Project Abstract, What Mic Should I Use?

Hello and thanks for checking in! Having recently completed IBM’s Data Engineering Professional course, I’m eager to apply what I’ve learned to a personal project involving a critical tool of my past trade as an audio engineer: microphones! I’ve only just begun, so there’s not much to see yet, but I’ll be adding to this repository regularly and encourage you to check in occasionally for updates. Because I’ll be working in an environment that is local to my machine, I’ll be using screenshots and pdf’s of reports generated with Power BI to document progress.

My goal is to develop programs, design and develop a database/data warehouse, and automate pipelines to generate source data for an analytics dashboard and Apache Spark machine learning algorithm to answer the question, “What mic should I use?” The intended specific context is the amplification/reinforcement of live music. Because the means is more important to me than the end (at least for now), I make no promises that the results will actually provide even remotely scientific evidence of what mic should be used for what instrument in any given situation! Most of the data will be at least partially, randomly generated for lack of an extant source of data available as fuel for my idea. However, I hope to involve the community of audio engineers I’ve gotten to know over the years to ultimately provide a resource of value to the audio community.

To begin, I’ll outline my plan for the basic components of the project:

1. An ERD created with Postgres utilizing a star schema (possibly snowflake schema?), along with a corresponding .sql script.
2. A MySQL staging database/data warehouse with the above Postgres .sql script as schema source.
3. A Python program to randomly generate data for analysis on a daily basis. This program will run via Apache Airflow, i.e. a ‘DAG’, or ‘directed acyclic graph’, and update the staging database with the new records.
4. A DB2 production database/data warehouse. This database will update daily from the staging database through a Python script via Apace Airflow (i.e., a separate, 2nd DAG).
5. An analytics dashboard connected to the production database. I haven’t chosen the platform just yet: IBM’s Cognos or Microsoft’s Power BI.
6. An Apache Spark job, likely triggered daily by a 3rd Airflow DAG(?), that will utilize machine learning to make a classification prediction regarding what mic should be used for what sound source in what situational context. For the moment, I plan to code this with Python in a Jupyter Notebook running on IBM’s Watson Studio in a serverless context… the job will spin up, run, and shut down once per day.

* David

Progress Update, 2.28.2023

Good progress thus far! The repo currently contains the following items:

1. A screenshot of an ERD created in pgAdmin (postgres) outlining the schema for the data warehouse. The structure is that of a snowflake schema composed of the primary table ‘factResults’; ‘dimSource’, ‘dimMics’, ‘dimBand’, and ‘dimVenue’ as the dimensions; and ‘flakeMicUsed’ and ‘flakeMembers’ as sub-dimensions of ‘dimSource’, ‘dimMics’, and ‘dimBand’.
2. ‘what-mic-schema.sql’ — this will be used to build the schema of the staging data warehouse (MySQL) and production data warehouse (DB2).
3. ‘records\_generator.py’ — this is the Python program I’ve developed to populate the staging database on day one. It creates all the data necessary to populate the tables as outlined in the ERD as .csv files. It uses a minimum of libraries: ‘random’ and ‘pandas’ (for DataFrame generation).

Other notes: I’m using an Ubuntu-based virtual machine (22.04.2) for development along with two conda environments: one for python development using Spyder as the IDE and one for Apache Airflow (which won’t play nice with Spyder!) to automate pipelines. I’ve also tested connectivity with the staging data warehouse and production data warehouse by moving some dummy data around with the Python libraries ‘mysql-connector-python’ and ‘ibm\_db’. All is working as expected thus far!

Progress Update, 3.04.2023

Lots of progress this week! All new files are located in the folder marked with the progress update date (03.04.2023). Here’s what’s included:

* generator\_functions.py: this acts as a module for import purposes
* records\_generator\_alt.py: this uses generator\_functions.py and creates the first run of records for all relevant tables to be loaded into the data warehouses (staging and production)
* init\_data\_warehouses.py: this loads all first-run records created with records\_generator\_alt.py into both the staging data warehouse and the production data warehouse. Both of these programs are run ‘manually’, i.e. not triggered by an automated process
* new\_records\_generator\_alt.py: this is used to create all records beyond the first run. Only records for 4 tables are generated (dimFactResults, dimBand, flakeMembers, and flakeMicUsed) as the other tables do not change. This program is run through an automated pipeline.
* update\_staging\_dw.py: this is used to load newly created records from new-records\_generator\_alt.py into the staging data warehouse (MySQL). This program is run through an automated pipeline.
* update\_production\_dw.py: this is used to load all new records from the staging data warehouse into the production data warehouse (DB2). This program is run manually. Concept: the data engineer checks the staging data warehouse each day to ensure all is well. Once verified, the production data warehouse is updated.
* create\_and\_load.py: this program is written specifically for Apache Airflow. It is the program that runs new\_records\_generator\_alt.py (‘create’) and update\_staging\_dw.py (‘load’). The program is run once per day without intervention from the engineer.
* what-mic-schema.sql: this is an update to the previous what-mic-schema.sql with changes specific to MySQL syntax. It is used as the source schema for the staging data warehouse. It is also used (with slight modification) as the source schema for the production data warehouse.

All in all, this is what I’ve got thus far in pipeline format:

generate records -> insert records into the staging and production data warehouses -> generate new records (daily) and load them into the staging data warehouse (automatic process through Airflow) -> manually check the staging data warehouse (daily) -> manually load any new records from the staging data warehouse into the production data warehouse

Up next: connecting IBM Cognos or Power BI to the production data warehouse for analytics