University of Basel

Prof. Dr. Heiko Schuldt
David Lengweiler, M. Sc.
Martin Vahlensieck, M. Sc.

Introduction to Databases — Autumn 2023

Exercise 4

(Practical) Hand-in: 26.11.2023 (ADAM, 23:59)

(Theory) Hand-in: 26.11.2023 (ADAM, 23:59)

Solving the Exercises: The exercises can be solved in small groups of a maximum of two people. Use the notations introduced in the lecture. The DMI plagiarism guidelines apply for this lecture.

Submission Information: Please upload all deliverables BEFORE the deadline to ADAM as **a single PDF** using the team hand-in feature. Solutions that are handed in too late cannot be considered. For practical exercises upload the deliverables to ADAM and present them to one of the assistants/tutors during the exercises, both is required to recieve the points!

Task (Practical) 1: SQL Queries (9 points)

For this exercise you will have to answer various questions about the dataset from exercise 3 using the SQL syntax introduced in the lecture. **There are many dialects of the SQL standard (like the one of PostgreSQL), to solve the exercise follow the standard syntax introduced in the lecture unless specified.** Every subtask should be solved using a *single* SQL statement. **Read the tasks carefully and only return the required columns.** If you need more than one statement, the subtask will say so. *Each correct substask is rewarded with 0.5 points. You can collect a maximum of 9 points.*

Aggregation

(1) List all different values, that appear in the `result` column.

(2) For each of the tuples of *(1)*, count how often it appears.

Joining Tables

(3) Create a list of all `eco_codes` and how many openings are associated with it.

(4) Each game opening has a specific `eco_code`, count how often each `eco_code` has been played and order it by the most played first.

(5) Retrieve all games played in a tournament where white was the winner. The list has to include the player names (white and black), the tournament URL, and the titles of the players. *Hint:* Watch out the title columns can be `NULL`, take those games into consideration as well.

String Matching

(6) How many of the openings contain the word "Queen" in their name?

(7) What is the longest opening name with the word "Queen"? *Hint:* You can use the `LENGTH(`*`string`*`)` function to count the amount of characters in a string.

(8) How often was the opening from the previous question played?

(9) What is the longest sequence of your first name (starting at the beginning) that can be found within the names of any of the players (ignoring the casing)? *Hint:* In PostgreSQL you can use the non-standard `ILIKE` operator as a case-insensitive variant of `LIKE`.

DDL & DML

(10) The `titles` table only lists the abbreviation of the title, but not the long name. Find out what the abbreviations stand for, and add them to the table as a new column. This will require multiple statements. *Hint:* We got the data set from Lichess.

(11) For completeness, add an entry for "Grand Master" with abbreviation "GM".

(12) You realize that the two references to the `players` table (`white` and `black`) might not be the most practical and decide to create a new table which links `players` and `games` called `plays_in`. It should contain a reference to both `games` and `players` in addition to an `is_white` field of type `BIT(1)` which is the bit 1 if the player played as white, else 0. The bit 1 would be written like this: `b'1'`. Add additional constraints, such as `NOT NULL`s and foreign keys, and ensure that there can only ever be two entries (one for black, one for white) per game.

(13) Fill the table using a *single* `INSERT` statement.

Additional Queries

(14) Use the table `plays_in`, which you created in the previous task, to find the player that played the most games. Assume that no player ever plays himself.

(15) Which player has won the most games?

(16) List all players that have played each other in at least 100 games.

(17) List the ten players with the highest win percentage. Only consider players with more than 100 games. As the SQL syntax to do so is not covered in the lecture, you will need to do this with a script.

Adding a Title Column

(18) After you successfully created your new table `plays_in` you realize that you want to include the titles of the players during the game into the `plays_in` as well. Do so without recreating the table. In the first step add the additional column `title_id`. Then fill the column with the correct values. As the SQL syntax to do so is not covered in the lecture, you will need to do this with a script.

(19) Retrieve each title a player ever had. Omit players, that never had any title. Your result should contain abbreviation of the title, the id of the player and the name of the player.

Views

(20) Create a view called `games_and_names` with the following columns: `game_id` the game id, `white_name` the name of the white player, `black_name` the name of the black player and `result` the result of the game.

(21) Write a view called `games_in_tournament` with two columns: the `tournament_id` and how many games were part of it.

(22) Use this view to compute the minimum, average and maximum games per tournament.

Hand-In: A single plain **text** file with all the statements must be submitted on ADAM until the deadline of the **(Theory)** part. Make sure to clearly annotate to which subtask a query belongs to.

Task (Theory) 2: JOIN (1 points)

customer	(customer_id, customer_name)
orders	(orders_id, order_date, customer_id)

You are presented with the following SQL query, which uses a `LEFT JOIN` to link the `orders` and the `customers` tables:

```
SELECT
    orders.order_id,
    orders.order_date,
    customers.customer_name
FROM
    customers
INNER JOIN
    customers ON customers.customer_id = orders.customer_id;
```

Adjust this query so it does not longer us a `JOIN` operator but still retrieves the same tuples. Shortly explain your logic, which your new query uses (max. 2 sentences)?