UDACITY

‹ Return to "Data Engineering Nanodegree" in the classroom

Data Modeling with Postgres

REVIEW

CODE REVIEW 1

HISTORY

Meets Specifications

Dear student,

Congratulations on completing the first project of the Nanodegree. You have put a lot of effort and I am sure you have learned a lot throughout the journey. Great job 👍 The submission shows that you have understood the concepts of dimensional modeling in Postgres pretty well. I have not added additional comments as I did the last review as well and put my comments.

All the very best for the upcoming projects. 😄

Table Creation

The script, `create_tables.py`, runs in the terminal without errors. The script successfully connects to the Sparkify database, drops any tables if they exist, and creates the tables.

CREATE statements in `sql_queries.py` specify all columns for each of the five tables with the right data types and conditions.

- Good work with the datatypes selection for the fields. This is important when we need to do data wrangling. A very small example is when you sum up a field, you need to have an integer or float field.
- The NOT NULL and PRIMARY KEY fields are also defined appropriately for each table.

ETL

The script, `etl.py`, runs in the terminal without errors. The script connects to the Sparkify database, extracts and processes the `log_data` and `song_data`, and loads data into the five tables.

Since this is a subset of the much larger dataset, the solution dataset will only have 1 row match that will populate a `songid` and an `artistid` in the fact table. Those are the only 2 values that the query in the `sql_queries.py` will return.
The rest will be `none` values.

INSERT statements are correctly written for each tables and handles existing records where appropriate. `songs` and `artists` tables are used to retrieve the correct information for the `songplays` INSERT.

- The INSERT statements handle the ON CONFLICT scenarios as appropriate. The `level` of a user is being modified upon any UPDATE request, rest are just ignored as most of the others are static data, which are very unlikely to change.

Code Quality

The README file includes a summary of the project, how to run the Python scripts, and an explanation of the files in the repository. Comments are used effectively and each function has a docstring.

Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Naming for variables and functions follows the PEP8 style guidelines.

- The scripts follow PEP8 guidelines. Each task is divided into tasks and the variable/function names are intuitive.

DOWNLOAD PROJECT

1 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review