# DT228/2, DT282/2 Databases I
## Joins and Functions

## Objectives

The objectives of this lab are to:

- To become more familiar with the SELECT statement
- To become familiar with using joins in the SELECT statement to traverse a physical database
- To use functions to format the output and within the where clause.

## Contents

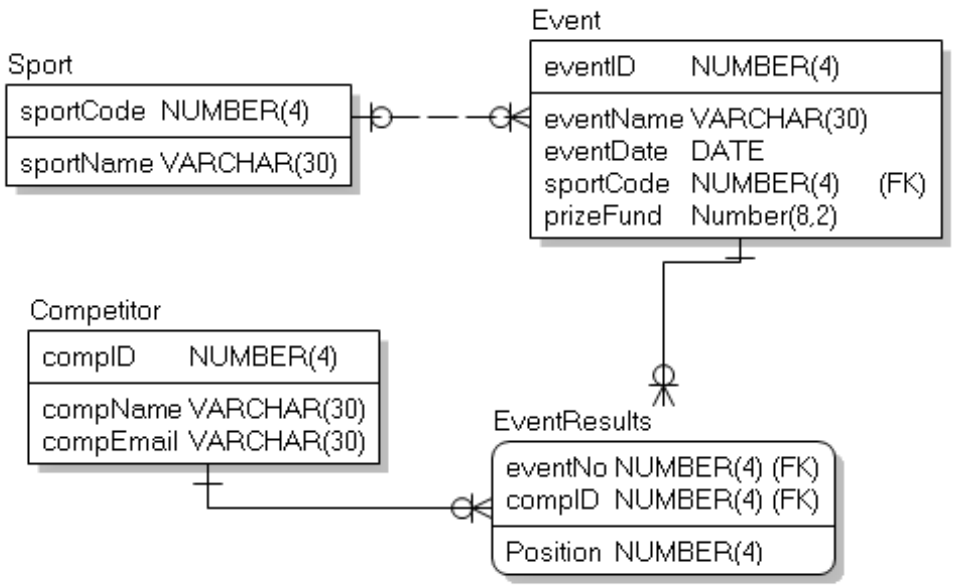
## 1. Create your data structures

Download DT228-DT282-Week6Lab.sql from Webcourses and run it in SQL Developer.

This will create the tables for the data model shown in the ERD below. It is the model created in the labs in week 4 to store details on sporting events scheduled in a sports competition, the competitors competing in these competitions and the positions these competitors finished in that competition with some small changes to the model and a little bit of extra data.

So if Usain Bolt, email address UB@jam.com, is competitor number 1, finishes 1st in the 100m Mens final on 16th Aug 2016, event 1, finishing in position 1, in the sport Athletics, sport 1, there will be a row in the SPORT table with a sportCode of 1 and a sportName Athletics; a row in the EVENT table event ID 1, eventName 100m Mens Final, eventDate 16 AUG 2016, a sportCode of 1 plus the amount of prize money available in prizeFund; a row in the COMPETITOR table with compID 1, compName Usain Bolt, compEmail UB@jam.com; and then a row in the EVENTRESULTS table with the eventNo 1, compID 1 and position 1.

At the end of the script are a number of select statements which will show you the data included in each table.



# 1. Using Functions

1. Find the names of all events and output the name in uppercase, lowercase and with an initial capital letter;

2. Find all competitors with a letter u at position 4 of email address.
   Hint: use SUBSTR

3. Output the email addresses of all competitors but replace usa by America in the email address in the output.
   Hint: use replace

4. Output the names and dates of all events with a prize fund of 250,000. Output the prize fund with a dollar symbol and formatted as $999,999.99 and the date formatted as DD/MM/YYYY.
   Hint: use to_char

5. Using a case statement print out the results for each event. Include the event ID, the competitor ID and instead of the position print First, Second or Third.
   Hint: use CASE

6. Output the names and prize funds of all events. If a prize fund is null then rather than outputting null output 0.
   Hint: use NVL

7. Output the names and instead of the prizefund if the prizefund has a value output 'Prize set' otherwise output 'No Prize'.
   Hint: NVL2

# 2. Joins

1. Find the names of all competitors who finished in third position in their events include the position in the output.
   Hint: Join competitor and eventResult. From should be eventResult.

2. Change the SQL for 1 to include the eventName.
   Hint: include a join to EVENT but you can't use using.

3. Change the SQL so that it reads as a sentence
   Competitor competitor x finished in 3rd position in event x on DD Month YYYY.

   Where DD Month YYYY is the event date.

4. Using a case statement change the SQL for 3 so that it considers all results and position print First Place, Second Place or Third Place depending on the position.
   Sort the output in ascending order of position.
   Hint: use CASE and || for concatenation

5. Output for all competitors who have a letter u in the 4 position of their email address their name, the name of the events they competed in and the position they finished in those events.

## 3. Grouping

1. Output the number of events in each sport. Include the sportcode in the output.

2. Change the SQL for 1 to include the sportname in the output

3. Find the average finishing position for each competitor. Include the competitor ID in the output.

4. Change the SQL for 3 to include the competitor name in the output.