CS 378 Final Project, due Monday, 12/09. **Due date is not flexible**.

Ground Rules

- Choose one of the four options outlined below based on your interests and the nature of your data
- If you choose options 1 or 2, decide which scenario to apply it to
- If you choose option 4, write up a short proposal on what you'd like to work on
- Make sure your partner is onboard with your selection (!)
- Let the Prof. or TA know which option you're signing up for by the end of class
- If you're choosing option 4, you should speak to the Prof. about your idea today even if you don't have all the details worked out
- Keep in mind that you only have two class periods to complete this project, including today's class
- We won't be able to offer a resubmission option for this project, so it's important that you get started early to give yourself enough time to work through any issues that come up

Option 1: Using text embeddings for data classification

Rather than calling the LLM one row at a time in order to classify a table of records (e.g. when doing sentiment analysis), use a text embedding model to compute the embeddings of the input records and the classification categories and then compute the distances between these embeddings. Find the category that has the smallest distance to the input records. Update the record in the table with their assigned category. Evaluate the accuracy and performance of your results.

Create a new BQ dataset called `fin-[your-domain]`. Choose your input tables from the intermediate layer of your warehouse (`[your-domain]-int`) and write all temp and final tables to the new dataset.

Do all your experimentations in a Colab notebook with SQL. Make use of BQ's built-in functions `ml.generate_embedding()` and `ml.distance()` to create the embeddings and compute their distance, respectively. Annotate your notebook with explanations throughout and include a short conclusion. Name your notebook `[your-domain]-scale-classification.ipynb`.

Code samples available to help you get started.

Note: This project does **not** involve dbt. If you want to do more with dbt, you should consider options 3 or 4, depending on your interests.

Option 2: Using text embeddings for entity matching

Similar idea to option 1, but applied to entity matching. With entity matching (aka entity resolution), the goal is to match records that refer to the same entity. This is a common scenario in warehousing as data extracts can arrive with duplicate records that contain some small variations (and therefore are not detected as the same by `select distinct`) or when datasets that are produced independently refer to the same entities but they are not represented with a standard identifier.

Create a new BQ dataset called `fin-[your-domain]`. Choose your input tables from the staging layer of your warehouse (`[your-domain]-stg`). Write all temp and final tables to the new dataset.

All experiments should be done in a Colab notebook with a mix of SQL and Python code. Create the embeddings with the BQML extension, `ml.generate_embedding()`. Use another built-in function, `vector_search()`, to search the embeddings and find semantically similar records. Use Python to evaluate the output from `vector_search()` and assign common keys to the matching records. Annotate your notebook with explanations throughout and include a short conclusion. Name your notebook `[your-domain]-scale-entity-matching.ipynb`.

Code samples available to help you get started.

Note: This project does **not** involve dbt. If you want to do more with dbt, you should consider options 3 or 4, depending on your interests.

Option 3: Handling deleted records

Extend the work from Project 7 to handle deleted records. Deleted records occur in the context of a warehouse when the operational systems that feed into the warehouse delete their records. This can happen for a variety of reasons like a customer leaves and is no longer considered an active customer, a transaction gets voided, etc.). The goal is to handle deletes in an incremental fashion, the same way that we have been able to process inserts and updates.

Start by learning about dbt's support for hard deletes and how its `invalidate_hard_deletes` parameter works. Design some experiments to test the effectiveness of this parameter on your end-to-end pipeline from Project 7.

For simplicity, you can work from the same dbt project folder as Project 7 and write to the same `inc_` datasets as Project 7. The goal is to show how the snapshot tables can record the deleted records and how the deletes get propagated all the way downstream to the target marts.

No code samples are available for this option.

Hint: In order to simulate deletes, prepare some incremental data that excludes a few records which are present in your warehouse and ingest this data into the raw layer.

Option 4: Choose your own adventure

If your group would rather work on something else and you have an idea in mind, you can write up a short proposal (1-2 paragraphs is sufficient). To get approval, you should make sure that your idea has a narrow enough scope and is doable in two weeks time.

Share your thoughts with the Professor during today's class period and email your proposal to the Professor and TA once it's ready.

Given the short timeline for this project and the holiday week, you should submit your proposal by **Sunday, November 24th**. Any proposals received after that date will likely be rejected due to our limited time.