# 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 Prepatory 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).

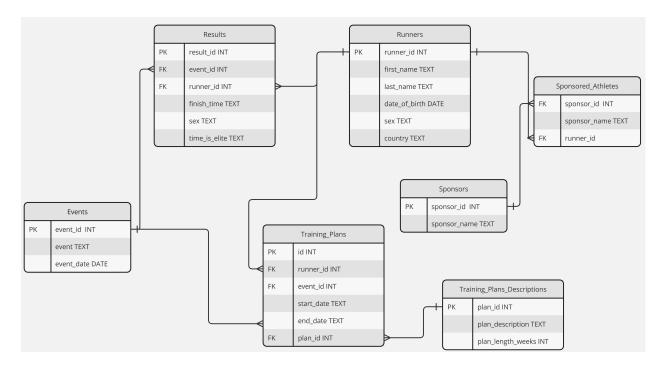
The marathon data generation python script generates the following tables:

- 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

Kipchoge 2023) or the female marathon world-record (2:11:53 Brigid Kosgei 2019). Also determines, with a True/False column, if a result is elite by the male standard (below 02:15:00) or the female standard (below 02:30:00).

- **4. Sponsors** Lists the following 10 major companies that typically act as sponsors to runners "Nike", "Adidas", "Asics", "Saucony", "Hoka", "Brooks", "New Balance", "Puma", "Under Armour", "Tracksmith".
- **5.** Sponsored Athletes A table listing the fraction of the elite athletes that have a sponsor.
- **6.** Training Plans Descriptions The descriptions of 10 different training plans and their respective lengths in weeks.
- 7. Training Plans The training plans of athletes. Only 72% of runner-event combinations have an associated training plan.

## Marathon Database Entity Relationship Diagram



# Intermediate SQL Problems

### **Indexes**

Q1. - Create a basic index on the runner\_id column in the Results table to speed up searches and joins involving runner information?

### Solution -

### CREATE INDEX idx\_results\_runner\_id ON Results(runner\_id);

**Q2.** - Construct a unique index on the event\_id column of the Events table to ensure no duplicate event entries exist?

# Solution -

# CREATE UNIQUE INDEX idx\_events\_event\_id ON Events(event\_id);

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