CS 329E Final Project, due Thursday, **04/25**. **Due date is not flexible.**

Rules

- Choose one of the three suggested ideas or propose your own
- Make sure your teammate is on board with your selection (!)
- Let the instructors know by 5pm today (04/12) which idea you're signing up for
- If you want to propose your own project, speak to Prof. Cohen during today's work period to get approval
- You only have two class periods to work on your project, including this one
- As you start implementing, keep in mind the most challenging technical aspects of the work so that you can share back during Presentation Day on Friday, 04/26

Option 1: Apply change data capture at-scale

Implement a CDC pipeline for each target table in the database using the same logic as Project 8. After applying the changes to the staging tables, perform the standard referential integrity checks, removing any orphan records found. Implement a verification function for each pipeline that compares the record count in staging with the active record count in consumption to ensure that they match. If the counts don't match, return an error to the user. Port your working pipelines to Airflow. The end result should be a one-click workflow that applies CDC on all your target tables. Publish the Colab notebooks and Airflow DAGs to your repo.

Note: This project option does not involve the use of a language model. If you want to work with Gemini, you should choose options 2 or 3 or propose your own project.

Option 2: Apply data enrichment at-scale

Iteratively enrich your staging tables with additional data signals, including replacing missing values with AI generated ones. The generated values should either be based on existing values from the same record or on a collection of values from related records. When working with more complex prompts, keep in mind that the language model's context window is limited to 1 million tokens.

This project is very similar to our work from Project 9, except that we are now broadening the scope as follows:

- Implement 5-10 enrichment scenarios (as opposed to only 3)
- Port the enrichment pipelines to Airflow
- Apply the enrichment to all the records of the table

Begin by prototyping your enrichment pipelines in Colab on a few hundred records. Port to Airflow only once you have vetted each pipeline. The end result should be a one-click workflow

that executes all the enrichments. Publish the Colab notebooks, Airflow DAGs, and ERDs to your repo.

Note: This project option requires a Gemini Pro quota limit of 1000 QPM or higher.

Option 3: Experiment with error detection

The basic idea of this project is to use the language model to detect any errors present in the attribute values of your staging tables. Begin by finding the functional dependencies in your dataset. Some common examples of functional dependencies include the state, country, zip code, category, and age.

Choose three of your functional dependencies which are not keys. For each one, manually replace a few values in your staging table with some erroneous ones. Then ask the language model to detect if an error exists in each record of the table and to suggest the corrected value. Experiment with prompts using zero shot and few shot learning.

Save the detected errors into an auxiliary table. Create the table in your AI staging dataset from Project 9 and name it Error. The table should include the following details:

- the event timestamp (timestamp when the error was detected)
- the name of the table in which the error was found
- the name of the column in which the error was found
- the primary key value of the record in which the error occurred
- the value of the detected error
- the suggested value from the LLM
- the LLM prompt used

Note: if we were evaluating multiple language models, we would also be storing the details of the model used.

Implement your error detection pipelines in a Colab notebook over a subset of records (a few hundred per table should suffice). Once this is done, port your pipelines to Airflow so that you can apply the detection functions to the full table. The end result should be a one-click error detection workflow that covers your three error detection scenarios.

Note: This project option requires a Gemini Pro quota limit of 1000 QPM or higher.