Hi Ben,

This was quite a fun data challenge.

**Some Assumptions I have inferred:**

You can only SKIP on the DISCOVER

You can only LIKE on the DISCOVER

You can only LIKE with a COMMENT on the DISCOVER

You can REPORT on all 3 Screens

**My pipeline in Airflow using a BATCH methodology (in order):**

Raw file arrives on s3

Push files on SQS Queue

Read S3 files from SQS Queue and if batch of files reach a certain volume threshold, move to EMR

Bootstrap EMR with HIVE

Use Scoop to pull Master data from Hinge Relational DB’s

Build HIVE Tables in EMR for Master Data + Raw Data

Process/Validate/Transform/Enrich to Master Data will be done in HIVE

Build Facts/Aggs/Derived Dims in HIVE

Copy data back to S3 from EMR in appropriate partitioned folders using .gz compression

Load data into Redshift Dimensional Model

**New File Detection Approach:**

Use a marker file system which is a dummy file created by the process. Check marker files and check the number of files in the bucket, if you don’t see a marker file corresponding to a particular file, then pick it up, place on the SQS Queue, and create a marker file for that file.

**Improvements and things still need to do:**

Ensure Fact/Dim tables have the correct dist/sort keys in Redshift. Build the best keys for these to ensure skew is very low, with even distribution across all nodes.

Load the Fact table [fact\_event] from [event\_staging] using a HIVE Window Function to determine which screen\_action\_key is used at the connection level.

If we want to get more closer to real-time/event driven ingestion, this BATCH architecuture will not work. What happens if a file goes missing? Our order will go astray..

**Analytical Query Questions (see analytical\_queries.sql file)**

The queries are working. However, they need to pull data from the FACT tables and not the raw/staging data. That can be done at a later time. These queries would sit in Looker

**Caveat Questions:**

1) What if it were the case that every day, more files is added to this S3 bucket, and that you can’t even predict their exact names. How will you make sure that you will catch these and include them in your final model?

*Answer:*

--We would not load the file until the one before it has been loaded, or arrives. Or, we process out of order and re-process everything from the late file arrival date through today.

2) What kinds of tests will you put in place to make sure that the final model is accurate? What about to ensure that the source data is accurate?

*Answer:*

--There would be pipelines that run and perform UAT integrity checks on the source files, multiple times a day.

--Data Latency in Files will be checked and we will provide a Data Confidence Dashboard

--Build audit tables in Postgres to monitor files through the various stages of the workflow and critical pipeline failures/success. This can be displayed on a DataDog Dashboard for DevOps

--We would parse the CSV and ensure columns are in correct order, via the headers.

--Examine the order of files via filename timestamps

--Look at filesize and ensure its within our specified threshold. If too small or large, send notification to DevOps.

3) A single rating\_type can have multiple semantic meanings. For example, a rating\_type of 3 either means “Remove this person from the list of people I can send likes to”, “Reject this person’s incoming like”, or “Remove this person from my list of matches”, depending on the preceding ratings between a pair of users. It's important that your model be able to distingush between these different meanings.

*Answer:*

1. Built the preprocessing [public.event\_staging] table exactly for this purpose. Build a WINDOW Function using LAG and LEAD, when querying this table to figure out (screen\_action\_key) to place into the base fact [fact\_event] table.