

Problem Sheet 2 - Intermediate SQL

George Melrose: george_melrose@yahoo.com, <https://github.com/georgemelrose>

Introduction

These problems aim to test your intermediate SQL knowledge, building on the basic SQL concepts tested in problem sheet 1. The questions and solutions are of a more esoteric nature than problem sheet 1 yet still useful as a SQL coder. For the purposes of this series of problem sheets, a database of dummy Marathon results data has been generated. More information on the **Marathon** database is presented below.

The concepts tested in this sheet are covered by the LinkedIn learning course **Intermediate SQL for Data Scientists** - (<https://www.linkedin.com/learning/intermediate-sql-for-data-scientists/>) .

Useful Preparatory Resources

In addition to this problem sheet, there are two useful resources you can draw upon to better understand these SQL concepts:

- **Two RMarkdown documents** - one to generate some dummy ‘Universities’ data (https://github.com/georgemelrose/SQL_Practice/blob/main/0_generating_databasestar_dummy_data.Rmd). This was copied from the excellent SQL learning resource databasestar (https://github.com/bbrumm/databasestar/tree/main/sample_databases/sample_db_university/sqlite). The other document is an RMD HTML going over intermediate SQL concepts and how they can be applied to databasestar dummy data (https://github.com/georgemelrose/SQL_Practice/blob/main/03_Intermediate_SQL_for_Data_Scientists.html).
- **A video presentation** - a recording of a meeting in which I presented the **Intermediate SQL for Data Scientists** HTML , explaining varying higher level concepts- (https://universityofcambridgecloud.sharepoint.com/sites/AD_Progress/SitePages/Learning-SQL-in-a-New-Format.aspx).

Marathon Database

Firstly, the data to be put into the Marathon database was formulated from the following Python script - (https://github.com/georgemelrose/SQL_Practice/blob/main/Dummy_Marathon_Data/marathon_data_generation.ipynb).

The **marathon data generation** python script generates the following tables:

1. **Runners** - Randomly generate 1000 runners with names common in their locale/country, together with their birth date and sex.
2. **Events** - The 6 Major World marathons (Berlin, Boston, Chicago,London,New York City, Tokyo), with an event per year from 2012 to 2023.
3. **Results** - Gives results for runners in hh:mm:ss format, ensuring there aren’t duplicate results for each runner per event. Prevents any results breaking either the male marathon world-record (2:00:35 Eliud

Q3. - Create a composite index on the runner_id and event_id columns in the Training_Plans table to improve performance for queries that frequently search for a specific runner's training plan for an event?

Solution -

```
CREATE INDEX idx_training_plans_runner_event ON Training_Plans(runner_id, event_id);
```

Q4. - Add a unique index on the first_name, last_name, and date_of_birth columns in the Runners table to ensure no duplicate runner profiles are entered, even if runners have similar names?

Solution -

```
CREATE UNIQUE INDEX idx_runners_name_dob ON Runners(first_name, last_name, date_of_birth);
```

Q5. - Give the code to see all the indexes present?

Solution -

```
SELECT name FROM sqlite_master WHERE type = 'index';
```

Q5. - Give the command to see all the indexes associated with a particular table, for example 'Runners'?

Solution -

```
PRAGMA index_list('Runners');
```

Views

Q1. - Create a view of all runners from the Baltic States? **Hint** - most of the code is already present in problem sheet 1.

Solution - Cr