University of Toronto
CSC343, Fall 2018

Assignment 2

Due: November 22nd, 11:59pm

Learning Goals

By the end of this assignment you should be able to:

- understand a new database schema
- write complex SQL queries
- test an SQL query (including designing a test dataset)
- work with a DBMS (e.g., postgreSQL) documentation
- embed SQL in a high-level host language. For this assignment, you will connect to a database and execute SQL queries using Java, and its Java Database Connectivity (JDBC). JDBC provides a mechanism for dynamically loading and registering Java packages with the JDBC Driver Manager, which in turn provides a mechanism for creating JDBC connections that support executing remotely a variety of SQL statements, e.g., SELECT, CREATE, INSERT, UPDATE and DELETE, etc.

General Instructions

You are strongly encouraged to test all your development for this assignment on the CS Teaching Labs machines. Your code will be tested on these machines. You are allowed to work with up to two partners for this assignment. You must declare your team (whether it is a team of one or more students) and hand in your work electronically using MarkUs.

Download the starter files provided on the course webpage:

- The database schema, `parlgovSchema.ddl`
- A sample data set, `parlgovData.sql`

Your code for this assignment must work on any database instance (including instances with empty tables) that satisfies the schema. An auto-tester will be provided to test the correctness of the queries.

Schema

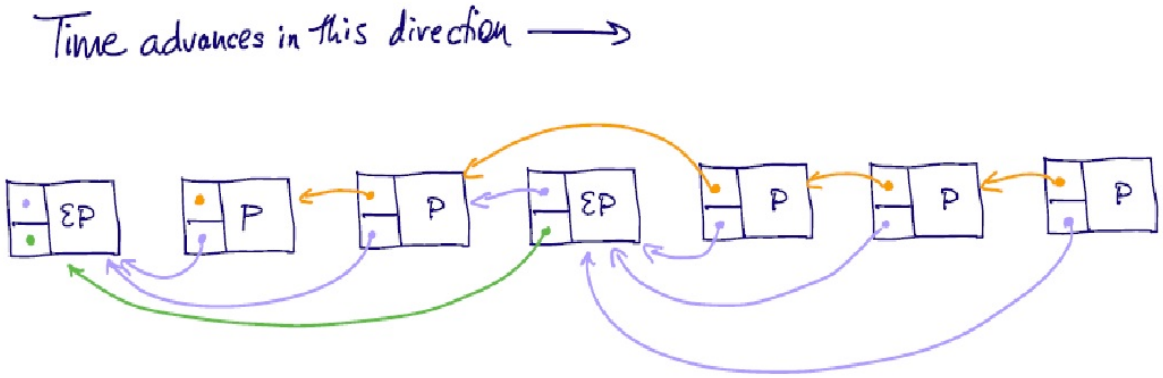
In this assignment, you will use SQL queries to analyze real election data. The data comes from ParlGov[1]. It contains information on political parties, elections and cabinets for countries in the EU (European Union) and the OECD (Organization for Economic Co-operation and Development)

The schema for this assignment is complex. Here are a few things to keep in mind:

- The schema can record information about two types of elections:
  - A **parliamentary election** is an election (within a country) to choose a national parliament (and government).
  - A **European parliament election** (EP election) is an election, held across all EU countries, to elect national representatives to the European parliament.
- The same schema is being used to represent election data from many countries, with numerous variations in their form of governance. Some of the terminology may be used in an unfamiliar way. Follow the definitions and descriptions in the schema, even if they sound different from the electoral system used in Canada.

Pay close attention to the following tables:

- Each row of the `election` table records information about a single election. Each row includes the ID of the previous Parliamentary election and the previous EP election (using attributes `previous_parliament_election_id` and `previous_parliament_election_id`). You can think of each of these two attributes as inducing a linked list datastructure, more specifically, they induce two linked lists within the data stored in the table. Keep in mind that a Parliamentary election has a reference to the previous EP election, and an EP election has a reference to the previous Parliamentary election. This diagram captures informally the structure embedded in the `election` table:



  The orange arrows show the linked list of parliamentary elections going back through time, and the green arrows shows the linked list of EP elections going back through time. (References across election types are also stored. When looking at the whole structure, it becomes apparent that there are more than just two linked lists.)

