

# SQL CoP Problem Sheet 1 - Basic Query Statements

George Melrose: gam55@cam.ac.uk (Research Information Office)

## Introduction

These problems aim to test your basic SQL knowledge, steadily building up in complexity. The questions and solutions are common ones you will come across when querying different datasets. 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 **SQL Server Fundamentals: Master Basic Query Techniques** - (<https://www.linkedin.com/learning/sql-server-fundamentals-master-basic-query-techniques>) .

## 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\\_databasesstar\\_dummy\\_data.Rmd](https://github.com/georgemelrose/SQL_Practice/blob/main/0_generating_databasesstar_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](https://github.com/bbrumm/databasestar/tree/main/sample_databases/sample_db_university/sqlite)). The other document is an RMD HTML I generated walking you through basic SQL concepts and how they can be applied to this databasestar dummy data ([https://github.com/georgemelrose/SQL\\_Practice/blob/main/01\\_Basic\\_Query\\_Statements.html](https://github.com/georgemelrose/SQL_Practice/blob/main/01_Basic_Query_Statements.html)).
- **A video presentation** - a recording of a meeting in which I presented the **Basic Query Statements** HTML , explaining key SQL concepts - ([https://universityofcambridgecloud.sharepoint.com/sites/AD\\_Progress/SitePages/Learning-SQL-in-a-New-Format.aspx](https://universityofcambridgecloud.sharepoint.com/sites/AD_Progress/SitePages/Learning-SQL-in-a-New-Format.aspx)). This video can be found on the aforementioned page under the **SQL and R** title. *Note that this can only be viewed by SQL " Community of Practice Members, email gam55@cam.ac.uk for access.*

## 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](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).

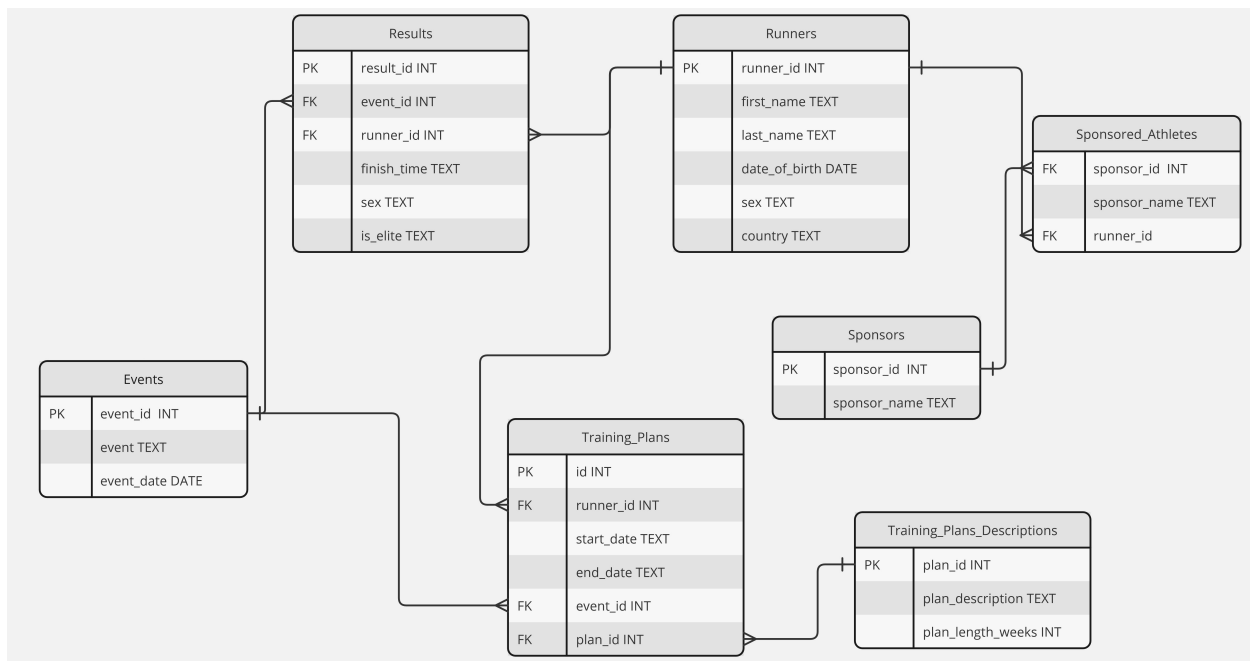
**4. Sponsors** - Lists the following 10 major companies that typically act as sponsors to runners - "Nike", "Adidas", "Asics", "Saucony", "Hoka", "Brooks", "New Balance", "Puma", "Under Armour", "Tracksmith".

