CS377, Spring 2025

Assignment 2 - Part 2 (60 points)

Due: Wednesday March 26 at 11:59PM. You can optionally work in pairs on this assignment. (Late submissions policy: see syllabus for details.)

General Instructions

This second part of A2 is worth the remaining 60% of the grade. We will be using the exact same DIME database from A2-Part1. You are free to work on your *Google Cloud Platform* PostgreSQL instance or on a *local* PostgreSQL server for this assignment.

Here is a reminder of the steps you need to setup your database. Download the starter zip folder from the A2 directory on Canvas. You will find:

- The database schema script (`a2_dimeDB.sql`). Execute the lines in this SQL script in your postgres instance. This script will create the schema and populate its tables from a public Google storage bucket. If you are working on a **local** server and using the pgAdmin tool, replace every `\copy` with `copy` in the script.
- Note that your submission will be tested later on a *different* dataset from the one that will be loaded into your database here, so make sure your queries can work on any valid database instance and don't contain *hard-coded* identifiers, for example.
- Expected answers for some of the queries (in `expected_answers/`), based on the provided test dataset. These answers were produced on a GCP PostgreSQL instance. There may be minor differences in the ordering of *strings*, if you run your queries on a different platform or operating system with different text encoding.

Once you have submitted your files, be sure to check that you have submitted the correct version; new or missing files will not be accepted after the due date (unless you and your partner -if applicable- have a late-token remaining and you resubmit within 24 hours of the deadline).

DIME Schema Description

Copying the schema description as-is from A2 Part1's handout, below (for easy reference):

Congratulations, you've been hired by a research think tank in D.C. to study how campaign financing and political donations impact politicians in America. You will be using a real-world data set collected and published by scholars at Stanford University, known as: The Database on Ideology, Money in Politics, and Elections (DIME) Adam [2016], Bonica [2016]. This database was developed as part of a large-scale effort to "construct a comprehensive ideological mapping of political elites, interest groups, and donors."

This database contains 5 tables, described briefly below. For detailed descriptions of the attributes in each table, you will read specific sections from the official DIME documentation (referenced below). You can find the official documentation files in our A2 Canvas folder.

- **Recipient**: A tuple in this relation represents a political candidate and contains information on the election cycle, fundraising statistics, election outcomes, and characteristics like name and gender, if applicable – among other features. Note that we refer to candidates as *recipients* because they *receive* donations from contributors. Each candidate is identified by a unique recipient ID referred to as "bonica.rid". [1]

  Read the detailed description of this table's attributes in **section 5.1** of the document **dime_codebook_v3_1.pdf**, pages 6-9.

[1] Bonica is the last name of the scholar who supervised the DIME project at Stanford.

- **Contributor**: A tuple in this relation represents a donor, also known as a contributor. Donors can be individuals or organizations. Each donor is identified by a unique contributor ID referred to as "bonica.cid". This table also contains a column for each election year covered by the data, storing the amount of donations given by a contributor during that year. For example, the column "amount.2014" stores the amount of campaign donations made by a contributor during the 2014 election cycle, and so on.

  Read the description of this table's attributes in **section 5.2** of **dime_codebook_v3_1.pdf**, pages 10-11.

- **Contribution**: A tuple in this relation represents a donation made by a contributor (identified by "bonica.cid") to a recipient/candidate (identified by "bonica.rid") during a given election cycle.

  Read the description of this table's attributes in **section 5.3** of **dime_codebook_v3_1.pdf**, pages 11-13.

- **Bill**: A tuple in this relation represents a congressional bill. Each bill is uniquely identified by a "bill.id". A bill has a name, a description, a list of sponsors and co-sponsors, among other things.

  Read the description of this table's attributes in **section 2.3** of the second document **dime_plus_codebook.pdf**, page 5. (Some additional attributes in the database are incorrectly marked as being part of another table called Congressional Text, described in **section 2.1**, pages 4-5, of the same document.)

- **Vote**: And finally, this table provides information on how congress members voted for a given bill. Each tuple represents a single vote made by a legislator (identified by their "bonica.rid" value) on a bill (identified by "bill.id"), showing their "vote.choice", whether they were the sponsor/cosponsor for the bill or not, and more.

  Read the description of this table's attributes in **section 2.2** of the document **dime_plus_codebook.pdf**.

