University of Sussex Autumn 2018
Informatics

Databases
Assignment 2 (Deadline 6.12.18, 4pm)

The coursework assigned below must be submitted online as Canvas E-submission *by 4pm on Thursday, 6th of December 2018. The submission must consist of the single file* `a2.sql` *that you download from the submission point in Canvas and into which you insert your code for each question after the corresponding question comment. You have to answer* all 14 *questions.*

You must work on this assignment on your own. The standard Informatics rules for collusion, plagiarism and lateness apply. Any cases of potential misconduct discovered will be reported and investigated.

Detailed Instructions (follow carefully)

This assignment refers to an implementation of the motorsport database as designed in the the first assignment. To be able to answer the questions of this second assignment you must first run the SQL script `a2-setup.sql` that defines the tables that your code will rely on. For the completion of this assignment it will be helpful to inspect the table structure set up by this script. Do not modify the structure of the tables when you write your answers unless explicitly told to do so.
Note that a few data records have been inserted into the tables to help you test your answers. It is recommended that you test your code with additional sample data you insert into the tables yourself. However, *do not include any of the test data or the corresponding insert statements in your submission*. Also, you must *not* include the code of `a2-setup.sql` in your answer.
In the following, SQL always refers to the MySQL 5.1 dialect and all your code must run *on our ITS server* where it will be tested for marking purposes. Make sure you get the expected results on *on our ITS server*. If you test it on other servers you might get different behaviour. Don't forget that in MySQL table names are case sensitive.
You must not deviate from the requested order and name of the columns in queries. Note that for every query the names and order of columns is clearly specified in the paper. Any change will most likely confuse the testing script and cost you marks.

Each question does only allow one SQL statement as answer. Copy this statement *directly below* the corresponding question comment, e.g. `--@@01` if you answer Question 1, on a new line in the template file. Below is an example where to put your answer yo Question 1 and Question 3 and how to leave Question 2 unanswered:

```
-- @@01

SELECT myanswertoQ1 FROM mytable WHERE 1=1

-- @@02

-- @@03

SELECT myanswertoQ3 FROM mytable WHERE 1=1

-- @@04
```

Do not remove any of the template comments as they drive the testing rig. Where a query is very complex, you might wish to add comments to help the marker appreciate what you have done in case your query is not correct.

Marking Guidance

- Marking will be mainly driven by testing your code.
- Statements that are not parsing, i.e. throw a syntax error, receive 0 marks!
- Statements that throw a runtime error when tested will receive an automatic penalty of half the available marks after the runtime error has been removed manually by the marker (in potentially a random fashion) and testing the resulting query as your answer.
- Correctly running statements will receive marks proportional to how close their result is to the correct answer.
- For *every* violation of the instructions (anywhere in this paper) that has to be manually fixed, a penalty of 2 marks is deducted. So don't insert any code that is not an answer to a question, do not delete any of the comments as they drive the test rig.

There are *14 questions* which you will find on the following pages.

1. Write *one* SQL statement to set up table *MoSpo_HallOfFame* according to the following Relational Schema:

   MoSpo_HallOfFame(hoFdriverId, hoFYear, hoFSeries, hoFImage,
   hoFWins, hoFBestRaceName, hoFBestRaceDate)
   primary key (hoFdriverId,hoFYear)
   foreign key (hoFdriverId) references MoSpo_Driver(driverId)
   foreign key (hoFBestRaceName,hoFBestRaceDate) references
   MoSpo_Race(raceName,raceDate)

   Your code *must* execute *without error*, assuming that all other tables have been set up by running script `a2-setup.sql`. [14 marks]

   The data types you choose for the columns should be most appropriate for the data they will contain. You must also accommodate the following requirements:

   (a) For table and column names you *must* pick exactly the names used in the schema above (otherwise you will lose marks as tests will fail).

   (b) *hoFYear* is a 4-digit number representing a year between 1901 and 2155 (or 0000).

   (c) *hoFSeries* is one of the following strings: *BritishGT*, *Formula1*, *FormulaE*, *SuperGT*. Please make sure you use the correct spelling. These column values, when ordered, should always appear in the order they have been listed above. Values for this attribute must not be missing.

   (d) *hoFWins* is a positive integer number and never larger than 99. The default is 0 but values can be missing.

   (e) *hoFImage* is a path to an image document which is a string never longer than 200 characters. This value can be missing.

   (f) Equip any foreign key constraints with constraint names of your choosing.

   (g) Ensure that if a driver is deleted from the database their corresponding hall of fame entries are deleted automatically too.

   (h) Ensure that if a race is deleted from the database then foreign key values in hall of fame entries that reference it are automatically set to null.

Instructions for Question 2–13

For each of the tasks specified below write *one single* SQL query, respectively, that solves the task. You can use nested queries (ie. subselects and subqueries) wherever you like.

You must not `CREATE` any tables of any form and you must not use (nor declare) any stored procedures or functions in this section.

