DATABASE SYSTEMS
Coursework 3

Dr Paolo Guagliardo

Released: Tue 15 November 2015 – **Due: Fri 25 November 2016 at 16:00**[*]

Before you begin, please read carefully the policy on Late coursework and Academic misconduct and related links, in particular the rules for the publication of solutions to coursework.

Database schema. We will use the schema available at

`<dbshome>/assignment3/schema.sql` (MD5: `c3489e269e569c4db80714c09098d9b8`)

where `<dbshome>` is `/afs/inf.ed.ac.uk/group/teaching/dbs`. You don't need to import this file into your PostgreSQL database, as the schema will be included in the test instances. In addition to the constraints explicitly mentioned in the schema file, you can further assume the following:

- the value of attribute *track_number* in each row of table Tracks is less than or equal to the value of attribute *tracks* in the corresponding row of table Albums;
- every row in table Albums has at least one corresponding row in table Tracks.

Assignment. Write the following queries in SQL.

(01) Find artists who have only released **studio** albums. Artists who have not released any albums are not included in the output. Return the artist's name.

(02) Find albums with a higher rating than *every* previous album by the *same* **band**. Return the album's title and artist.

(03) Find **live** albums released by British artists (i.e., whose country is `GBR`) and having a higher rating than the average rating of all albums released (by any artist) in the same year.

(04) Find tracks shorter than 2 minutes and 34 seconds from albums rated 4 or 5 stars and released after 1995. Return the album's title and artist and the track's number.

(05) Find albums with missing tracks. Return the album's title and artist and the number of missing tracks. Albums without missing tracks are not included in the output.

(06) Find the average total running time of all albums released in the '90s with at least 10 tracks and where no tracks are missing. Round the average to the nearest integer and return two integer columns: one for the number of minutes and one for the number of seconds.

(07) Find artists whose combined number of **live** and **compilation** albums exceeds the number of **studio** ones. Return the artist's name.

[*]The submissions will be collected on Tuesday 29 November at 23:59. Coursework that has been submitted after the "official" deadline up until the collection time will be considered (without any penalty). There is **no exception** after that time, unless you have been granted a coursework extension.

(08) Find the number of US (i.e., country is `USA`) **bands** whose debut album (or at least one of them, if there are multiple ones) is rated 5 stars.

(09) For each artist, find the percentage p of their albums rated less than 3. Artists who have not released any album are not included in the output. Artists who have only released albums rated 3 or more are included in the output with $p = 0.00$. Return the artist's name and the corresponding percentage p as a number between 0 and 100 with 2 decimal digits.

(10) Two studio albums are *consecutive* if they are released in different years, say Y_1 and Y_2, and there is no **studio** album released strictly between Y_1 and Y_2. Find artists who have *never* released two consecutive studio albums more than 4 years apart. Artists who have released less than 2 studio albums in total do satisfy this condition and must be included in the output. Return the artist's name.

Submission instructions.

- Each query must be written in a text file named `<xx>.sql` where `<xx>` is the two-digit number in the list of queries above. For example, the first query will be written in file `01.sql` and the last one in file `10.sql`. **The character encoding of the file must be ASCII; if you use UNICODE your queries will fail.**
- Each file consists of a single SQL statement, terminated by semicolon (`;`). Submitted files that do not contain *exactly one semicolon* will be discarded when the submission is processed and consequently they will not be assessed. **Please pay attention to this; even if it looks like a trivial detail, it is not.**
- If a file contains more than one statement, or a statement that is not a query (such as `CREATE VIEW` or `UPDATE`), it will fail. Do not use comments either, as they may cause a correct query to fail if they contain semicolons.
- Submission is via the `submit` command on DICE.[1] You can submit all your files as follows:

  ```
  submit dbs 3 01.sql 02.sql 03.sql 04.sql 05.sql 06.sql 07.sql 08.sql 09.sql 10.sql
  ```

  You can also submit your files one by one, or in smaller groups, and in any order. For example:

  ```
  submit dbs 3 03.sql
  submit dbs 3 01.sql
  submit dbs 3 04.sql 02.sql
  ```

  Files can be submitted more than once, in which case the previously submitted version will be overwritten (`submit` will ask you whether you want to proceed).
- Before submitting your files, you should check that they run smoothly in PostgreSQL on DICE[2] with the command `psql -h pgteach -f <file>.sql`

Assessment. Your queries will be executed on 5 database instances *without nulls.* Each query is worth 10 marks, which are allocated as follows:

- 2 marks are given for each test database on which the query returns *all and only* the correct answers.
- 1 mark is given for each test database on which the query returns *at least 50%* of the correct answers (but not all of them) and *no wrong answers.*

The queries will not be assessed for their performance or style, as long as they terminate after a "reasonable" time; each query should not take more than a few seconds to run.

[1] See how to submit a practical exercise.
[2] See how to use PostgreSQL on DICE machines.

Test data. One of the 5 instances on which your queries will be assessed is publicly available at:

- `<dbshome>/assignment3/data/db1.sql` (MD5: `c8cdac6a6c0f2517804b6bb3e9a83bc3`)

You can load this instance into your PostgreSQL database on DICE with the following command:

```
psql -h pgteach -1 -f <dbshome>/assignment3/data/db1.sql
```

One additional test instance will be made available soon.

Query answers. The answers your queries are expected to return on the test database are available in CSV format at:

- `<dbshome>/assignment2/answers/db01/`

`01.csv`	(MD5: `e8c4249dd940c721567b159599d52f24`)
`02.csv`	(MD5: `113b2e6b82cefb7fe20b4df6053f802f`)
`03.csv`	(MD5: `4e989f36965a1b99e4bea9f8ed36b3b7`)
`04.csv`	(MD5: `27acc95f34ab538d0a011a3242ba986d`)
`05.csv`	(MD5: `4027e994f487487faeac23ef3c4711a9`)
`06.csv`	(MD5: `f1cf93237fdeb723b7db3e3737166841`)
`07.csv`	(MD5: `6ba1cad647dab51aa72599792f6ae200`)
`08.csv`	(MD5: `10400c6faf166902b52fb97042f1e0eb`)
`09.csv`	(MD5: `5757b10b8f99ebce49e7e6329001ffd4`)
`10.csv`	(MD5: `b688dcfa9874622a3b8ea6a570f3f8a1`)

The order in which the rows appear in the answer to your queries is irrelevant for this assignment (no ordering is enforced on the answers, so the DBMS will output rows in an arbitrary order). The names of the columns in the answers are also irrelevant (the above CSV files have no header) so they can be renamed as you wish in the `SELECT` clause. **What is important is the number of columns, their format and the order in which they appear in the output**: the pair $(1, 0.00)$ is not the same as $(0.00, 1)$ or $(1, 0)$. Your query gives a correct answer if it outputs all and only the rows (with repetitions, if that is the case) listed in the corresponding CSV file, no matter on which line.

Other comments. All of the queries can be written using the constructs we have seen in class. Do not use exotic features, especially non-standard ones you may find online in some internet forums. Use `CAST` for type conversions and rounding. In query 06 you may find useful the modulo (%) operator, which returns the remainder of an integer division. The automarker does not have privileges to modify the database; if you want to use views in your queries define them using the `WITH` construct, not the `CREATE VIEW` statement.