‹ Return to "Data Engineering Nanodegree" in the classroom

DISCUSS ON STUDENT HUB ›

Data Warehouse

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Your work is amazing!

Keep doing this to go on with your great trajectory.
😄 😄 😄

Good luck with your next projects!

If you want to add me on the Linkedin (Rafael Buck) feel free.

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.**

Good job here, the create_tables.py runs without errors, drops any tables if they exist, and creates the tables as expected 😄

✓ **CREATE statements in sql_queries.py specify all columns for both the songs and logs staging tables with the right data types and conditions.**

Good, the queries for both the songs and logs staging tables are with the correct data types. Awesome!

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

ETL

✓ **The script, `etl.py`, runs in the terminal without errors. The script connects to the Sparkify redshift database, loads `log_data` and `song_data` into staging tables, and transforms them into the five tables.**

Good job here 😄

✓ **INSERT statements are correctly written for each table and handles duplicate records where appropriate. Both staging tables are used to insert data into the songplays table.**

Nice work on adding the ARN!

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.**

Great and informative docstrings!

Suggested: You can improve the README by including images of the schema design, query results, etc. Here a link about this https://help.github.com/en/github/creating-cloning-and-archiving-repositories/about-readmes

✓ **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 code is optimized with an easy-to-follow structure and follows the guidelines as expected.

Suggested: here's a reference, PEP-8, which guides you how to leave your Phyton code documented with the best practices in the market 😄

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review
☆☆☆☆☆