Ghent University - TELIN - DDCM
2021-2022

Assignment 1: SQL Basics

In this assignment, you will test your skills related to interpretation of relational schemas (part 1) and basic SQL (part 1 and part 2) acquired during the last lectures and lab sessions. The SQL questions will only feature basic functionality and combination of multiple tables, so no knowledge about subqueries and aggregation is expected at this moment.

All practical information related to this assignment is listed in the SQL introductory document (`sqlintroduction.pdf`) on Ufora. We recommend to read carefully through this document before starting the assignment and, maybe a second time, before the deadline (March 11th, 2022, 10pm) in order to submit correctly. If you have any additional questions, do not hesitate to contact us.

1 Exercises

Part 1: Interpretation of a relational schema and basic SQL

Provide solutions to the following exercises related to the `rollsrobin` database in a short report (in any format of your choice). Add this report to your final .zip file.

1. Consider the relational schema of the `rollsrobin` database.
   a) Is it possible to assign multiple (different) locations to the same branch? Why (not)?
   b) Is it possible to store locations that are not linked to a branch? Why (not)?
2. Suppose one wants to select all data (i.e. data stored in columns `license_plate`, `enterprisenumber`, `brand` and `model`) related to cars of type ‘cabriolet’. What

is a potential issue that one may have when executing the following SELECT-query for this task and why? How would you solve this? Explain in your own words.

```
SELECT DISTINCT c.license_plate, c.enterprisenumber, c.brand, c.model
FROM car c
    INNER JOIN carmodel cm ON c.brand = cm.brand
    INNER JOIN type t ON cm.type = t.name
WHERE t.name = ’cabriolet’;
```

3. Consider the following SELECT-query.

```
SELECT * FROM contract c1
    LEFT JOIN contract c2 ON c1.employeenumber = c2.employeenumber
                                AND c1.enterprisenumber = c2.enterprisenumber
                                AND c1.period_begin < c2.period_begin
WHERE c2.period_begin IS NULL;
```

   a) Explain, in your own words, what this SELECT-query achieves. You should not give the result table of this query or explain this query in technical terms, but explain what the semantical outcome is of this query when executed on data that is stored in the rollsrobin database.

   b) Draw the entire table that will be returned upon completion of the given query, starting from the example data of the contract table, given in the appendix of this document.

   c) Suppose one removes the entire WHERE-clause from the SELECT-query. Draw, again, the entire table that will be returned upon completion of the given query (now without the WHERE-clause), starting from the example data of the contract table, given in the appendix of this document.

   d) Is it possible to add the row {employeenumber: 1, period_begin: 12/01/2021, period_end: 31/01/2021, enterprisenumber: 103} to the contract table given in the appendix of this document, considering the relational schema of the rollsrobin database and the example data? Why (not)?

4. Imagine that the designers of the rollsrobin database would have left out the employee table from the relational schema. Instead, all data related to employees would then be stored in the person table (which would include an extra column employeenumber). The foreign key in the table contract, originally pointing to the employee table, would then point to the column employeenumber in the person table. What are possible (dis)advantages of this approach? Explain in your own words.

Part 2: Basic SQL

Provide SQL `SELECT`-queries as solutions to the following exercises related to the `rollsrobin` database. Each query should be added to a separate .sql file with filename `studentcode_firstname_lastname_X.sql` in which you substitute 'studentcode', 'firstname' and 'lastname' with your studentcode, firstname and lastname respectively, and 'X' with the number of the question (1 to 5). Add these .sql files to your final .zip file.

1. Construct a list of unique email addresses of all persons who did a registration for which (1) the rented car is owned by a branch for which the sum of the numbers in the branches postal code (`branch.postalcode`) equals the sum of the numbers in the postal code of the person, who rented the car (`person.postalcode`), and (2) the first date of the rental period (`registration .period_begin`) is in a month which name ends with letter 'r' (in English). You may assume that all postal codes consist of 4 digits between 0 and 9. In the result table, we expect one column with name `email` of datatype `varchar`.

   Example: Table 1 shows a list of example registrations. Given this data, the persons with email address person1@ugent.be and person3@ugent.be did a registration meeting the listed requirements and should, therefore, be adopted in the result table.

email	**postal code person (sum)**	**postal code branch (sum)**	**begin registration (month)**
person1@ugent.be	9000 (9)	8100 (9)	10/09/2021 (September)
person2@ugent.be	2000 (2)	9000 (9)	24/12/2021 (December)
person2@ugent.be	2000 (2)	1100 (2)	03/03/2022 (March)
person3@ugent.be	1000 (1)	1000 (1)	15/10/2021 (October)
person3@ugent.be	1000 (1)	2000 (2)	31/10/2021 (October)

Table 1: Example data exercise 1.

2. Construct a list of unique email addresses of all employees who rented a car at a company at which they have worked. Moreover, the first date of the rental period should fall within the period they were working at the company (i.e. `registration.period_begin` should be between `contract.period_begin` and `contract.period_end`, boundaries inclusive). In the result table, we expect one column with name `email` of datatype `varchar`. Order the results in this table alphabetically according to the values in the column `email`.