**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.

## Marathon Database Entity Relationship Diagram



## Basic Query Statement Problems

### Single Table 'Select' Statements

**Q1.** - Get all the tables in the database, inspect their layout, and get the first 5 rows of each table?

**Solutions** -

**.tables** - to get all the tables.

**.schema *table*** - to get the schema of a particular table.

**select \* from *table* limit 5** - to get the first 5 rows of a particular table.

**Q2.i.** - Obtain the full names and countries of the runners?

**Solution** -

**select first\_name || ' ' || last\_name as full\_name, country from runners;** The || symbols concatenate the first\_name and last\_name columns into one. The **as** operator allows a column to have a different name displayed in the query result.

**Q2.ii.** - Find the top 10 Marathon results by time, male or female?

**Solution -**

**select \* from Results order by finish\_time asc limit 10;** The **order by** operator orders the results by whichever column is specified. The **asc** operator orders the results in ascending order, from the highest to the lowest time.

## Filtering on Single Conditions

**Q1** - Find the top 20 male marathon results and the top 20 female marathon results?

**Solution -**

**select \* from Results where sex= 'Male' order by finish\_time asc limit 20;**

**select \* from Results where sex= 'Female' order by finish\_time asc limit 20;**

The **where** clause filters rows based on specified conditions, in the above cases the different sexes.

**Q2.i.** - Find all the runners from Lithuania, Latvia, and Estonia?

**Solution -**

**select \* from Runners where country in ('Lithuania', 'Latvia', 'Estonia');**

Using the **in** operator, rows fulfilling a list can be found.

**Q2.ii.** - Find all the runners *not* from Poland, Czechia, Slovakia or Hungary?

**Solution -**

**select \* from Runners where country not in ('Poland', 'Czechia', 'Slovakia', 'Hungary');**

Using **not** in conjunction with the **in** operator, rows not fulfilling a list can be found.

**Q3.i.** - Find all the runners whose names begin with 'J'?

**Solution -**

**select \* from Runners where first\_name like 'J%';**

Using **where** clause in conjunction with the **like** operator, rows with a first name beginning with J can be found. Note that in string matching, 'string%' find a pattern at the beginning of a string. '%string' means finding a pattern at the end of a string.

**Q3.ii.** - Find all the runners whose names don't with 'J'?

**Solution -**

**select \* from Runners where first\_name not like 'J%';**

The answer is exactly the same as in part i. albeit with the **not** operator to signify everything *not* fulfilling the condition.

**Q3.iii.** - Find all the runners whose surnames contain 'son'?

**Solution -**

**select \* from Runners where last\_name like '%son%';**

In this solution, all the matching rows that *contain* the son string regardless where it is in the names are found by wrapping the string in % signs in the query code.

**Q4.i.** - Gather all training plans that have a length between 10 and 12 weeks?

**Solution** -

```
select * from Training_Plans_Descriptions where plan_length_weeks between 10 and 12;
```

The **between** operator together with the required numerical parameters and the **and** operator

**Q4.ii.** - Gather all training plans that have a length less than 12 weeks?

**Solution** -

```
select * from Training_Plans_Descriptions where plan_length_weeks <12;
```

The less than < operator after the **where** clause fetches results below a certain numerical threshold, 12 in this case.

**Q4.iii.** - Gather all training plans that have a length more than 12 weeks?

**Solution** -

```
select * from Training_Plans_Descriptions where plan_length_weeks >12;
```

The greater than > operator after the **where** clause fetches results above a certain numerical threshold, 12 in this case.

**Q4.iv.** - Gather all training plans that have a length more than or equal to 12 weeks?

**Solution** -

```
select * from Training_Plans_Descriptions where plan_length_weeks >=12;
```

The greater than or equals to >= operator after the **where** clause fetches results above or equal to a certain numerical threshold, 12 in this case.

## Filtering on Multiple Conditions

**Q1.i.** - Find all female runners from the United Kingdom born in the 20th century?

**Solution** -

```
select * from Runners where sex = 'Female' and country = 'United Kingdom' and date_of_birth > 01-01-2000;
```

The greater than > operator is used, converse to normal logic, to find dates *less than* 1st January 2000.

