School of Computer Science, McGill University

COMP-421 Database Systems, Winter 2020

Project 2: Creating Your Database

Due Date February 28, 11:59pm

In this assignment you are going to refine your schema, create your database using DB2 or PostgreSQL and write and run some SQL queries. We have created a UNIX and a DB2/PostgreSQL group account for each registered group. Information on group names and passwords can be found on myCourses. It is important to reset the default password for your group's unix account. You will loose points if you do not ! Make sure all members in your group is aware of the password.

It is suggested that for all your deliverables you keep files with all statements. In particular, it would be good to generate a script that populates your database so that you can delete and recreate your database as needed. This will allow you to experiment with queries that potentially delete large parts of your database, and be able to get back to the original database.

Please note, that for this and the next project deliverable you might frequently need to work with the DB2/PostgreSQL online information (link from myCourses) in order to figure out how things work in the database system. It is an essential part of the project to learn how to find the needed information in the online help. To make life a bit easier at the beginning, some extra links/documents are given on myCourses. They describe the most essential things.

In this assignment, you can use any of the interactive user interfaces offered by DB2/PostgreSQL when you start development. Some are shortly introduced on myCourses.

You will have to provide the queries you write as well as the output. You have to figure out how to do so with the database system of your choice (DB2 or Postgresql). See P2_notes for some suggestions.

WARNING !! Do not generate more than 500 records per table.

1 Assignment

1. (0 Points) Please attach a copy of your modified relational schema of Project #1 according to the feedback given on it. The new design will not be graded but will be used as a basis for the future assignments.

2. (25 Points) Write a SQL database schema for the relational schema you have designed using the `CREATE TABLE` command and enter it in the database. Choose suitable data types for your attributes. Indicate primary keys, foreign keys or any other integrity constraints that you can express with the commands learnt. Indicate the constraints you cannot express. The Online Information contains detailed information about data types, and the `CREATE TABLE` statement.

   Turn in your CREATE TABLE statements.Furthermore, (i) for DB2 use command line processor command DESCRIBE tablename (ii) for PostgreSQL use \d table name (prints the description of the relation on the screen) for each of your relations, print the result and turn it in.

3. (5 Points) Execute five INSERT commands to insert tuples into one of your relations.
   Turn in your INSERT statements. Furthermore, print and turn in the response of CPL/psql when you type the INSERT commands. Print and turn in the result when you issue a SELECT * FROM relationname command.

4. (10 Points) Insert in all your tables enough meaningful information so that the queries that you create provide meaningful results. The results of the following queries that you develop should have a reasonable number of results so that we can be convinced that your queries are correct (maybe 5-10 tuples). If you have real-world data, feel free to import it (but do not insert more than 500 records in a table). Information of how to import data into DB2 tables is provided on my courses.
   For each table show the output, truncated to the first 5-10 tuples, that are returned when you issue a SELECT * FROM relationname command. Hint:- I showed a trick in class to do exactly that.

5. (20 Points) Write five queries on your project database, using the select-from-where construct of SQL. The queries should be typical queries of the application domain. To receive full credit, all but perhaps one of your queries must exhibit some interesting feature of SQL: queries over more than one relation, subqueries, aggregations, grouping etc.
   Turn in a description of all the relations that you use in your queries (e.g., the original create statements or printouts from the SQL "describe table yourname" function), a description of what each of your queries is supposed to do, the SQL statement of each query, along with a script illustrating their execution (for example the screenshot when you execute the query). Your script should be sufficient to convince us that your commands run successfully. Please do not, however, turn in query results that are more than 50 lines long.

6. (12 Points) Write four data modification commands for your application. Most of these commands should be "interesting," in the sense that they involve some complex feature, such as inserting the result of a query, updating several tuples at once, or deleting a set of tuples that is more than one but less than all the tuples in a relation.
   Turn in a description of all the relations that you use in your modifications but are not described so far. Provide a short description of what each of your statements is supposed to do, the SQL statements themselves and a script or screenshot that shows your modification commands running in a convincing fashion.

7. (10 Points) Create two views on top of your database schema.
   Turn in an informal description what data each of the views represents, show your CREATE VIEW statements and the response of the system. Also, show a query involving each view and the system response (but truncate the response if there are more than a few tuples produced). Finally, show a script of what happens when you try run an SQL UPDATE statement on each of your views. Are either of your views updatable (that is, the database system will automatically translate the update into an update on the base table(s). Explain why or why not. Summarize the conditions that must hold so that DB2/PostgreSQL allows updating a view.

8. (8 Points) Add two CHECK constraints to relations of your database schema.
   Turn in the revised schema, its successful declaration, and the response of database to modifications (insert/update) that violate the constraints.

9. (10) Points. **C**reativity Points:
   The purpose of this is to encourage students to try innovative approaches in their solutions. Some of these approaches can even reduce your overall effort. You can implement either one of it. If you have other cool ideas, come talk to me to see if I am cool about it as well.

   - **Automated data generation:** Instead of hand coding all the insert statements, can you use a tool (may be your own), websites etc that can generate the required data sets for you ?
     Do this for 2 - 3 tables in your project.
     WARNING !! Do not generate more than 500 records per table.
   - **Real data sets:** Generate data which are more real world like (For example, instead of names like 'customer001', 'customer002' etc, it should be like 'Mike McCarthy', 'Katie Lohan' etc. same with attributes from other domains.
     A minimum of 100 records is required for a table to be considered for this.
     You may use data provided by other websites etc for this purpose, but references needs to be provided.
     Do this for 2 - 3 tables in your project.
     WARNING !! Do not generate more than 500 records per table.
   - **Cool SQL Features:** Use advanced SQL features, not covered in class. This includes Recursion, OLAP SQL Functions(Such as Window Functions), etc. For this. You can use internet/Manuals to find lot of examples and information on syntax. **You need to turn in these SQLs separately, along with their output(truncate if too big) and a brief description of the purpose the SQL is answering.**
     Implement 1 - 2 SQLs using this approach.
     Note:- Stored Procedures, User functions, triggers, etc. do not count for this, they will be covered in Project 3. If you want to try something not listed here, and is not sure if that counts, ask me.

- **Complex Analytical Queries:** Come up with a complex business requirement (to read data) and develop SQL queries to substantiate them.
  Implement 1 - 2 SQLs using this approach.
  Analytical queries are characterized by the fact that they work over many tables, and a larger portion of records from those tables and provides a very selective or summarized output. Aggregation operators are often an integral part of such queries.

  Some examples are given below.

  Give a list of customers between the ages of 30 and 40 who has been depositing more than $500 into their savings account every month for the last one year and has not enrolled in any pension plans. (This is useful in finding a potential list of customers with savings/habits who will make a good target for marketing pension plans).

  Find the average insurance claims by various age-categories, marital status, income, make of primary car, ownership of home, etc. for the customers in PQ. (This is useful in determining what are the 'risk' categories, that can be charged higher premium).

  Find customers who used to spent more than $50 on average per visit and has made at least 10 visits in the last three years, but has not been visiting in the past six months and has address in 'Montreal'. (This is good target to sent some discount vouchers to bring back old customers).

  Write your SQL, also provide the output (truncate if required) and also provide a "business case" for your SQL.

Write a brief description about which of the above items you were able to incorporate, provide references to websites etc if you used some tools, if you built your own, attach it separately.

For items that are not very evident from your previous questions outputs, provide some evidence to substantiate your claim.