CS2102-2310: Database Systems

Assignment 1

Deadline: Friday, 29 September 2023, Time: 11.59 pm

Instructions

- The assignment consists of 10 half-mark SQL questions; thus a total of 5 marks. This also means that there will be no partial marks for individual queries.
- The deadline for submission is September 29 (Friday) at 11.59 pm.
- **Late submission penalty:** One mark will be deducted for each late day up to two late days; submissions after the second late day will receive zero marks and will not be graded.
- The assignment is to be submitted using Examplify.
- As the assignment will be auto-graded, it is very important that your submitted answers conform to the requirements in the Instructions (Section 1).

Good Luck!

1 Instructions

In this assignment, you will formulate 10 SQL queries. As the assignment will be auto-graded, it is very important that your submitted answers conform to the following requirements:

- Each submitted query must be a single `SELECT` statement.
- Your queries must be executable on PostgreSQL – that is, each query must be a syntactically valid SQL query without any typographical errors; a SQL answer that is syntactically invalid or contains very minor typographical error will receive 0 marks even if the answer is semantically correct.
- Your results must not contain any duplicate records.
- Your queries should not require any additional tables or views.
- Your queries must return the correct results for this year's Tour database and for previous and future Tour databases with the same schema; see additional details in Section 3.
- You are encouraged to explore alternative solutions using different constructs (simple, algebraic, aggregate, and nested queries, for instance, no unnecessary table, no unnecessary `DISTINCT`, `ORDER BY`, etc.) as such additional requirements may appear in midterm questions.
- To help you test and debug your solutions in PostgreSQL, we will provide a Web interface where you can check the correctness of your query. More details can be found in Section 4.

Before you Submit. Before you submit your queries, check (a) if all queries are syntactically valid SQL, and (b) if all queries are only using the tables of the given schema. If you have created views to help with finding the solutions, make sure that your final queries do not require them.

Submission via Examplify. We will create and provide an Examplify assessment for submitting your queries. The assessment will consist of 10 questions that will require you to copy-&-paste the corresponding query into a text box. Make sure to paste each query in the correct text box, as well as to include all the queries. We will provide the Examplify assessment and more details closer to the deadline.

2 Database

2.1 Overview

The database is about the Tour de France 2023. The Tour de France is an annual professional road cycling race that takes place primarily in France, although it occasionally passes through neighboring countries. It is one of the most prestigious and challenging events in the cycling world. The race is divided into a series of stages, with each stage being a separate race in itself. The overall winner of the Tour de France is the rider who completes all the stages in the shortest total time. There are various classifications within the Tour de France, including the General Classification (GC), which determines the overall winner, and additional categories like the Points Classification (for sprinters), the King of the Mountains Classification (for climbers), and the Young Rider Classification (for riders under a certain age). We already provided you with a more detailed description on Canvas.

2.2 Download

All the data in the database was collected directly from the official website of the Tour de France. To download the database and to import it into your local PostgreSQL installation, we provide with 4 `.sql` files to make this process easy:

- `01-tdfdb-clean.sql`: Drops all tables from the database.
- `02-tdfdb-schema.sql`: Create all tables in the database.
- `03-tdfdb-data.sql`: Inserts all rows/tuples into all tables.
- `04-tdfdb-check.sql`: Checks if all data was been correctly inserted.

If you have any questions about the database (e.g., the meaning of individual columns), you can post your questions to the Canvas Discussion for Assignment 1.

3 Questions

Before writing queries, please take note of the following guidelines:

- Since the data has been collected from the official website of the Tour de France, you can use this site to validate the correctness for some of the queries. However, the data may not 100% accurate and complete. For example, queries regarding exact points or timings might vary from the official results on the website. Disclaimer: some information in the handout and some data in the database may not be exact or complete. The queries may not reflect the Tour de France 2023 official results.
- Write queries that can be used for another instance of the Tour with the same rules but different results. Although there are always 21 stages in the Tour de France, do not use this number as a constant in your queries. In short, do not use constants for values that can be derived from the data!
- Each question indicates the expected result schema. For example, a question requires you to return the `rider_name` and `team_name` for each result row, your query should return these to columns w.r.t. their name and order. As such your query is likely somewhere contain:

```
[...]
SELECT ... AS rider_name, ... AS team_name
[...]
```

So, let's get started...

Q1: Which riders ride for teams of their own country? For example, *Victor Lafay* is from France and rides for the French time *COFIDIS*, so he should be in the result set. Return each riders `bib`, `name`, `team_name`, and `country`.