   Example: Table 2 shows a list of contracts of employees and Table 3 shows a list of registrations done by these employees. Given this data, only the person with email address person1@ugent.be did a registration meeting the listed requirements and should, therefore, be adopted in the result table.

email	**begin contract**	**end contract**	**enterprisenumber contract**
person1@ugent.be	01/02/2022	01/02/2023	101
person2@ugent.be	16/01/2022	16/04/2022	101
person2@ugent.be	17/04/2022	01/06/2022	102
person3@ugent.be	01/02/2022	01/03/2022	103
person4@ugent.be	01/02/2022	01/03/2022	104

Table 2: Example contract data exercise 2.

email	**begin registration**	**end registration**	**enterprisenumber rental car**
person1@ugent.be	10/04/2022	13/04/2022	101
person2@ugent.be	12/01/2022	18/01/2022	101
person2@ugent.be	22/02/2022	25/02/2022	102
person4@ugent.be	02/05/2022	07/05/2022	104

Table 3: Example registration data exercise 2.

3. Construct a list of locations (postal code and municipality) in which each location (1) is not registered as a residence of a person in the database, (2) has an even postal code and (3) does not start with letter 'B' (case insensitive). In the result table, we expect two columns with corresponding datatype: `postalcode` (`varchar`) and `municipality` (`varchar`).

   Example: Table 4 shows a list of example locations. Given this data, only the location with postal code '8400' and municipality 'Ostend' meets the requirements and should, therefore, be adopted in the result table.

postalcode (even?)	**municipality**	**residence_of**
9000 (true)	Ghent	person1@ugent.be, person2@ugent.be
8400 (true)	Ostend	/
9120 (true)	Beveren	/
8000 (true)	Bruges	person3@ugent.be
9031 (false)	Drongen	/

Table 4: Example data exercise 3.

4. Give for each location in the database (uniquely defined by postalcode and municipality), the number of parts that make up the name of the municipality. Here, a part is defined as a substring of the municipality that is separated from other substrings by either a hyphen (“-”) or a space (“ ”).

   Remark: Your solution should account for the fact that a municipality name could contain both hyphens and spaces at the same time. The hyphens or spaces themselves should not be counted as parts. Thereby, you may assume that there always is only one space or hyphen separating two substrings (so two substrings are always separated by simply *one* hyphen or *one* space, and not by *multiple* hyphens/spaces, or by *combinations* of hyphens/spaces.

   In the result table, we expect three columns with corresponding datatype: `postalcode` (`varchar`), `municipality` (`varchar`) and `number` (`integer`).

   Example: Table 5 shows a list of example locations, together with the number of parts out of which the municipality name is constituted

postalcode	**municipality**	**number**
9000	Ghent	1
9051	Sint-Denijs-Westrem	3
9120	Beveren	1
4780	Sankt Vith	2
9031	Drongen	1

   Table 5: Example data exercise 4.

5. Give all possible couples of registrations that were done by the same person (i.e. the table that is returned by your query should contain rows consisting of two registrations that were conducted by the same person). Be aware of the fact that no rows may be present in your result that contain two identical registrations (i.e. same person, car, begin date and end date)! In order to prevent this, the value in column `period_begin` of the first registration should be chronologically strict before the value in column `period_begin` of the second registration. However, if this value is the same for the two registrations, the value in column `license_plate` of the first registration should come alphabetically strict before the value in column `license_plate` of the second registration.

   In the result table, we expect five columns with corresponding datatype: `email` (`varchar`), `license_plate1` (`varchar`), `period_begin1` (`date`), `license_plate2` (`varchar`) and `period_begin2` (`date`).

   Example: Table 6 shows a list of example registrations. Based on these registrations, the output of your query should look like Table 7

email	**license_plate**	**period_begin**	**period_end**
persoon1@ugent.be	1-PZR-102	2018-03-15	2018-03-20
persoon1@ugent.be	1-IZX-389	2018-04-15	2018-04-20
persoon1@ugent.be	1-PPA-110	2018-03-15	2018-03-27
persoon2@ugent.be	1-PZR-102	2018-03-21	2018-03-24

Table 6: Example data exercise 5.

email	**license_plate1**	**period_begin1**	**license_plate2**	**period_begin2**
persoon1@ugent.be	1-PZR-102	2018-03-15	1-IZX-389	2018-04-15
persoon1@ugent.be	1-PPA-110	2018-03-15	1-IZX-389	2018-04-15
persoon1@ugent.be	1-PPA-110	2018-03-15	1-PZR-102	2018-03-15

Table 7: Result data exercise 5.

Appendix

In Table 8, you find example data of the `contract` table that you have to use when solving exercise 3 (part 1) in this assignment. You may assume that employees with numbers 1, 2 and 3 exist, as well as branches with numbers 101, 102, 103 and 104.

employeenumber	**period_begin**	**period_end**	**enterprisenumber**
1	12/01/2021	31/05/2021	101
1	01/07/2021	31/12/2021	102
1	01/02/2022	30/06/2022	101
1	01/08/2022	01/08/2023	101
2	24/02/2022	15/05/2022	101
2	31/05/2022	30/06/2022	104
3	01/10/2021	30/09/2022	103

Table 8: Example data `contract` table.