EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced SQL
Sommersemester 2022
Prof. Torsten Grust, Christian Duta
WSI — Database Systems Research Group

# Assignment 8 (01.07.2022)

Handin until: 08.07.2022, 09:00

**Important:** All the queries you write for this assignment most probably will utilize **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 table `tour` of file `tour.sql`. 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* grade of the 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. [8 Points] **Consecutive Logins**

   We define a table `work` as follows:

```sql
CREATE TABLE work (
  emp_id int,
  login  date
);
```

   If row $(e,d)$ is in table `work`, then employee $e$ has been logged in on day $d$. For each employee $e$, find their longest streak of work days, i.e., the **maximum** number of consecutive days they were logged in.

   **Example:**

   In the following set of sample logins, the query results in a table where the employees with `id = 2` and `id = 4` had, at most, 2 consecutive days of logins and the employee with `id = 5` had, at most, 3 consecutive days of logins.

```sql
INSERT INTO work(emp_id, login) VALUES
(2,'2016-04-11'), (4,'2016-04-06'),
(4,'2016-04-07'), (2,'2016-04-06')
(2,'2016-04-07'), (5,'2016-04-07'),
(5,'2016-04-08'), (2,'2016-04-10')
(2,'2016-04-06'), (5,'2016-04-09');
```

| emp_id | streak |
|---:|---:|
| 2 | 2 |
| 4 | 2 |
| 5 | 3 |

3. [7 Points] **Treasure Chests**

You are handed one-hundred coins—either gold or silver—from a huge pile of treasure. You place the coins in one treasure chest until it contains **three gold coins**. Then, you close that chest and open the next (empty) chest to fill with coins. You repeat the process, until every coin is placed inside treasure chests. You can assume an infinite supply of treasure chests.

The following query populates a table `coins` with random coins. The column `id`, when sorted in ascending order, represents the order in which the coins are given to you.

```sql
CREATE TYPE coin AS ENUM('gold', 'silver');

CREATE TABLE coins (
  id   int PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
  coin coin
);
```

```sql
INSERT INTO coins(coin)
SELECT CASE WHEN random() < 0.3
            THEN 'gold'
            ELSE 'silver'
        END :: coin AS coin
FROM generate_series(1,100) AS _;
```

Write a SQL query which returns the treasure chests and the amount of coins they hold.

Example:

In this simple example you are given ten coins which results in two treasure chests with coins.

| id | coin |
|---|---|
| 1 | silver |
| 2 | silver |
| 3 | gold |
| 4 | gold |
| 5 | silver |
| 6 | gold |
| 7 | gold |
| 8 | silver |
| 9 | silver |
| 10 | gold |

| chest_id | # of coins |
|---|---|
| 1 | 6 |
| 2 | 4 |