CS 378 Project 6, due Thursday, 11/06.

Objectives

Our high-level objectives are to orchestrate and modularize the data transformations for our warehouse so that they can become more automated and portable. We will do that by standing up a dbt project and converting our staging, intermediate, and mart layers from Colab to dbt.

Conversion Strategies

Go through our staging, intermediate, and mart Colab notebooks and identify the SQL and Python blocks which mutate the database. Follow the conversion strategies detailed below based on the type of mutation in question:

- For create table statements, which can be in the form of create-table-as-select, insert-table-as-select, and pandas Dataframe writes, convert them to a dbt model. SQL statements should be translated into SQL models while Python blocks should be translated into Python models:
    - SQL models comprise of one or more common table expressions, followed by a select statement for dbt to evaluate and persist as a table.
    - Python models contain data manipulations that can't be easily done in SQL, namely interacting with the LLM and converting the output to a PySpark Dataframe. Note: Python models get executed on a Dataproc cluster which uses a PySpark runtime, hence the need to convert from Pandas to PySpark Dataframe.
    - All models that source from the raw layer should call the source() function to read the raw tables.
    - All models that source from the staging layer or above should call the ref() function to read the staging or intermediate tables, whether they are the final table or just a temp table.
- For any DML statements, copy them into post-hooks or rewrite them into select statements.
- For uniqueness, not null tests and referential integrity tests, convert them to yaml format and add them to in `models/intermediate/schema.yml` using dbt's unique and not null and relationship constructs. Note that in order to test for the uniqueness of a combination of fields, an additional macro is required (for more details on macros, see hints section).
- Copy any drop table statements to remove temp tables into post-hooks and attach them to the final model so that they get deleted only after the final table has been created. For example, if `tmp_airports` is a temp table that persists some intermediate results and is used in the creation of the `Airport` table, add the drop table statement into the post-hook of the Airport model.

Implementation Details

- Materialize all models as BQ tables. This is defined in the dbt_project.yml file, which also contains the folder structure and BQ datasets.
- Follow the naming convention of `dbt_[your-domain]_stg`, `dbt_[your-domain]_int`, and `dbt_[your-domain]_mrt` for naming your datasets. Follow the dbt_project.yml example file.
- Specify your raw tables in a `models/staging/schema.yml` file. This will allow you to refer to them using the source() function from your staging models. Follow the schema.yml example file.
- Run the dbt run command to compile your models. This should not only create the tables, but also create the primary key constraints on those tables. At this time, there is a bug in the bigquery adaptor that is preventing the foreign key constraints from getting created. Don't worry if they don't show up so long as you specify them in `schema.yml`.
- Run the dbt test command to run your data integrity tests, namely uniqueness, not null, and relationship checks.
- When you are ready to commit your code, create a new folder in your repo for this project and name it `project6`. Copy your top-level dbt project folder (e.g. air_travel) into the `project6`. You can exclude the dbt logs and target folders from your commit.
- Use custom schemas to specify the naming of your BigQuery datasets. This comes in the form of a macro, `generate_schema_name`, which you can copy from the snippets repo, and place in your macros folder.
- Use the additional macro, `unique_combination_of_columns`, to test for the uniqueness of composite primary keys. You can copy the code from the snippets repo and place it in your own macros folder.
- Generate one or more lineage graphs that capture the dependencies between the models from your warehouse. To do that, you'll first need to compile the documentation for your warehouse and then bring up the UI, just like we did during the setup. Take a screenshot of your full lineage graph. If your graph has > 30 nodes and is difficult to visualize in its entirety, take additional screenshots of each layer (staging, intermediate, and mart). Save the screenshots in a new folder under the `project6` folder, called `lineage`.
- Create the usual submission.json file and upload it to Canvas by the deadline. Only one person per group needs to do this step.

Hints

- To get started, run the `dbt init [your-domain]` command to generate a new dbt project for your warehouse. You'll need to modify `.dbt/profiles.yml` just like we did during the setup. From there, you should create the proper directory structure under the models folder and then modify the default dbt_project.yml. You should also create the two macros and the `models/staging/schema.yml` file.

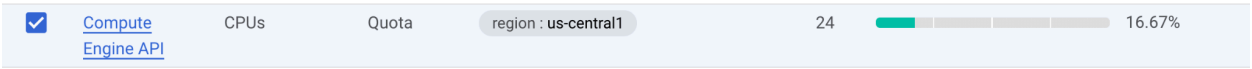
- I suggest keeping around your temp tables in BigQuery until you have verified the final models that they need to feed into. Otherwise, you will be wasting time waiting for dbt to compile and recompile the temp models.