- The `election_result` table contains data about political alliances formed between different parties in an election. To represent that a set of parties formed an alliance in an election, the database singles one party out (the **head** of the alliance). It is arbitrary which party is the head of an alliance. All the other parties in the alliance refer to it in their `alliance_id` attribute. More specifically, the other parties in an alliance refer to the party that is the head of the alliance by storing in `alliance_id` the `id` of the election result for the head party. The `alliance_id` value for the head party of the alliance is NULL. (Keep this in mind, as `alliance_id` may sound like a unique identifier for the alliance rather than a reference to one of the parties in the alliance!) For example, if parties A, B, C, and D formed an alliance in election e1, then the table could include these rows:

[1] http://www.parlgov.org

id	election_id	party_id	alliance_id	seats	votes
id1	e1	A	NULL		
id2	e1	B	id1		
id3	e1	C	id1		
id4	e1	D	id1		

Part 1: SQL Statements

General requirements

Write SQL statements corresponding to the following query definitions. Write your SQL statement(s) for each question in separate files. For each question `qi`, you will submit a `qi.sql` file. The `qi.sql` file should contain:

(a) the table definition that is required for that query,

(b) the SQL query and INSERT INTO command for adding values returned by the query to the table you defined.

You can use views to make your queries more readable. However, each file should be entirely self-contained, and not depend on any other files, as each file will be run separately on a fresh database instance, and so (for example) any views you create in `q1.sql` will not be accessible in `q5.sql`.

Each of your files must begin with the line `SET search_path TO parlgov;` We will be testing your code in the **CS Teaching Labs environment** using PostgreSQL. Make sure that your code runs in this environment. **Code which works on your machine but not on the CS Teaching Labs will not receive credit**.

The queries

We define the **winning party** to be the party that won the most votes. If several parties are tied for the most votes, we say that there are multiple winning parties.

Write SQL queries for each of the following:

1. For each country, and for each political party, for each of the years between 1996 to 2016, both inclusive, report the name of the country, the name of the party, and a description of the range into which the number of valid votes it received falls, in the following format: *(Lb-Ub]*, for example, (20-30]. (*Lb* is the lower bound of the range and *Ub* is the upper bound of the range.)

   These are the ranges to consider:

   - non-zero and below 5 percent of valid votes inclusive,
   - 5 to 10 percent of valid votes inclusive,
   - 10 to 20 percent of valid votes inclusive,
   - 20 to 30 percent of valid votes inclusive,
   - 30 to 40 percent of valid votes inclusive, and
   - above 40 percent of valid votes.

   Important:

   - If there is more than one election in the same country in the same year, report the average of the percent of valid votes that a party received across those elections.
   - If there are no parties in a given range, do not report that range, i.e., the ranges are defined only for the parties and elections for which the number of votes is recorded.
   - A country that does not have any elections in a year, should not be included in the results for that year.

Attribute	Description
year	year
countryName	name of a country
voteRange	the percentage range that the party falls into
partyName	short name of a party
All years?	Every year where at least an election has happened should be included.
Duplicated info?	No year-country-party combination should occur more than once in the answer.

2. Retrieve the parties that have won more than three times the average number of elections won by the parties in the same country. List the country name, party name, its party family name as well as the total number of elections won, as well as the id and the year of the most recently won election.

Attribute	Description
countryName	Name of the country
partyName	Name of the party
partyFamily	Name of the family of a party
wonElections	Number of elections the party has won
mostRecentlyWonElectionId	The id of the election that was most recently won by this party
mostRecentlyWonElectionYear	The year of the election that was most recently won by this party
All countries, parties?	Include only countries and parties who meet the criteria of this question.
Duplicate countries and party families?	Countries and party families can be included more than once with different party names.

