UDACITY

Return to "Data Engineering Nanodegree" in the classroom

Data Modeling with Postgres

REVIEW

CODE REVIEW 3

HISTORY

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

Dear student,

This is a really good start. Your project is in good shape and working without any error. The submission shows that you have understood the concepts of dimensional modeling in Postgres pretty well. However, you need to make a few changes as per a few rubrics. I have tried to explain the required changes in detail and how to make those changes. Kindly go through them. I hope you will take this constructively and consider as an opportunity to learn and grow. 😄

To answer your question, you are worried about what if `user_id` contains a record of a mismatched datatype. There are multiple approaches to it

- Data Collection/Pre-processing: While gathering the data from raw sources, you need to analyze and clean the data properly, maybe by removing the

data.
- Exception Handling: You can use `try-except` block to handle any exception thrown during insertion.

I hope it will help you. But from this project perspective, you don't need to worry about this at least in case of `user_id` INT type with NOT NULL will work perfectly.

All the very best for your next submission and the rest of the 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.

- The `create_tables.py` drops the tables if exist and creates using the schema defined in the `sql_queries.py`

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

- This is a really good start. You have added appropriate datatypes to almost all the fields apart from `start_time`. The PRIMARY KEYs are also defined. But the NOT NULL keys of the fact table are not defined. Kindly re-evaluate which fields in the tables should be NOT NULLs.
- Additionally you may take a look at the official documentation of PostgreSQL on constraints.

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.

- The `etl.py` runs smoothly. The data are extracted from the logs and loaded into the database.

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.

- For the `user_table_insert` you have put `DO NOTHING` when there is a conflict. But would we like to ignore it? The `level` field is used for `free` and `paid` services for a user. Consider someone subscribes by paying a fee who happened to be a free user before. In that scenario, we would like to update the `level` of the user to `paid`. The current implementation will ignore the update, which is not desired. You can do that by `ON CONFLICT (user_id) DO UPDATE SET level=EXCLUDED.level`
- For more info take a look at Lesson 2.25.

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.

- While functionality is `important, it's also equally important that you present your project in a professional and polished manner. That's the objective of this rubric. The README file explains the project, the schema description and how to run the scripts.

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

- This rubric is to ensure that the scripts are written following the PEP8 guidance.
- Kindly make sure that the lines are not too long (ideally 72 characters) and docstrings are provided for each function. You can add what a function is doing, what are the arguments taken, what is going to be the output.

An example would be:

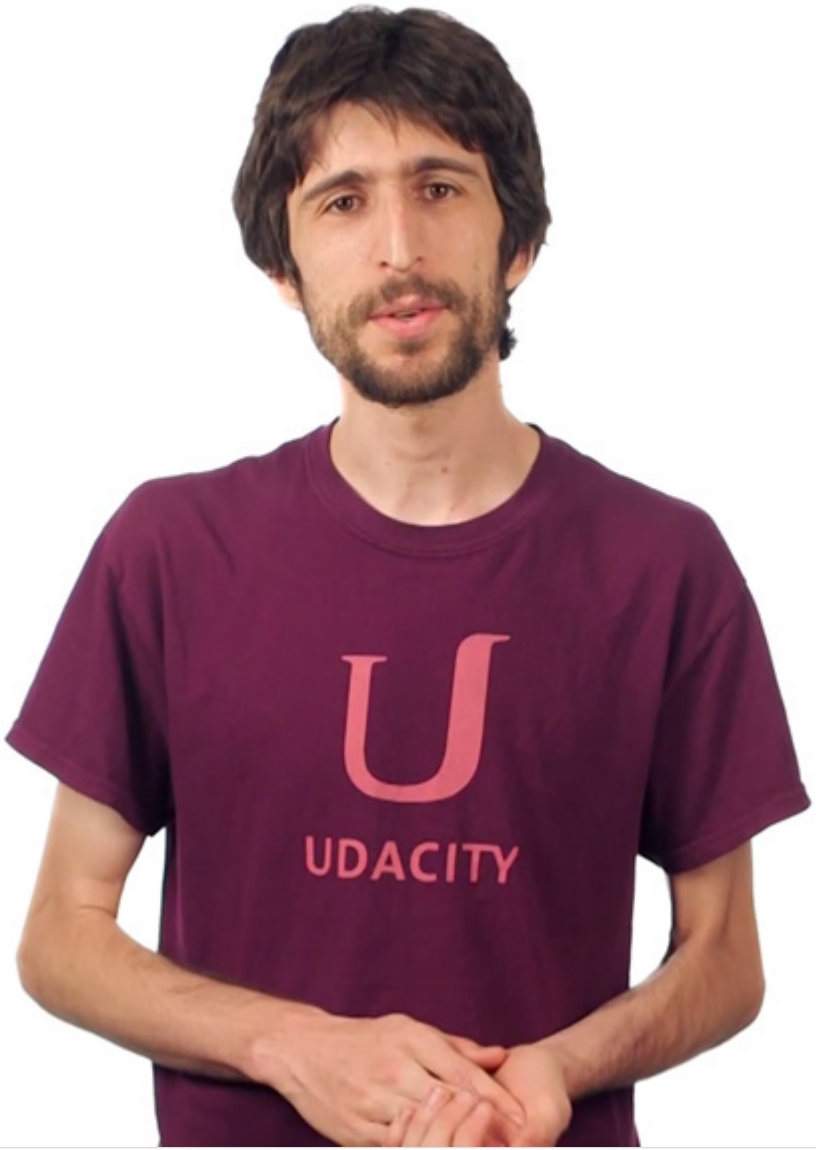
```
def process_log_file(cur, filepath):
"""
Description: This function can be used to read the file in the filepath (data/log_data)

to get the user and time info and used to populate the users and time dim tables.

Arguments:
    cur: the cursor object.
    filepath: log data file path.

Returns:
    None
"""
```

- Kindly take a look at the official PEP8 documentation here

RESUBMIT

DOWNLOAD PROJECT

3 CODE REVIEW COMMENTS ›


U
UDACITY

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review