**Q1.ii.** - Find all male runners from Brazil born in 1997?

**Solution** -

```
select * from Runners where sex = 'Male' and country = 'Brazil' and date_of_birth between '1997-01-01' and '1997-12-31'
```

In this solution, despite the date column having the format yyyy-mm-dd we can still whittle the results down to the year. Using the **between** and **and** operators, together with the date range we seek, those rows fulfilling the question's conditions can be found.

**Q2.** - Find all runners whose names begin with the letters G or J?

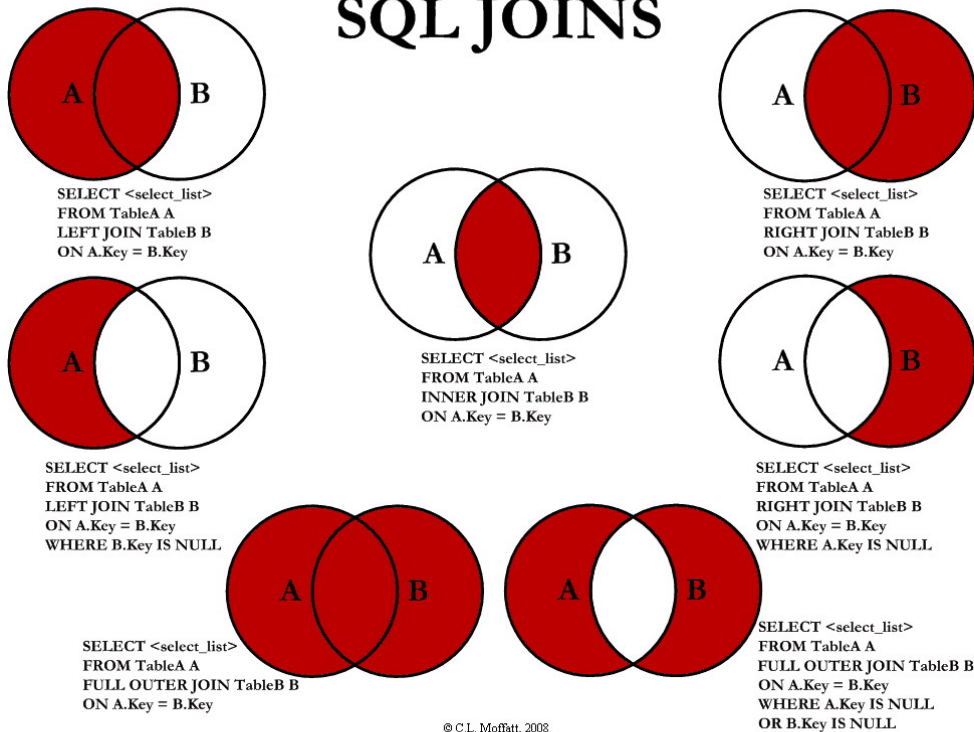
**Solution** -

```
select first_name || ' ' || last_name as full_name from Runners where first_name like 'J%' or first_name like 'G%';
```

As in the 'Filtering on Single Conditions' section, string matching and the **where** operator are used to obtain rows fulfilling these multiple conditions.

## Single Inner Joins

# SQL JOINS



**Q1.** - Obtain the full names of all 'elite' runners?

**Solution -**

```
select first_name || ' ' || last_name as full_name from Runners inner join Results on Runners.runner_id = Results.runner_id where is_elite = 'True';
```

Utilising the **concatenation operator** || again, the full name of each runners is obtained. A single inner join with the Results table allows for a filter on the condition of 'is\_elite' to be made.

**Q2.** - Find the the following information for runners of any of the Boston Marathon events in the database: runner\_id; finish\_time; sex; is\_elite?

**Solution -**

```
select runner_id, finish_time, sex, is_elite from Results inner join Events on Events.event_id = Results.event_id where event = 'Boston Marathon';
```

The 4 necessary columns are selected from the Results table, which is then joined on to the Events table by the column they have in common - event\_id. Filtering specifically down to the Boston Marathon events is done by using a **where** clause.

## Multiple Inner Joins

**Q1.** - Obtain the full names of all 'elite' runners as well as their finish times, position, and sex? Additionally, gather the information on the events when these elite times were made?

Solution -

```
select first_name || ' ' || last_name as full_name, finish_time, position, sex from Runners inner
join Results on Runners.runner_id = Results.runner_id inner join Events on Results.event_id
= Events.event_id;
```