Design Doc – Data pipeline from S3 to auroradb

**Aim:** To load tables in Auroradb using S3 csv files.

**Flow Diagram:**

AWS S3

Lambda

Auroradb

When source csv files are placed in S3 bucket (schoolfuel-{entity\_id}), it will trigger a lambda function (schoolfuel-{entity\_id}-lambda-s3-to-auroradb-load) which load data to tables in auroradb.

**Development Components:**

**Tables:**

Dimension Tables:

1. Entity table
2. School table
3. Student table
4. Teacher table
5. Parent table

Fact Tables:

1. Student\_performance table
2. Student\_Discipline table
3. Student\_Health table
4. Student\_Attendance table
5. Teacher\_Performance table
6. Feedback table

The table’s structure can be found in the following document.



In addition to the columns mentioned the above document we will be adding an additional column “Process\_date” for all the tables with datatype as timestamp. “Process\_date” column consists of the timestamp on which the record has been inserted into the table. One way of achieving this is by using current\_timestamp() in insert statement.

Update tables:

Currently the assumption is that the client (school’s) will send files once a month. If in case client wants to update specific data in a fact table, they are expected to send entire file again along. In such cases, for fact tables, we can use process\_date columns to delete existing records of that month and reinsert the records.

In dimension tables we will not get full file every month but will only get updated or new records. So, for Dimension tables we can use primary key to delete and reinsert the records when ever we get a file.

**Python Scripts:**

Script to load from S3 to Aurora

1. Make sure script is in small modules
2. Let naming convention of main handler lambda script be sf\_s3\_to\_aurora\_load.py
3. Let naming convention of each module be sf\_{table\_name}\_table\_aurora\_load.py. In each module we can place script to load specific table.
4. Initially script should check if there are any existing records with current process\_date (For Fact tables). If exists delete them and print the deleted record count in logs. For Dimension tables use primary key do to the same. Reinsert the new or updated records.
5. Print number of records inserted into the tables and compare it with number of records we got from the source file. If the number doesn’t match throw an exception with error message.
6. Only commit after delete and insertion is successful.
7. Catch exceptions in python script if any errors occurred.
8. Add 3 retries if the lambda if there is any error due to connection to the database. Time interval between each retry can be increased gradually.