Overall with this assignment I wanted to show my technical ability to build a data pipeline using SQL and Python coding. I created a demonstration Redshift and S3 environment inside of publicly assessable VPC and used my local machine for execution of the code. The three scripts are logically divided between the loading of raw, transformation of raw data, and final loading of raw data into the reporting schema. I had the following insights and practices into my approach:

**Observations & Approach:**

- I saw one key rating fact missing the data that could be deduced from the data set – when a subject receives a like from someone. This can occur later than when the like is sent by the the player. In the transformation layer, I deduced this fact by transferring the likes a player sends into a separate table to be matched up with when the subject is active on the application.

- Taking the assignment assessment on action 3 of blocking and also extending that to action 4 of reporting someone, I added an addition column to the data to track rating\_activity to provide context to the rating received. For example, rating 3 will show a rating activity for whether the block was received after matching, after receiving a like from a person, or simply because the person did not want to see them again on their feed.

- I noticed that the rating type of 0 was very dominant in the dataset, in some cases there are rating facts showing a skip between two players multiple times within a span of a half hour. I created a final fact table that reduced these 0 ratings and consolidates them to the 15 minute mark. This seemed to the be best way of reducing the large dataset with the least loss of information.

**Technical Design:**

Overall, I tried to follow best principles of Redshift design by keeping de-normalized tables throughout the flow of the data processing, utilizing S3 when possible to perform COPY/UNCOPY commands that take advantage of Redshifts parallel processing paradigm, and focused upon approach column compression, sort key selection and loading order of the data. There are no foreign keys or primary keys because I did not de-normalize the model at all. In practice with Redshift, it’s better to focus on compression and sorting rather than normalization for performance. This does take a truly ELT approach and pushes all of the data transformation onto the database rather than onto Python or another NoSQL solution. There are pros and cons to this approach overtime, but the scaling and processing power of Redshift is hard to beat if the data transformations can be modeled as sets rather than procedures by the data engineers.

One area where I did use Python instead of pure SQL was generation of a UDID called a pair\_id to make it easy to track the interactions between two people. I created a function as a Python UDF that takes the player\_id and subject\_id and returns a unique value for a pair\_id. This function has a property such that f(player\_id, subject\_id) = f(subject\_id, player\_id) always. It was important to me that I add this to the data model because overtime we will want to track the interactions between two people and perhaps use insights from the Data Science team to form dimensional clusters on the types of two people to be most likely or not likely to match.

Finally, I designed the processes to be independent of each other by placing key process indicators within the transformation tables. In theory I could run all three of these scripts continually without conflicts or dependency management. Load times were reasonable an under one hour to process end-to-end

I invite you to read through my code as I have commented in detail all my thought processes and design considerations. I also have a sample Redshift cluster and S3 bucket setup that you can examine. I also have include some visuals I created in Tableau around like metrics over time and also by hour of the day.

|  |  |
| --- | --- |
| URL to connect to the cluster database with a JDBC client. | jdbc:redshift://hinge-demo-cluster.codrsvjs7qnu.us-east-1.redshift.amazonaws.com:5439/dev |
| URL to connect to the cluster database with an ODBC client. Note that you must replace the password with the Master User Password. | Driver={Amazon Redshift (x64)}; Server=hinge-demo-cluster.codrsvjs7qnu.us-east-1.redshift.amazonaws.com; Database=dev; UID=hingeadmin; PWD=mO0ZZqeV8z8h; Port=5439 |