3. For an election, the **participation ratio** (a value between zero and one) is the ratio of votes cast to the number of citizens who are eligible to vote. Find the countries that had at least one election between 2001 to 2016, inclusive, and whose average election participation ratios during this period obey the following: for a pair year Y1 and year Y2, where at least one election has happened in each of them, if Y1 $<$ Y2, then the average participation in year Y1 is $\leq$ the average participation in year Y2. (If more than one election happens in a country in a year, we calculate the participation ratio as the average participation ratio across those elections.) For such countries, report the name of the country and the average participation ratio per year for the years between 2001 to 2016.

Attribute	Description
countryName	Name of the country
year	year
participationRatio	The average percentage ratio of citizens who voted in this year
All countries?	Include only countries that meet the criteria in the question.
Duplicate info?	Do not include a country if there are no elections in it between 2001 to 2016.

4. The database records several policy positions of political parties, including their left-right positions. Suppose the left-right range is divided into 5 intervals ([0,2), [2,4), [4,6), [6,8) and [8,10]). Create a histogram of parties in the database and their left-right position.

Attribute	Description
countryName	Name of the country
r0_2	Number of parties that have left/right position in [0,2).
r2_4	Number of parties that have left/right position in [2,4).
r4_6	Number of parties that have left/right position in [4,6).
r6_8	Number of parties that have left/right position in [6,8).
r8_10	Number of parties that have left/right position in [8,10).
Every country?	Every country should be included, even if it has no parties with recorded party position information.
Duplicate info?	No country should be included/counted more than once.

Part 2: Embedded SQL

Implement functionality for an electoral analytics app. The app is written in Java and connects to a database containing election data. The functionality will be implemented as Java methods that act as wrappers around SQL queries.

General requirements

- Do not use the standard input or output. (Doing so will result in the autotester terminating.)
- Write a method called `connectDb()` to connect to the database. When your method calls the `getConnection()` method, it must pass the database URL, username, and password that were passed as parameters to `connectDb()`. These values must not be hard-coded in the method as the autotester will use the `connectDb()` and `disconnectDB()` methods to connect to the database with its own credentials.
- Do **not** call `connectDb()` and `disconnectDB()` in the other methods you are asked to implement; you can assume that they will be called before and after, respectively, any other method calls.
- All of your code must be written in the file `Assignment2.java`. This is the only file you should submit for this part.
- You should not change the interface of any of the methods you are asked to implement, but you can write helper methods.
- To run your code, you will need to include the JDBC driver in your class path.
- JDBCSubmission is an abstract class that is provided to you. You should not make any changes in this file and you do not need to submit this file.

To Do

Open the starter code in `Assignment2.java`, and complete the following methods.

1. `connectDB`: this method connects to a database using the supplied credentials.
2. `disconnectDB`: this method disconnects from the database.
3. `electionSequence`: this is a method that, given a country, retrieves the list of elections in that country and the cabinets formed after that election and before the next election (of the same type).
4. `findSimilarPoliticians`: A method that, given a politician, returns other politicians that have similar comments and descriptions in the database. See section Similar Politicians below for details.

You will have to decide how much functionality will be accomplished on the database side and how much in Java. Encapsulate in SQL as much functionality as possible.

Similar Politicians

This method identifies similar politicians using the descriptions that are available about them in the parlgov database. Two politicians are considered similar if the information available about them is similar enough, as assessed using the Jaccard similarity measure, which provides a simple similarity score (between 0 and 1). For two sets of strings, the Jaccard similarity is defined as the size of the intersection divided by the size of the union of two sets. For example, the Jaccard similarity of `S1 = {Ontario, Quebec}` and `S2 = {Alberta, Ontario, Manitoba}` is 0.25 as their intersection is `{Ontario}` and union is `{Alberta, Ontario, Manitoba, Quebec}`. We provide a helper method, `similarity`, which computes the Jaccard similarity of two sets of strings. You should use it to find the politicians whose descriptions, i.e., the contents of their `description` attributes, are similar above a given threshold.