You *must produce column headings as specified for each query*. *Do not change order or name of the columns as this will cause tests to fail which will cost you marks*.

Double check that headings are *exactly* as specified.

It is important that your queries will work correctly with *any* data (according to the schema).

All references to time, when not explicit, are relative and refer to the time of running the query.

2. The weight of drivers has been omitted from the *MoSpo_Driver* table. Without deleting and recreating the table, add a column *driverWeight* to the already created table that allows values to be missing.

   Take into consideration that a driver's weight is always in the range 0.0 to 99.9. [4 marks]

3. Change the postcode of the racing team *Beechdean Motorsport* to (the following string) HP135PN. [5 marks]

4. Remove all drivers with last name *Senna and* first name *Ayrton* (whatever the capitalisation) from the database. [5 marks]

5. Find out how many racing teams are on the database. The heading must be `numberTeams`. [3 marks]

6. List all racing drivers (driver id, name and dob) whose last name begins with the same letter as their first name. The name of the driver should be given as a string consisting of the initial from the first name, followed by a blank, followed by their last name. So a driver with first name *Alan* and last name *Turing* would be listed as *A Turing*. The headings must look like this: [6 marks]

```
driverId driverName driverDOB
```

7. List for each racing team how many drivers they have associated with them. Only include teams with more than one driver. The headings must look like this: [6 marks]

```
teamName numberOfDriver
```

8. For each race list the fastest lap time. The information provided should include race name, race date, lap time. No races must appear for which there is no proper such minimal time available. The headings must look like this: [7 marks]

```
raceName  raceDate  lapTime
```

9. Given a race (name) and a year, 'total pitstops' is the total number of pitstops of all cars in the given race that year. For each race name compute the average of the number of 'total pitstops' based on the years we have data for. The headings must look like this: [7 marks]

```
raceName avgStops
```

10. A car (of a race entry) retires in a lap if the corresponding attribute `lapInfoCompleted` has value 0. Find out all the (different) makes of cars that had to retire in a race in the year 2018. The heading must look like this: [7 marks]

```
carMake
```

11. For each race, compute the highest number of pitstops any car had. Provide race name and date as well as the highest number of stops. Races with no pitstops recorded at all should appear with a 0. The headings must look like this: [8 marks]

```
raceName  raceDate  mostPitstops
```

12. List all drivers (id, last name) who had no retirement ever. Note that the reason for not having had a retirement may well be that the driver never participated in a race.

    The headings must look like this: [7 marks]

```
driverId driverLastName
```

13. For any given care make m and time period t, let *RetirementsRate*$_m(t)$ be the total number of retirements of cars of make m divided by the total number of cars of make m taking part in a race during time t. In case no car of make m participated in race r during period t this number is undefined (NULL).

    For example, let t be the year 2000 and $m = Porsche$. Assume that in the year 2000 there were two races with *Porsche* cars involved. In the first race 2 cars of that make raced and 1 had a retirement. In the second race 3 cars of that make raced with 0 retirements. Therefore, we get that *RetirementsRate*$_{Porsche}(t) = \frac{1}{5} = 0.2$.

    For a period t, let *AverageRetirementRate*(t) be the *average of retirement rates for period* t *across all makes* m, i.e. the average of *RetirementsRate*$_m(t)$ ignoring undefined values, over all makes m.

    List for each car make m the retirement rate *RetirementRate*$_m(t)$ where t is the current year. Only select car makes m with a retirement rate *above* the average retirement rate across all makes for the same period t, i.e. where *RetirementRate*$_m(t) >$ *AverageRetirementRate*(t). The headings must look like this: [9 marks]

```
carMake  retirememententRate
```

Additional Instructions (Stored Procedures) Question 14
For developing answers to Questions 14 you can use any delimiter you like (e.g. `$$`). But do *not* declare the delimiter in the submission file and *remove the delimiter symbol from the end of your routine declaration* in the submission file. This is important for the automatic testing.

Note that successfully declaring a stored procedure does not necessarily mean it runs without error. You need to run and test your procedures to ensure that. Strictly name the stored procedure as indicated in the question. You are not allowed to include any other stored routine definitions.

14. Write a stored function `totalRaceTime` that, given a racing number, the name of a race, and the date of a race, returns the total race time for the car specified by the racing number in the given race. If the given race does not exist, the routine should throw the error *procedure Race does not exist*. If the specified racing number did not take part in the existing race, the routine should throw an error *procedure RaceEntry does not exist*.

    In the case that not *all* required lap times for the (existing) car in the (existing) race are available either until race finish or retirement, the routine should throw the error *procedure TimeForAllLaps does not exist*.

    If the (existing) race was not completed by the (participating) car in the race due to retirement but all lap times were available until retirement, the routine must not throw an error but return *null*.

    The error handling should be implemented as explained in Lecture 15. Note that in those error cases the function must not return a string but produce an SQL error.

    The total race time should be returned as an integer denoting milliseconds. Note that this stored routine has three arguments and you must declare them in the order given above. [12 marks]