Q2: Which stages started and ended in different countries? For example, Stage 10 started and ended in France, and should therefore be not in the result set. Return the `stage_nr` and the codes of the country where the stage started (`cc_start`) and ended (`cc_end`)!

Q3: Which stages featured the highest mountains? For each stage, return the `stage_nr` and the `max_height` of the tallest mountain. Sort the result from the highest to the lowest `max_height`.

Q4: Which stages ended with a mountain finish? For example, Stage 6 finished on a mountain. Note that category 'H' represents the highest category. Return the `stage_nr`, as well as the `category` and `percent` value for the mountain. Sort the result by `category` (descending) and `percent` (descending).

Q5: How many "flat" stages are followed by a "hilly" stage? For example, the "flat" Stage 7 is followed by the "hilly" Stage 8, so it needs to be considered in the count. Return the result as column `num_stages`!

Q6: What is the final ranking? Important:

- Consider riders race times incl. the bonus and penalty seconds as both are relevant for the ranking!
- Consider riders who have finished the tour!

Return the `name` and the `total_time` in seconds for each top-10 rider, and sort the result w.r.t. total time in ascending order! You can assume that there are no ties regarding the total time.

Q7: What was the average speed of each rider? The average speed of a rider is calculated by dividing the total distance cycled during the tour by the total time taken. Important:

- You do not need to exclude riders who finished some stages but abandoned the tour before its end.
- You do not need to consider the one rider who did not even finish the first stage.

Return the `name` and `avg_speed` for each rider, sort the result w.r.t. the average speed in descending order.

Q8: Who was the current leader of the Sprint Classification after each stage? The leader is the rider with the most sprint points after that stage. Hint:

- You can check the Wikipedia page for the TDF '23 – cf. "Classification leadership by stage", green column – for the expected result.
- Your result will not exactly match with the one from Wikipedia!

Return the `race_day` of the stage and the `name` of the rider of that stage, sorted by the day in ascending order!

Q9: On what days were all teams represented in the top 50 of the day's stage rankings? In other words, on what `days` did all teams had at least one of its riders in the top-50 for that stage. Return the `race_day` of each stage!

Q10: Which team won the "Best Team" award? The winning team is the team with the lowest aggregate time using the sum of the three best riders' times. Hints:

- Check PostgreSQL
  `ROW_NUMBER() OVER (PARTITION BY <column> ORDER BY <expression> ASC|DESC)`
- The bonus and penalty seconds of riders are relevant for this ranking.

Return the `team_name` of the winning team!

4 Check Your Solution

To help you test and debug your answers in PostgreSQL, we will provide a Web interface where you can check the correctness of your queries. But please keep the following things in mind:

- Please use this interface only to test the correctness of a query **after** you have written and run the query using your local PostgreSQL installation of the database!
- If you think that your solution is correct, but the check (partially) fails, please send us an email with your solution so we can clarify if maybe the question can be misinterpreted or our reference solution is flawed.
- For the marking, we will use a modified version of the database. So do not "hard-code" your questions. For example, do not write `"SELECT 176;"` instead of `"SELECT COUNT(*) FROM riders;"` if you query asks you to return the number of riders.

4.1 Web Interface

You can currently find the Web interface via two URLs:

- http://172.26.191.108/courses/cs2102/sql/ – fast but needs SoC VPN.
- http://185.170.113.47/courses/cs2102/sql/ – public but slow.

To test the correctness of a query, you need to do the following steps:

- Select the query you want to check using the drop down box (e.g., Query 1)
- Copy your solution for the query into the text box.
- Click "Check Query" and wait for the report.

The screenshot in Figure 1 shows an example for a report. The correct answer for the test query `'Query 0'` is:

```
SELECT bib, name
FROM riders
WHERE country = 'ITA';
```

Feel free to use and modify the query to see how the changes affect the results of the report. For example, in the screenshot, the correct attribute `name` has been renamed to `dummy` to illustrate the warning that the returned attribute names do not match the expected attribute names.

