18| = 2.

`nail`

pos	diff
6	2

Task 3: Replace `NULL`

WINDOW FUNCTIONS

The physicists from assignment 3 is back and needs your help again. Back then she was new to the world of RDBMSs, but now she has read up on the topic and found out that she shouldn't be storing multiple measurements in a single row. Instead, she proudly tells you, she now stores all of her measurements in a two-column table and hands you the example instance `measurements` to the right.

She explains: every measurement comprises a unique timestamp `ts` and the measured value `val`. But sometimes the measurements are inconclusive—be it due to a bug in her software or a problem with the sensors—in which cases she documents the values of the respective measurements as being `NULL`.

Now, these `NULL` values are giving her headaches. Handling them in subsequent processing is doable, but not perferrable. Rather she would like to handle this using SQL, but she doesn't know how. Can you help her?

`measurements`

ts	val
2025-12-04 07:34:59	NULL
2025-12-04 07:37:16	42.0
2025-12-04 07:38:36	4.1
2025-12-04 07:42:33	NULL
2025-12-04 07:55:06	NULL
2025-12-04 07:57:06	12.3
2025-12-04 08:03:18	NULL
2025-12-04 08:15:44	15.1
2025-12-04 08:22:21	2.2
2025-12-04 08:37:31	NULL

A **Write a SQL query** which replaces each `NULL` in the `val` column with the *most recent* conclusive measurement preceding it. Since there is no reasonable value to assume, drop any leading inconclusive measurements in your query.

> 🧐 Mission statement unclear?
> If you are unsure what your query is supposed to do, take a look at `result` to the right for what the results for the example instance `measurements` should look like.

`result`

ts	val
2025-12-04 07:37:16	42.0
2025-12-04 07:38:36	4.1
2025-12-04 07:42:33	4.1
2025-12-04 07:55:06	4.1
2025-12-04 07:57:06	12.3
2025-12-04 08:03:18	12.3
2025-12-04 08:15:44	15.1
2025-12-04 08:22:21	2.2
2025-12-04 08:37:31	2.2