UDACITY

‹ Return to "Data Engineering Nanodegree" in the classroom

DISCUSS ON STUDENT HUB ›

Data Modeling with Postgres

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hello Udacity Student,
Great job done with this submission. All the requirements have been met successfully 💯. Congratulations on making it through this project👏. I want to encourage you to keep learning and do not be afraid of trying new things This is just a springboard to a successful career. I wish you success in the projects ahead as I look forward to seeing future submissions from you. Have a nice wonderful day.

Best regards,
Reviewer.🏆

> The directory for this project is located here: https://github.com/tommytracey/udacity_data_engineering/tree/master/p1_data_modeling_postgres

Well done!🙏

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.

Nice implementation!
The script, create_tables.py, perfectly runs in the terminal without errors. Also, 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 specifying the right data types and conditions with the primary keys indicated for each table. Nicely done!

Extra Learning

- Please check this resource to know more about SQL Data Warehouse supported data types.
- SQL data types divided into categories.
- Here a nice discussion about varchar vs int for storing a variable
- Here is a very nice explanation about SQL Constraints
- SQL Keys and Constraints
- Different types of constraints in SQL.

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.

Good work!
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.✓

✓

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.

Awesome!
This is good understanding demonstrated here. Good work with the update statement for the users table.💪
However, apart for the user table, the rest could do nothing on update.

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.

Impressive README file! It contains all the necessary details for a writeup and comments were effectively used to explain each functions. 👍
Docstrings were missing in the functions. Though there are no standard and rules for doing so, please note that it is an essential part that documenting your code is going to serve well enough for writing clean code and well-written programs.

Pro Tips

You may check some links below to know more about python code documentation:

- Documenting Python Code: A Complete Guide
- PEP 257 -- Docstring Conventions
- Python Docstrings
- Different types of writing Docstrings

✓

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

Good code practice in this submission. Moreover, variables and functions follow the PEP8 style guidelines.✓

Suggestion

For details, visit PEP 8 -- Style Guide for Python Code | Python.org.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review