SQL Statements

Important things to keep in mind when writing your queries:

- Write your SQL statement(s) for each question in separate files named `q6.sql`, `q7.sql`, ...etc., and submit each file on **Gradescope**. You are encouraged to use views to make your queries more readable. However, each file should be entirely self-contained, and not depend on any other files; each will be run separately on a fresh database instance, and so (for example) any views you create in `q6.sql` will not be accessible in `q10.sql`.
- Each of your files must begin with the line: `SET search_path TO dimeDB;`

  Failure to do so may cause your query to raise an error, leading you to get a 0 for that question.
- The output of each query must match **exactly** the specifications given in the question: attribute names and order, ordering of the tuples, and how to treat duplicates. It is your responsibility to make sure your code runs with no errors and the output matches our expectations exactly, to avoid losing points due to autograder crashes.
- Some attributes in the schema have a dot "." symbol in their name. These attributes must be enclosed in double quotes within your SQL queries. For example, every time you need to reference the attribute `bonica.cid`, you must type it as "`bonica.cid`". E.g.: `SELECT "bonica.cid" FROM Contributor ....`
- None of your query results should report duplicated rows.

6. **(10 points) Never say Nay (if it's your own bill!)**

   Legislators can sponsor a bill in congress but then vote against it (I know..I only learned this recently). Among the legislators who sponsored bills in the database, which ones *never* sponsored a bill and voted against it? For these legislators, report their `bonica.rid`, their party code, and the total number of bills they sponsored.

   The output of your query must match the exact format described below:

Attribute	
bonica.rid	Unique ID of the legislator
party	The code for the legislator's party affiliation
numbills	Total number of bills they have sponsored
Order by	numbills, in descending order

7. **(12 points) The marriage of top recipients and top donors.**

   Remember the top 10 contributors you identified in **Query 4** from Part 1 of this assignment (i.e., top contributors across all years)? Now let's consider only the top 3 contributors (also across all years); you can still assume there are no ties in these top donors.

   For each one of these top 3 contributors, identify the candidate who received the maximum amount of contributions from them. Note that 'candidate' here refers to individuals specifically, not committees. You can find information on recipient types in different parts of the data, but the most reliable method is to use table Contribution (rather than Recipient). [2] If a contributor's donations were exclusively made to committees, then report *null* for their top recipient (i.e., the top recipient should appear as an empty cell in the final result, not the literal string 'null').

   If there are ties in the recipients who received the maximum amount for a given donor, report them all. For example, if candidates A and B received the same maximum amount of donations from Contributor C, then the combinations `(C, A)` and `(C, B)` should appear in separate tuples in the final result (along with any other required attributes, of course).

Attribute	
bonica.cid	Unique ID of the contributor
donor_name	Contributor name
bonica.rid	Unique ID of the top recipient
recipient_name	Top recipient name
max_amount	Amount of donations this top recipient received from this top contributor
Order by	donor_name ascending, then max_amount descending

8. **(12 points) Gender gap?**

   Analysts often talk about a gender gap in political donations, where men are more likely to contribute to political campaigns than women. But how does this gap (if any) vary by election level (i.e., federal, state, or local)?

   In the data, every contribution record has information on the gender of the contributor and the "seat" sought by the candidate they contributed to. For contributions where gender was logged as 'M' or 'F': report the average amount of donations made per gender group, computed separately for each election level.

   To identify election levels, use the attribute "seat". If the value of seat starts with 'federal:', then this is a donation for a federal election. If seat starts with 'state:' or 'local:' then these donations are for state and local levels, respectively. For this query, ignore any other "seat" value that does not start with one of these 3 specific prefixes.

   When reporting the final result, the column title should be "level" and the column values should be "federal", "state" or "local" depending on which seat level this tuple represents.

Attribute	
level	Level of election being compared: `federal`, `state`, or `local`
contributor.gender	This will either be `M` or `F`
count	The number of contributions counted for this (level, gender) category
average_amount	The average amount of contributions made for this (level, gender) category
Order by	level descending, then contributor.gender ascending

[2] We will discuss this DB design flaw in a future topic! But let's work with it for now.

9. **(12 points) Hey hey, ho ho, Independents've got to go.**

   A new bill was just signed into law, which forbids people from running for office or voting on bills if they are independent candidates/legislators. This law also requires removing any traces of such independent folks from the database, whether it is records of recipients registered as independents, or votes made by independent legislators, records of any contributions they received, and so on.

   Write a sequence of SQL statements that take care of deleting any traces of independents from the DIME database.