- To speed up the `dbt run` command, you can increase the number of threads from 1 to 3 in your `.dbt/profiles.yml`. This will allow dbt to execute up to 3 models concurrently without violating dependencies.

- If you are renaming a model or changing the schema of an existing model, use the `dbt run --full-refresh` option. More details here.

- For the primary key, foreign key, uniqueness, not null and relationship tests, I would suggest copying one of the `schema.yml` files from the snippets repo (such as this one) and using it as your starting point.

- If you get an "Insufficient quota" error, whether it be CPU or Persistent Disk, go to the Quotas page and request an increase for the resource in question. For CPU, I would recommend requesting 48 CPUs in the us-central1 region, which would be double from the current allocation. For storage, I would recommend also doubling the volume of available disk to 8 TB. These requests should get auto-approved within minutes. If they don't, please contact me as soon as possible.



- To create the lineage graphs, run `dbt docs generate` followed by `dbt docs serve --port 8181` on the machine where you are running dbt. If you are running dbt on a VM, you'll also need to bring up a local terminal, run `gcloud init`, and then set up the ssh tunnel to connect to the VM. You can use this command to create the tunnel:
  `gcloud compute ssh dbt --project [GCP-PROJECT_ID] --zone us-central1-c --
  -NL 8181:localhost:8181`
  Be sure to replace `[GCP-PROJECT_ID]` with your own project. Then, open a browser on your local machine and navigate to `http://localhost:8181`. To view the lineage graph, click on this icon:  on the bottom-right corner of the home page. The graph should look similar to the image below:


Lineage Graph
air_travel_raw.airlines
airlines
air_travel_raw.countries
countries
Country
tmp_airline_countries_unmatched
tmp_airline_countries_mapped
tmp_airline_countries_matched
Flight_Routes
Airline
flight_delay_frequency_by_airline
air_travel_raw.flight_delays
flight_delays
tmp_flight_delays
Flight_Delays
flight_delay_lengths_by_airline
air_travel_raw.flight_routes
flight_routes
tmp_flight_routes
tmp_flight_routes_with_id
airport_travelers_by_month_year
air_travel_raw.aircrafts
aircrafts
tmp_aircrafts
Aircraft
Route_Equipment
tmp_routs_equipment
tmp_route_equipment_unmasked
tmp_airports_filled_out_icao_non_us
tmp_airports_missing_icao_non_us
tmp_airports_merged_icao_non_us
tmp_airports_filled_out_icao_us
air_travel_raw.airports
airports
tmp_airports_missing_icao_us
tmp_airports_merged_icao_us
Airport
tmp_airports_with_icao_non_us
tmp_airports_with_icao_us
tmp_tsa_traffic_all
tmp_tsa_traffic_duplicates
TSA_Traffic
airport_travelers_by_weekday
air_travel_raw.tsa_traffic
tsa_traffic
Airport_Businesses
air_travel_raw.airport_businesses
tmp_airport_businesses
airport_businesses
tmp_business
tmp_business_ranked
Business
business_category_popularity_by_airport
Menu_Items
food_items_popularity_by_airport
highest_rated_airports_us
highest_rated_airports_world
air_travel_raw.airport_reviews
airport_reviews
tmp_airport_reviews
tmp_airport_reviews_with_sentiment
Airport_Review
lowest_rated_airports_us
lowest_rated_airports_world
resources
All selected
packages
air_travel
tags
untagged
--select
...
--exclude
...
Update Graph

CS 378 Project 6 Rubric
Due Date: 11/07/24

dbt project folder is thorough and meets all requirements **-5** for each missing table or model file in staging, intermediate or mart datasets **-4** for each empty table in staging, intermediate or mart datasets **-2** for each failing uniqueness, not null or relationship test **-2** for each missing primary key or foreign key constraint from `schema.yml` **-2** for each missing primary key constraint from a final intermediate table **-2** for each model file not using `source()` or `ref()` **-3** for not following dataset naming convention **-90** missing dbt project folder under `project6`	90
Lineage folder contains one or more screenshots of the model dependency graph **-1** for each missing model or dependency that doesn't correspond to the model files in the repo **-2** model names are not legible in the provided screenshot(s) **-10** missing lineage folder under `project6`	10

`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**