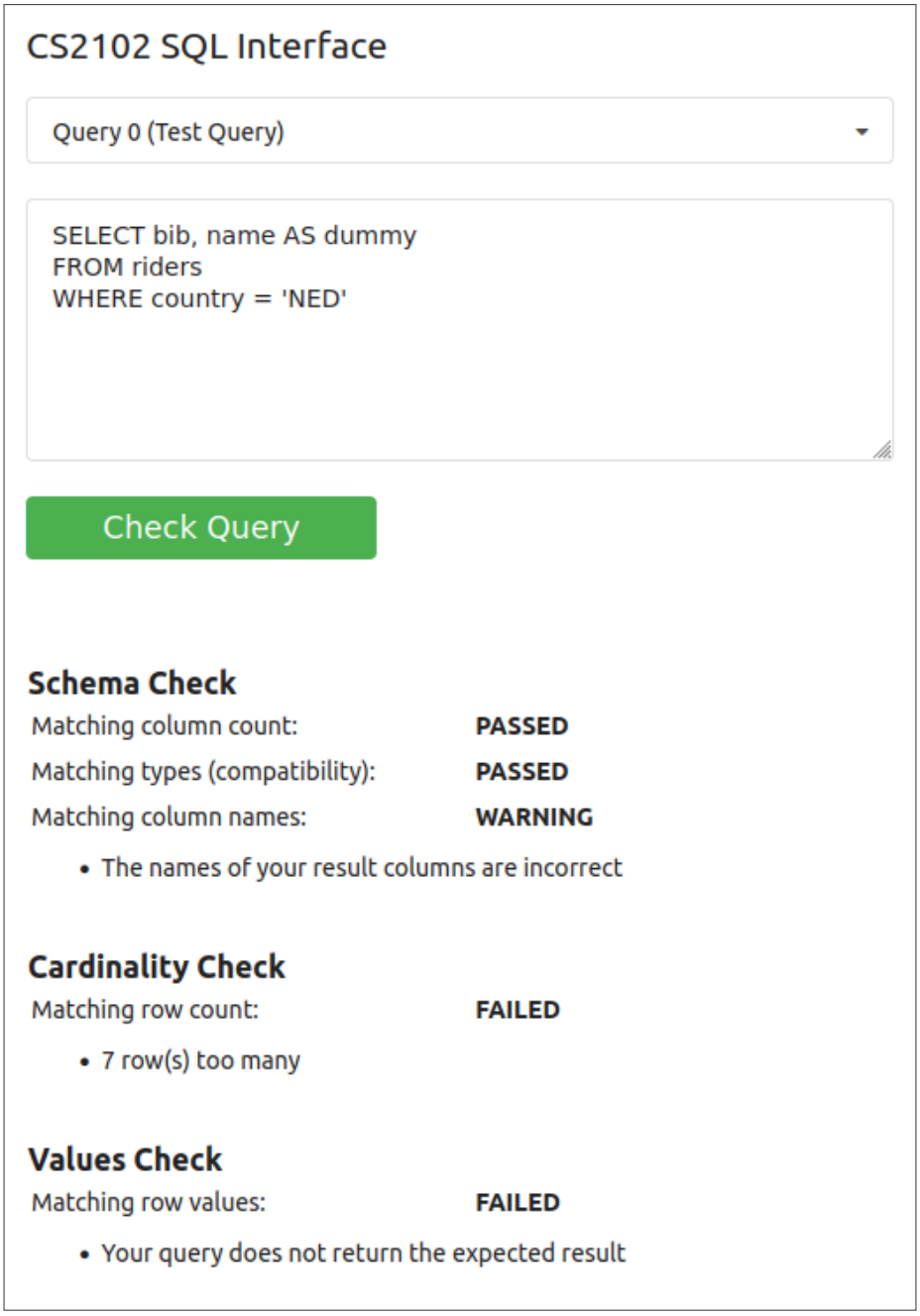

Figure 1: Screenshot of SQL Web Interface to check your queries.

4.2 Description of Report

The report includes 3 types of checks:

- **Schema Check:** The Schema Check will check if your solution and the reference solutions have "comparable" schemas. This means:
  - Both schemas have the same number of columns.
  - Both schemas are union-compatible (i.e., the respective columns have comparable data types).

  - Both schemas should have matching column names (note this will throw only a warning as the final column names will be enforced when creating the view)

- **Cardinality Check:** The Cardinality Check will check if your solution and the reference solutions have the same number of output rows/tuples. If the cardinalities do not match, the report will indicate how many rows/tuples are missing.
- **Values Check:** The Values Check will check if your solution and the reference solutions return the same result.

Note: The checks are not independent. For example, if the Schema Check fails because the number of columns do not match, the Values Check will naturally also fail. So try to address any failed checks in a meaningful order.