CS 378 Project 4, due Thursday, **10/10**.

Recall our acceptance criteria introduced in Project 2. The relevant sections for this project are reproduced below for convenience. This project assumes that the data in your staging layer satisfies the anomaly types 8, 9, and 10.

Our focus will be to construct an intermediate layer in the warehouse that addresses multiple entity-level anomalies, namely anomaly types 8, 9, and 10. This work will entail refactoring your staging tables such that the identifiers are consistent, each value can be queried in standard SQL, and referential integrity is enforced.

Warning: This project is similar in scope to Project 2 and that is why you have two weeks to work on it. Please don't wait to get started or you will have a difficult time meeting the deadline. You want to start by coming up with a plan for each of your transformations. Feel free to consult with me if you have questions.

Anomaly Type	Description	Applicable to Project	Air Travel Examples
8	There exists a field in any table of the warehouse that stores multiple values in the same cell. The values represent a list of elements for the same attribute.	4	flight_routes sometimes stores a list of equipments in the same cell, airport_businesses sometimes stores a list of menu items in the same cell.
9	There exists two tables in the warehouse which originated from different sources and which have similar data. Moreover, the tables in question use two different identifier systems to refer to the same entity.	4	airport information is repeated across multiple tables in a non-standard way. See for example airportRef and aiportIdent versus airport_id and airport_code.
10	There exists a table in the warehouse that models more than one logical entity. This can lead to storing repeated values within the same table.	4	The flight_delays table has information about airports, airlines, and flight delays. Fields like carrier_name and airport_name shouldn't be in this table.

Data Transformation Strategies

- Create an intermediate area in BigQuery and populate it with the results from your transformations. The transformations should be applied to the staging tables and materialized in the intermediate dataset. **Do NOT mutate the staging area**. All mutations should be applied only to the intermediate area of the warehouse.
- Implement transformations in SQL and Python that resolve at least one anomaly type 8, 9, and 10 in the intermediate layer:
    - Resolve anomaly type 8 by converting the list of embedded values to either an array type in BQ within the same table or moving them to an auxiliary table with proper references to the main table.
    - Resolve anomaly type 9 by creating a universal identifier for a core entity and propagating it to all of its dependent tables.
    - Resolve anomaly type 10 by decomposing a table that models two different entities into multiple tables. To decide what those tables should look like, think about the relationships between the two entities.
- In addition to the three anomaly types, also perform these transformations:
    - Enrich at least one entity with a new attribute using the LLM.
    - Ensure that referential integrity holds across all final tables in this layer. This may require you to delete duplicate records, look up missing values, delete orphan records, and filter out unwanted records.

Implementation Guidelines

- Create a new folder in your repo and name it `project4`. Store all of your artifacts for this project in the `project4` folder.
- Develop a Colab notebook that creates the intermediate layer and performs the appropriate transformations. Name your notebook `4-[your-domain]-int.ipynb`.
- The tables should be stored in their own dataset in BigQuery. Name the dataset `[your-domain]-int` where `int` is short for intermediate. For example, air_travel_int.
- Copy all the tables from the staging area into the intermediate area, even those which don't exhibit any anomalies.
- If you need to split up a complex transformation, you can materialize the intermediate results as a temporary table. Be sure to create the temp table in the intermediate area and drop it after you have finished creating the final table.
- Document your assumptions and design decisions as short Markdown comments in your notebook so that the reader can follow along with your reasoning (and so that you can remember your own thinking in a few weeks time). In particular, be sure to annotate the sections where you are resolving an anomaly type.

- Each table should have a well-defined primary key and child tables should also have proper foreign keys. Create the key constraints both in code as well as in your ERD.
- Reorder the table columns such that the primary key fields appear at the top.
- The first letter of the intermediate table names should be in uppercase. If the name has multiple words, capitalize the first letter of each word and use an underscore between the words. The only exception to this rule are the temporary tables. These should be all in lowercase letters and contain the prefix `tmp_`.
- Most intermediate table names should be in singular form except for junction tables, which can be either in singular or plural. For example, in the Air Travel warehouse, the Airport_Businesses table is a junction table because it models a many-to-many relationship.
- Column names should remain in lowercase across the board.
- Do not carry over the two system columns, `_data_source` and `_load_time`. The lineage of a record in the intermediate layer can get quite complex and is often best represented by a graph; this is beyond the scope of this project.
- Create an ERD that reflects the design of your intermediate tables. Bold all the fields, data types, and entities which have changed from the staging layer. You do not need to prepare a data dictionary.
- Create a submission.json file and upload it to Canvas by the deadline. Only one person per group needs to do this step.

CS 378 Project 4 Rubric
Due Date: 10/10/24

`4-[your-domain]-int.ipynb` is thorough and meets all requirements **-10** incorrectly used SQL commands and/or Python code **-15** missing or incorrect resolution of anomaly type 8 **-15** missing or incorrect resolution of anomaly type 9 **-15** missing or incorrect resolution of anomaly type 10 **-5** missing or incorrect data enrichment **-7** missing primary key constraints and/or logical checks **-7** missing foreign key constraints and/or logical checks **-5** notebook lacks Markdown annotations and is hard to follow **-5** for each empty intermediate table in BQ dataset **-30** did not create intermediate tables in BQ dataset **-5** did not follow naming convention for dataset, tables or columns **-80** missing file	80
`[your-domain]-erd-int.pdf` accurately depicts the intermediate tables schema and logical relationships between tables **-2** for file named incorrectly **-3** for each missing intermediate table **-3** for each missing relationship or key (primary or foreign) **-2** did not bold the changed fields, types, or entities to this layer **-10** ERD not aligned with intermediate tables **-20** missing file	20

`submission.json` submitted into Canvas. Your project **will not** be graded without this submission. The file should have the following schema:
 	
`{`
`  "commit-id": "your most recent commit ID from Github",`
`  "project-id": "your project ID from GCP"`
`}`
 	
Example:
 	
`{`
`  "commit-id": "dab96492ac7d906368ac9c7a17cb0dbd670923d9",`
`  "project-id": "some-project-id"`
`}`
	Required

Total Credit:	**100**