10. **(14 points) Vote Prediction: Data Prep!**

   Your think tank just started a research project with Dr. Bonica in Stanford to study the impact of political donations on legislative voting behavior using machine learning Bonica [2018]. Specifically, we want to analyze the correlation between two variables: the 'topics weights' (tw) already computed and stored for each Bill in the database on different topics (e.g., `tw.healthcare`), and a variable we will call 'donation weight' (dw) which reflects the weight of donations received by a legislator from entities known to advocate for certain issues. For example, the `dw.healthcare` value for a candidate reflects the weight of contributions received from entities focused on improving the healthcare system (and can be correlated with 'tw.health.care' in table Bill), and so on.

   You will create a table called `Influence` by executing the SQL statements below, but after making some important changes. This table uses foreign key constraints to reference a bill (via "bill.id") and a vote on that bill (via "vote.id"). For each (bill, vote) combination, it will store the donation weight (dw) for several areas.

   Task 1: Modify the CREATE TABLE statement such that this table would only accept dw values (for all topics) between 0 and 1, inclusive; i.e., any attempt to store a dw value that is < 0 or > 1 would be rejected. Additionally, the default value for all dw columns should be 0.

```
DROP TABLE IF EXISTS Influence; -- to start with a clean slate
CREATE TABLE Influence ( -- now create the table
  "bill.id" varchar REFERENCES Bill("bill.id"),
  "vote.id" varchar REFERENCES Vote("vote.id"),
  "dw.economy" float,
  "dw.environment" float,
  "dw.foreign.policy" float,
  "dw.womens.issues" float,
  "dw.guns" float,
  "dw.healthcare" float,
  PRIMARY KEY("bill.id", "vote.id")
);
```

   After applying the changes described above and executing the modified CREATE TABLE statement, next we want to populate this table with some data.

   Task 2: The bill and votes we want to insert into this table should reflect the bill(s) with the highest number of co-sponsors in the database (i.e., the same criteria you applied in **Query 5** from Part 1 of A2). If there are ties, we want to add rows that reflect them all.

   More specifically, use `INSERT INTO` to add tuples to table Influence which pertain to that special bill (with highest number of cosponsors):

   - the values for "bill.id" and "vote.id" need to reflect votes made for that bill;
   - the values for "dw.environment" and "dw.foreign.policy" should be 0.3 and 0.7, respectively;
   - the rest of the columns should take the default values (i.e., do not set their values explicitly in your insertion statement!)

Your solution for this question should be dynamic, such that if it is tested on any valid database instance it would properly identify the bill(s) with the highest number of cosponsors and would insert tuples into table Influence, accordingly.

Your submission for this question must begin with:

```
DROP TABLE IF EXISTS Influence;
```

followed by your own modified `CREATE TABLE Influence` statement, followed by the `INSERT INTO` statement. Don't forget **semi-colons** after every statement, otherwise the script will not execute.

And finally, **do not DROP** the table at the end of your submission file! The autograder will assume that the table exists so it can test it.

Important Submission Remark: Please follow all submission instructions mentioned in this handout very carefully. One of the key challenges in grading Part 1 was the presence of issues like wrong submission file names, forgetting semi-colons, typos in queries or setting the search_path, etc. We were lenient in part 1 since it was your first SQL assignment, but for this part we will not be manually fixing these errors on behalf of students, so please make sure your submission adheres to all requirements to avoid getting a zero. Thanks!

Honor Code: Solve this assignment without collaborating with classmates (besides your assignment partner), and without consulting external/online resources. **Suspicion of using AI to generate solutions will cause that submission to be reported to Emory's Honor Council.** The assignment is governed by the College Honor Code and Departmental Policy. Remember, any work you submit must be your own; otherwise you risk being investigated by the Honor Council and facing the consequences of that.

Please remember to include the following comment at the beginning of each SQL file you submit:

```
-- THIS WORK WAS MY (OUR) OWN WORK. IT WAS WRITTEN WITHOUT CONSULTING
-- WORK WRITTEN BY OTHER STUDENTS OR COPIED FROM ONLINE RESOURCES.
-- _Student1_Name_, _Student2_Name_
```

References

Bonica Adam. Database on ideology, money in politics, and elections: Public version 2.0. *Computer file, Stanford University Libraries, accessed at https://data.stanford.edu/dime*, 2016.

Adam Bonica. Dime plus.[computer file]. *Stanford, CA: Stanford University Libraries, accessed at https://data.stanford.edu/dime-plus*, 27:2021, 2016.

Adam Bonica. Inferring roll-call scores from campaign contributions using supervised machine learning. *American Journal of Political Science*, 62(4):830–848, 2018.