Data Modeling with Postgres

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations on completing the **Data Modeling with Postgres** project. You have demonstrated a significant understanding of table design in Postgres. The script ran without any errors updating the final tables as specified. Keep up the good work.

Here are some good articles on the database schema:

- What is a Database Schema - What are your database diagram needs?
  https://www.lucidchart.com/pages/database-diagram/database-schema
- Instance and schema in DBMS
  https://beginnersbook.com/2015/04/instance-and-schema-in-dbms/
- Database Normalization (Explained in Simple English)
  https://www.essentialsql.com/get-ready-to-learn-sql-database-normalization-explained-in-simple-english/

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 ran without any errors creating tables, dropping them if it already exists.

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

- PRIMARY KEYs and NOT NULLs specified appropriately.

Note:

Need not specify NOT NULL along with PRIMARY KEYs.

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 with values for value containing ID for both `songid` and `artistid` in the fact table. Those are the only 2 values that the query in the `sql_queries.py` will return that are not-NONE. The rest of the rows will have NONE values for those two variables.

It's okay if there are some null values for song titles and artist names in the `songplays` table. There is only 1 actual row that will have a songid and an artistid.

- The ETL script ran without errors updating the final tables.

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

- Nice job updating the `level` column for `users` table for the existing records. Users might change from 'free' to 'paid' and vice versa.
- For the rest of the tables, artists, songs, time, ON CONFLICT DO NOTHING is fine.

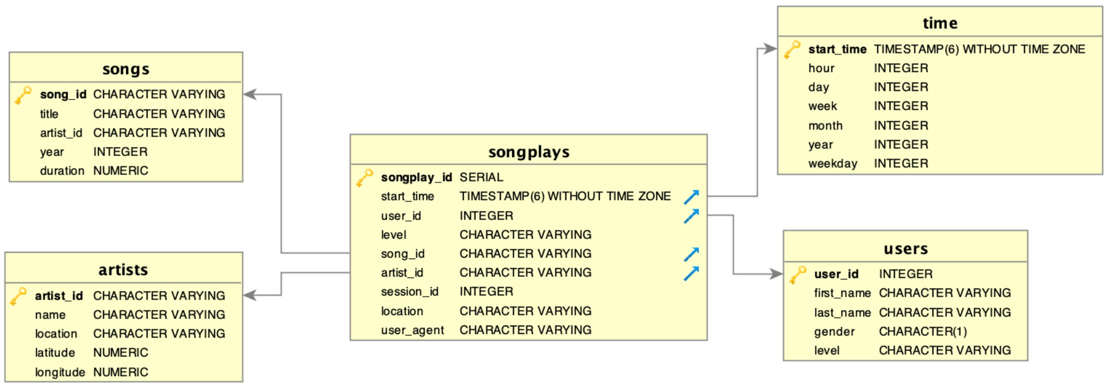
Code Quality

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.

✅ README

Good job with the well-structured and detailed README. A nice README is a great way to showcase your project to potential employers. Suggestions:

- Add a screenshot or an image (ER Diagram) showing how the fact and dimension tables are connected. Example:



songs
song_id CHARACTER VARYING
title CHARACTER VARYING
artist_id CHARACTER VARYING
year INTEGER
duration NUMERIC
artists
artist_id CHARACTER VARYING
name CHARACTER VARYING
location CHARACTER VARYING
latitude NUMERIC
longitude NUMERIC
songplays
songplay_id SERIAL
start_time TIMESTAMP(6) WITHOUT TIME ZONE
user_id INTEGER
level CHARACTER VARYING
song_id CHARACTER VARYING
artist_id CHARACTER VARYING
session_id INTEGER
location CHARACTER VARYING
user_agent CHARACTER VARYING
time
start_time TIMESTAMP(6) WITHOUT TIME ZONE
hour INTEGER
day INTEGER
week INTEGER
month INTEGER
year INTEGER
weekday INTEGER
users
user_id INTEGER
first_name CHARACTER VARYING
last_name CHARACTER VARYING
gender CHARACTER(1)
level CHARACTER VARYING

You can make use of online tools like https://www.lucidchart.com/

✅ Docstrings.

Nice job adding docstrings. The docstrings are essential in describing what a function does. It's not just going to help you understand and maintain your code. It will also make you a better job candidate.

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 mostly follows the PEP8 style guide
- The SQL statements are well-formatted.