INFO 290T Data Engineering
Project 5: Data Pipeline
Yin Qiu

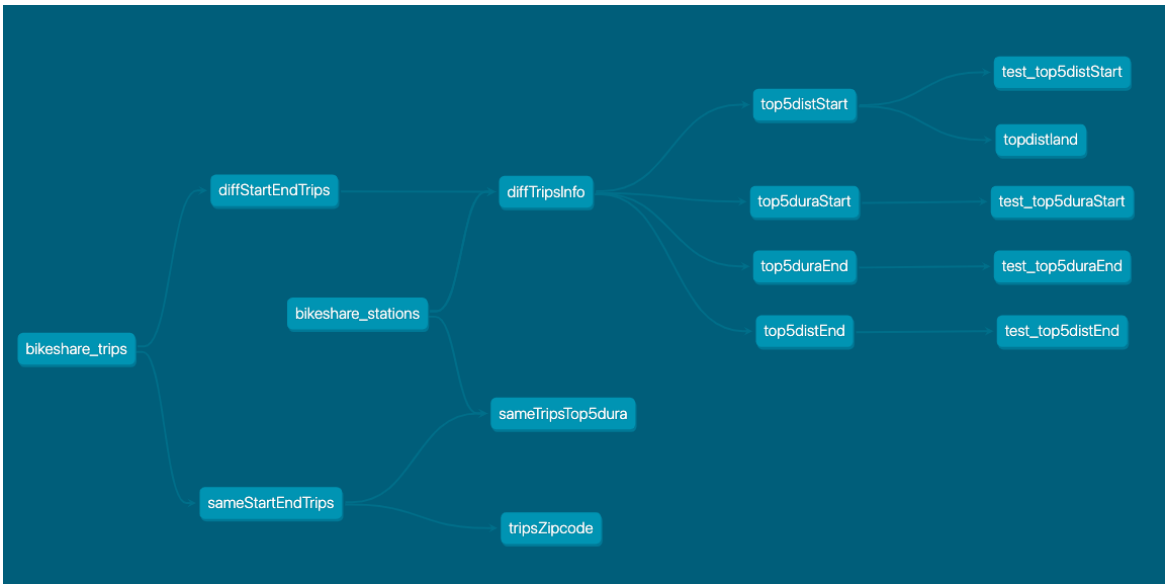
1. Overview

This project is using a BigQuery Public Dataset on San Francisco Bikeshare. I connected my dbt project to the BigQuery dataset, as guided in the dbt official tutorial. I am particularly interested in transport topics, so I applied the filter of transportation and found this bikeshare dataset for San Francisco. I think it would be great to explore a topic and place that I am familiar with so that I can test some hypothesis developed through my experience.

The initial goal of the data pipeline was to understand the different trip patterns in bikeshare in San Francisco. Although there are many possible ways to categorize these bikeshare trips, I am interested in knowing the differences between the trips with the same start and end station and the trips with different start and end stations. My hypothesis is that trips with different start and end stations will have longer duration time for the trips than the trips that start and end in the same station. I will focus more on the top 5 duration trips for each station in my data pipeline.

2. Model Structure

It took me about 4 days to write my data pipeline, and it took my computer about 38 seconds to run the pipeline. I included 12 models for the data pipeline, each has descriptions in the schema yml file. I also developed 4 data quality tests and 21 schema tests. In the end, there are 2 table models, 2 incremental models and 12 view models. The data pipeline can be grouped in to 6 steps. I will explain the relevant models briefly in each step. Below is a Lineage DAG diagram for an overview of the models:



Step 1: Make a copy of the two original tables in BigQuery Public Dataset

Relevant models:

- bikeshare_trips (table)
- bikeshare_stations (table)

These two data are the foundation of the data pipeline. Since the dataset are on the cloud, I have to first make a copy of the two datasets as tables in dbt, then I can make reference to them to build my pipeline further down the road. What I did are simply projection of all data from the two relations.

Step 2: Build two incremental views based on the tables in Step 1, one for trips of the same start and end station, and one for trips with different start and end stations

Relevant models:

- sameStartEndTrips (incremental)
- diffStartEndTrips (incremental)

Assuming the database might be updated each time when we request, we need to build the incremental models to make sure we add new data if the start_date is later than the latest start_date we had in the previous request. I used a where clause and the MAX function for the incremental condition.

The two incremental models followed the same patterns of two mini steps using CTE. The first mini-step is use projection of start_station_id = end_station_id and start_station_id != end_station_id, respectively, to divide the dataset into two. The second mini-step is to add three new columns for date information (extract year, month and day of week) for future analytics. I think this allows us to do aggregation by time to analyze the patterns of different types of trips when we expand the data pipeline in the future.

Step 3: Build two views based on views in Step 2, both have Trips data left join with Station data to add the landmarks for start and end stations; build an additional view for trips with same start and end to understand the trips and stations better.

