CS 378 Project 7, due Thursday, 11/21.

Objectives

Our high-level objectives are to extend the dbt implementation of our warehouse to handle incremental updates or change data. This includes recording the history of the updates so that we can go back to a point-in-time and recreate the state of the tables as they existed in the past.

Strategy

- Go through your marts and choose a subset of them which source from the same tables and whose source data has changed since you last pulled it in August (refer to your data dictionary for source details). You can ignore all the other tables in your warehouse for the rest of this project.
- Collect the new file(s) for your changed data and upload them to your existing bucket in GCS. These should be in csv or json format.
- Configure snapshot tables using the `_load_time` field. This will be used to detect when a record in the table has changed. We want to snapshot each final table per layer from raw to mart. To do that, we will include the `_load_time` field in all the tables except for the marts. Note that we do not want to snapshot any temp tables.
- Configure incremental models using a unique key and merge strategy.
- A model should source its data from the previous layer's snapshot table as long as it is the first model in the layer. This dependency chain is shown below:

  raw_source -> raw_snapshot -> stg_model -> stg_snapshot -> int_tmp1_model -> int_tmp2_model -> int_tmp*n*_model -> Int_model -> Int_snapshot -> mrt_model -> mrt_snapshot

  Note: Int_model cannot source its updates from stg_snapshot because it depends on int_tmp*n*_model, which does not have a snapshot given that it is only a temp table.

- dbt treats the snapshot table creation/updates as separate from the model creation/updates and does not orchestrate these two functions. As a result, we will provide our own orchestration in the form of simple shell scripts.

Implementation Details

- Create a new profile and dbt project folder for project 7. Copy into it the relevant code from your project 6 folder (i.e. the model files and yaml files that are needed to refresh your chosen mart(s)).

- Create a `dbt_project.yml` file with incremental materialization and merge strategy. See sample.
- Follow the naming convention `inc_[your-domain]_raw, inc_[your-domain]_snp, inc_[your-domain]_stg`, `inc_[your-domain]_int`, and `inc_[your-domain]_mrt` for naming your datasets for this project.
- Copy the relevant raw tables from your original raw dataset to `inc_[your-domain]_raw` with a simple CTAS statement or with the Copy function from the BQ UI. Edit the schema to include the default values for `_data_source` and `_load_time`.
- Specify your raw tables as sources in the `snapshots/schema.yml` file. See sample.
- Create the snapshot tables. See samples.
- Reference your raw tables from the raw snapshot tables using the `source()` function.
- Update your staging models to reference the raw snapshot tables (instead of the raw tables directly). You'll also need to specify a unique key and use the condition `where dbt_valid_to is null`. See samples.
- Snapshot your staging models, this time the source of your snapshot staging tables will be the staging models. See samples.
- Create your intermediate models. The first ones in the chain should source from the snapshot staging tables, the remaining will source from the temp models. In order to merge into all the tables (as opposed to re-creating them from scratch each time), you will continue to specify a unique key and a filter condition to select the new records. The models that are at the start of the chain in this layer will use the `where dbt_valid_to is null` condition while the others will use the is_incremental macro. See samples.
- Snapshot your final intermediate models. See samples.
- Create your mart(s) by sourcing them from the intermediate snapshot tables. See samples.
- Run through your pipeline using the original data you had collected at the start of the term. You'll use the dbt snapshot command to compile your snapshot tables and the usual dbt run command to compile your models. Capture the dbt snapshot and run commands in a simple shell script. Name it `orchestrate_first.sh`. See sample.
- Collect some change data for your datasets and upload this data to your existing bucket. Place it in an `incrementals` folder with subfolders underneath that denote the name of the data producer.
- Prepare a Colab notebook that ingests the change data into your raw tables. Name the notebook `7-[your-domain]-change-data.ipynb`. See sample.
- Run your notebook to populate your raw tables in the `inc_[your-domain]_raw` dataset.
- Create a second shell script called `orchestrate_subsequent.sh` that contains all the dbt snapshot and run commands needed to process the change data from raw to mart such that the mart table(s) are refreshed and snapshotted. See sample.
- Run `orchestrate_subsequent.sh` and verify that the change data has been merged into the snapshots and models without creating duplicate records.
- Run dbt test to check that your data integrity tests still pass. If any of them fail, you should debug and patch the models until they all pass.

- When you are ready to commit your code, create a new folder in your repo for this project, naming it `project7`. Copy your top-level dbt project folder (e.g. air_travel) into it. You can exclude the dbt logs and target subfolders from your commit. Place the notebook file in the root of your `project7` folder.
- Generate one or more lineage graphs that capture the data flows from raw to mart. As before, you'll need to generate the catalog for your warehouse and then bring up the UI. Take a screenshot of your full lineage graph and subgraphs if the full graph is not legible. Save the screenshots into a `lineage` subfolder under `project7`.
- Create the usual submission.json file and upload it to Canvas by the deadline. Only one person per group needs to do this step.

CS 378 Project 7 Rubric
Due Date: 11/21/24

dbt project folder is thorough and meets all requirements **-5** for each snapshot or model table missing from the raw, staging, intermediate, mart or snapshot datasets **-4** for each empty table in staging, intermediate, mart or snapshot datasets **-2** for each snapshot or model file not referencing the expected data source **-2** for each snapshot or model file missing a unique key **-2** for each snapshot or model file missing an appropriate condition that filters rows where `dbt_valid_to is null` or uses the `is_incremental()` macro **-5** missing `orchestrate_firt.sh` and/or `orchestrate_subsequent.sh` **-2** for each missing primary key or foreign key constraint from `schema.yml` **-2** for each failing uniqueness, not null or relationship test **-2** for each missing primary key constraint from the final intermediate table **-3** for not following dataset naming conventions **-90** missing dbt project folder under `project7`	90
Change data stored in bucket and ingested into raw tables via Colab notebook **-2** notebook does not append the change data to the existing raw tables **-2** raw tables are missing proper values for `_data_source` and/or `_load_time` **-5** notebook is missing from the `project7` folder **-5** change data is missing from the bucket or `incrementals` folder not found	5
Lineage folder contains one or more model and snapshot dependency graphs **-1** for each missing model or dependency that doesn't correspond to the model files in the repo **-1** for each missing snapshot or dependency that doesn't correspond to the snapshot files in the repo **-2** model names are not legible in the provided screenshot(s) **-10** missing lineage folder under `project6`	5

`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**