UDACITY

Return to "Data Engineering Nanodegree" in the classroom

Data Warehouse

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear student

Congratulations on completing the project! You should be very proud of your accomplishments in building an ETL pipeline for a star schema on Redshift! In the review, you'll find feedback and some tips to help you continue to improve on your project. Also, you can shutdown your redshift now. 😊👏

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.

The script successfully runs, dropping and creating all tables in the redshift database.

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

Great job with your staging table definitions!

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

Great job with your staging table definitions!

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.

The script successfully runs, copying data into your staging tables and inserting data into your final tables in the redshift database.

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.

Great job with your INSERT statements and using DISTINCT to address duplicate entries. To make this even better, consider how you'd want to handle duplicate entries with the same id but different values for other columns. For example, two user entries with the same information except for the level? This could happen when a user upgrades from a free to a paid level. Would you want to keep both entries or just the one with the most recently updated level?

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 job including information on the project, instructions, and files in your README file! You can preview your README by right clicking your file and clicking preview in your text editor. This will look pretty different than it does when you're editing your README file in a plain unformatted text file. Remember to use two spaces after a line to start a new line, or enter a blank line to include an empty space in between two lines. You can also use three backticks to format text as code. Use them to start and end a code block. In addition, consider adding some examples in your README to demonstrate the use of your database. You can add code blocks or images that show example queries and results that the analytics team would find useful.

Great job with your clear and concise docstrings and comments! 👏🏽

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

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review