**Advanced SQL**
Forum: `https://forum-db.informatik.uni-tuebingen.de/c/ss20-asql`

# Assignment 7

Relevant videos: up to #38
`https://tinyurl.com/AdvSQL-2020`
Submission: Tuesday, 23.06.2020, 10:00 AM

**Please note** that all tasks below are meant to exercise SQL's *window functions*.

1. [15 Points] **Bike Tour**

   We recorded a dataset of waypoints and elevation of a biking tour from Rottenburg to the WSI. The dataset is provided in the file `tour.sql` inside table `tour`. This file also contains a function `distance` which calculates the earth distance between two waypoints in meter. A waypoint is defined through its latitude and longitude.
   Write SQL queries to calculate the following:

   (a) For each waypoint $w$, calculate the distance from home to $w$ as the crow flies, i.e., the direct earth distance between home and the waypoint. The result lists the `id` of $w$ and the **earth distance** between home and the waypoint $w$.

   (b) For each waypoint $w$, calculate the cycled (waypoint to waypoint) distance from home to $w$. The result lists the `id` of $w$ and the **cycled distance** between home and the waypoint.

   (c) For each waypoint $w$, calculate the *estimated* slope at w from the height differences of the waypoints just before/after $w$. The result lists the `id` of $w$ and **slope** in %.

2. [10 Points] **Seesaw Balance**

   We want to figure out where to position the pivot to try and balance a seesaw. As such we provied you with a table `seesaw` populated by 100 random weights uniquely identifiable by their position `pos` $\in (1,2,\ldots,100)$ on the seesaw:

```sql
CREATE TABLE seesaw (
  pos    int GENERATED ALWAYS AS
     IDENTITY,
  weight int NOT NULL
);
```

```sql
INSERT INTO seesaw(weight) (
  SELECT floor(random()*10) AS weight
  FROM   generate_series(1,100) AS _
);
```

   Write a SQL query determining which positions `pos` would be best to balance the seesaw. To balance the seesaw optimally we have to determine the minimal weight difference between the sum of the weights left and right of the pivot. The result of your query produces a table with a column `pos`, which holds the positions at which to place the pivot, and a column `diff`, which holds the difference between the sum of the weights left and right of the pivot placed at `pos`.

   **Simplified Example:**

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

   Let us assume a seesaw with only 9 positions. We find that, at `pos` = 6, the difference between the sum of the weights left and right of the pivot is $|16 - 18| = 2$, which is minimal. The result of the query for this seesaw is therefore:

| pos | diff |
|---|---|
| 6 | 2 |

3. [5 Points] **Replace NULL**

The physicist from assignment 2 needs your help again. Since then, she caught up on various aspects of RDBMS and hands you a measurement table defined as follows:

```sql
CREATE TABLE measurements (
  ts  timestamp PRIMARY KEY,
  val numeric
);
```

She explains: every measure has its own unique `timestamp` and is represented as a `numeric` value. But some measurements were inconclusive and, as such, are represented as `NULL`.

**Example:** A possible set of measurements may look as follows:

| ts | val |
|---|---|
| 2019-12-04 07:34:59 | ☐ |
| 2019-12-04 07:37:16 | 42.0 |
| 2019-12-04 07:38:36 | 4.1 |
| 2019-12-04 07:42:33 | ☐ |
| 2019-12-04 07:55:06 | ☐ |
| 2019-12-04 07:57:06 | 12.3 |
| 2019-12-04 08:03:18 | ☐ |
| 2019-12-04 08:15:44 | 15.1 |
| 2019-12-04 08:22:21 | 2.2 |
| 2019-12-04 08:37:31 | ☐ |

She then asks you to write a SQL query which replaces each `NULL` value with the previous most recent conclusive measurement. She also tells you to drop any leading inconclusive measurements.
For the example above, your query should produce the following result:

| ts | val |
|---|---|
| 2019-12-04 07:37:16 | 42.0 |
| 2019-12-04 07:38:36 | 4.1 |
| 2019-12-04 07:42:33 | 4.1 |
| 2019-12-04 07:55:06 | 4.1 |
| 2019-12-04 07:57:06 | 12.3 |
| 2019-12-04 08:03:18 | 12.3 |
| 2019-12-04 08:15:44 | 15.1 |
| 2019-12-04 08:22:21 | 2.2 |
| 2019-12-04 08:37:31 | 2.2 |