Relevant models:

- diffTripsInfo

For trips with different start and end station, I also join geographic point location and calculate the distance between start and end stations, so that later I can find out the trips with longer distance by station id.

- sameTripsTop5dura

For the trips with the same start and end station, since the distance between start and end station is 0, I do not compute the distance between stations. In this step, I use DENSE_RANK window function PARTITION BY start_station_id (start and end is the same so it doesn't matter)

and then order by duration_sec in descending order. This gets us the top 5 trips in duration by each station. I will do the same thing for trips of different start and end in later step in order to have some comparison.

- tripsZipcode

For the trips with the same start and end station, I went a little further. I aggregated the sameStartEndTrips from Step 2 by zip code to find out the count of trip_id, the count of distinct station_id involved, and the average duration_sec. These can help us understand in which zip codes these round trips mostly occurred, how dense the stations are, and how long on average the trip durations are.

Step 4: Build four views based on diffTripsInfo in Step 3, each shows the top 5 distance trips or top 5 duration trips grouped by start station id or grouped by end station id

Relevant models:

- top5distStart
- top5duraStart
- top5distEnd
- top5duraEnd

Focusing on the trips with different start and end stations, it will be interesting to check not only the duration but also how far away the destination is from the origin. I also used DENSE_RANK window function PARITION BY start or end stations to have a more comprehensive picture of where the longer trips (time-wise and geo-wise) come from or go to. It doesn't give us an absolute answer to the questions asked in the hypothesis, but this data pipeline allows us to do the comparison between the top5duraStart/top5duraEnd with the sameTripsTop5dura from Step 3. This can become our key takeaway.

Step 5: Build a view for start and end landmark pair that have the longest distance trips based on top5distStart in step 4

Relevant model:

- topdistland

This model is used as additional description about the longest trips where the different start and end trips start and end, but group by pairs of start landmark and end landmark. It is possible that these trips start and end in different stations but the stations located in the same landmarks, e.g. a trip start in one station in San Francisco and end in one station in San Francisco. This rolls up to the landmark level from the station level, and provide us more information about where the longer trips start and end.

Step 6: Developed 4 data quality tests (SQL models) to test the four top 5 models for different start and end models, making sure not return any rank higher than 5 for each station

Relevant models:

- test_top5duraStart
- test_top5duraEnd
- test_top5distStart
- test_topdistEnd

I chose them as the data quality tests to make sure the returning results are not any above 5 in ranking of distance or duration. I also made sure there are no replicate results for top5 distance for each station, because there can be many trips logging with the same distance between stations but less likely to have trips logging with the same time duration between two stations.

3. Reflection

The outcome of my pipeline does not match my initial goals exactly. At first I was thinking to use duration or distance per trip per station to measure and compare whether same start-end trips or different start-end trips have longer trips. However, once I dived into the data I found out we don't have a clear cut metric to answer such question. For example, a rider can possibly spend a long idle time in the trip that extends its trip duration but doesn't go far; a rider can also ride a much longer way than the distance between the start and end location, so the distance between two stations either tell us much for certain. Besides, for trips with the same start and end station, meaning a round trip, we don't even have the distance to gauge how far away the rider has gone. In that case, I think I don't have enough information to test my hypothesis. I will need further data and other techniques to find out the answer.

If I have more time, I will revise my hypothesis and research question to something more feasible to answer with the current dataset. Perhaps something related to the time series travel patterns, or something related to the consumer types.

My experience with the dbt tool is relatively positive. I think I will further try out connecting it to a local PostgreSQL database next time, and check out a dataset I'm more familiar with in my past project. I really like the dbt tool has many integration possibility with different data warehouse and can make use of existing SQL queries for a data pipeline that scales up. Compare to a graphical UI like Trifacta, I will actually prefer to write my own SQL models and dbt specs, because it may be easier to debug. I will hope to make use of some of the visualization functions to assist debugging though.

4. Link to Code

Github repository: https://github.com/qyinhelena/dbt-project

5. Sources

BigQuery Public Dataset – San Francisco Bikeshare:
https://console.cloud.google.com/marketplace/product/san-francisco-public-data/sf-bike-share?filter=solution-type:dataset&filter=category:transportation.
Installation dbt CLI, configure profile and dbt_project:

https://docs.getdbt.com/dbt-cli/installation
https://docs.getdbt.com/dbt-cli/configure-your-profile

Creating repo, create a project, connect to BigQuery, perform first dbt run, commit changes:
https://docs.getdbt.com/tutorial/create-a-project-dbt-cli
Adding schema tests, adding data tests, perform dbt test:
https://docs.getdbt.com/reference/declaring-properties
https://docs.getdbt.com/docs/building-a-dbt-project/tests
https://docs.getdbt.com/tutorial/test-and-document-your-project
Deploy the project:
https://docs.getdbt.com/tutorial/deploy-your-project