DATABASE SYSTEMS

Lab Homework 2

CMPT 354, course section of Dr. Ternovska
Head TA: Alireza Ensan aensan@sfu.ca

Released: 27 February 2019 – **Due: 13 March 2019 at 23:59**

Important Note: *Before you begin, please read carefully the policy on Academic misconduct, in particular about the publication of assignments and tests in online repositories. Students must work individually on this, and other CMPT 354 assignments. You may not discuss the specific questions in this assignment, nor their solutions with any other student. You may not provide or use any solution, in whole or in part, to or by another student. Students are not allowed to publish tests, assignments and their solutions (e.g., put them on an open public repository). You are encouraged to discuss the general concepts involved in the questions in the context of completely different problems. If you are in doubt as to what constitutes acceptable discussion, please ask!*

Database schema. For this assignment we will use the schema `schema.sql` available on Piazza under Resources/Assignments. The schema file is annotated with comments consisting of directives for `datafiller` to generate more realistic-looking test data. These directives will be completely ignored by the DBMS and do not in any way constrain the database.

Note that table PRODUCTS has a column *ptype* that indicates the type of product. The same product cannot have two different values of *ptype*. In the test database, there are three types of products: `BOOK`, `MUSIC` and `MOVIE`. If a query asks for books, this means products where the value of *ptype* is the string `BOOK` (case-sensitive), and similarly for the other two product types. However, if a query asks something *for each product type*, you cannot assume that the values for *ptype* will only be `BOOK`, `MUSIC` and `MOVIE`: there is no constraint in the database schema that enforces that, and your query must work independently of which and how many types of products there are in the database it is run against. Indeed, the instances on which your queries will be executed may well contain other, different types of products (as well as customers' countries).

Assignment. Write the following queries in SQL.

(01) Total number of orders placed in 2016 by customers of each country. If all customers from a specific country did not place any orders in 2016, the country *appears in the output with 0 total orders*. Return all countries and corresponding total counts.

(02) For each product type calculate the average number of times products of that type have been included in an order (taking into account quantities). Orders that do not include any product of a certain type do not contribute to the average for that type (rather than contributing 0). Return the product type and the corresponding average; the latter must be a number with *exactly two decimal places*. Types of products that have never been ordered are not included in the output.

(03) Find invoices that have been taxed. The amount of such invoices is the total of the order they refer to plus 20%. Return the invoice ID. Keep in mind the datatype of "amount" in invoices and *pay special attention to rounding*. Do not use PostgreSQL's `round` function, which is not SQL standard and would likely give you a wrong result.

(04) For each type of product, find the customer who ordered the highest number of products of that type (taking into account quantities). Return the product type and the ID of the customer. If two or more customers are tied for a specific product type they will all be included in the output. If no products of a specific type have ever been ordered, that type will not be in the output.

(05) For each customer, calculate the number of orders placed and the average spend, which is the average total (taking into account quantities and unit prices of ordered products) across all orders placed by that customer. Return the customer ID, the number of orders, and the corresponding average spend. The latter must be a number with *exactly two decimal places*. Orders without detail must be considered in the calculation of the average as having a total of 0. Customers who did not place any orders will be included in the output with 0 orders (not 0.00 or anything else, just 0) and `NULL` average spend.

Submission instructions.

- You will be submitting your work through CourSys. What to submit:
  - a .zip file consisting of 5 files with the SQL queries (so that we could run them). Before producing the .zip file, put all your queries in a directory (folder) called

    `Lastname-Firstname-queries`

    Then produce .zip file of that folder.
- 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 `05.sql`.
- 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 (as if they were not submitted). Please pay attention to this; even if it looks like a trivial detail, it is not.
- Files can be submitted more than once, in which case the previously submitted version will be overwritten.
- Before submitting your files, you should check that they run smoothly in PostgreSQL Server in CSIL, using the exact database schema provided in the link above.
- The final deadline for the submission is on listed above. As file history is not recorded, files whose latest version is submitted after the deadline will not be considered for assessment.

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, as long as they terminate after a "reasonable" time; each query should not take more than a few seconds to run. Style will not be assessed this time either: you could write a query on single line, but this is not recommended for your own sake.

Test data. One of the five instances on which your queries will be assessed has been provided to you on Piazza. It is posted on Piazza under Resources/Assignments. Feel free to create your own instances for testing. You already know how to insert data by hand, but you can also use `datafiller` if you want (this is neither required nor supported by us) for your own interest.

Query answers. The answers your queries are supposed to return on the given database instances are available in CSV format in Piazza, under Resources/Assignment. 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 CSV files we provide have no header) so they can be renamed as you wish in the `SELECT` clause. What is **important** is the number of columns and the order in which they appear in the output: $(1, 2)$ is not the same as $(2, 1, 1)$. Your query gives a fully correct answer if it outputs all and only the rows (with repetitions, if that's 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. Remember that you are here to learn and think with your own head. If you really want to use views in your queries (we have not covered views), define them using the `WITH` construct, not the `CREATE VIEW` statement.