UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Data Modeling with Postgres

REVIEW

CODE REVIEW 7

HISTORY

Requires Changes

3 specifications require changes

3 specifications require changes

Dear Student,

The good news is that the program is running without any error. Good job!
Just a few more steps and you will be good to go!
Here are the main issues you need to address. I gave you some useful notes on how to improve
your output in the project review.

- There are just a few issues around the NOT NULL constraint; update the primary key for the songplays table and check the data types. For more information, check the project review.
- It would help if you worked on the conflict for users table, and you don't need to insert songplay_id in the songplayplay_table_insert since it is a serial data type computer-generated—the insert statement.
- You need to update your Readme.md file; please check the project review.

Good Luck with your next submission!

Here are some **essential resources:**

- https://datatofish.com/convert-pandas-dataframe-to-list/
- https://www.postgresqltutorial.com/postgresql-upsert/
- https://www.geeksforgeeks.org/python-docstrings/

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.

Nicely done! Your create_tables.py drops the tables if they exist, and creates them properly!

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

Good first attempt! 👍

white_check_mark: You have created dimension and fact tables as required in the template.
✅ Specified the PRIMARY KEYS to user_table_create, song_table_create, time_table_create
And artist_table_create.
Required Changes:
❌ songplays_table_create primary key is not correct. songplay_id is computer generated. If you want to learn more about serial data type, please check this link.
The role of a Primary Key provides a unique identifier to each row in a table.

- The data type for the start_time is bigint or timestamp.
- Please check the types of duration, latitude, and longitude.
  You will see those are numbers, now try to find out if those are int or float or numeric. For More information please go through this link:
  [The difference between real and numeric]

please try to find out which fields in the fact table should be NOT NULL.
Hint
This is a data warehouse, not an OLTP database. Please remove all of the null constraints except the FOREIGN KEYs in this design. The FOREIGN KEYs in here are start_time, user_id, song_id, and artist_id. Having NOT NULL will cause you problems with the load MANY times.
The NOT NULL constraint enforces a column NOT to accept NULL values.
For further assistant in NOT NULL constraint, please check this link

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.

Awesome work with the ETL script. Your ETL script connects with Sparkify database, extracts both the log and song data and loads them into the five tables as required in the rubric.

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.

Good first attempt!
✅The conflict statement for the song, artist and time tables looks good.

- You don't need to insert songplay_id into the songplay table as it is computer generated.
- External Resources
  If you want to learn more about how to insert in postgreSQL, please go through this link
  - You need to update the conflict statement in the user's table.
- Think about what happens when there is a CONFLICT. For e.g., in user_table_insert, what if there is a CONFLICT on user_id? How do you want the table to handle that? Should it UPDATE the level? Again, check this link to think about that.
- External Resources
  If you want to learn more about how to insert in PostgreSQL, please go through this link

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.

This is a good start!

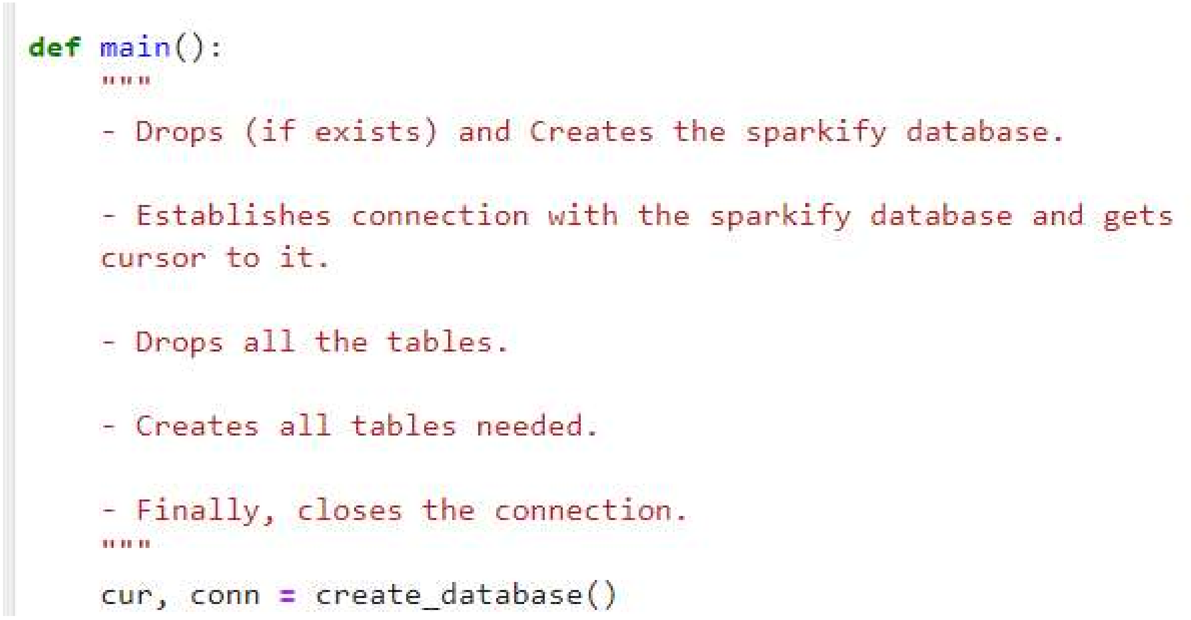
- You can elaborate on the ETL pipeline.
- It would be nice to see an example of how the duplication occurred (maybe just referencing the variable/column name) and which tables?
- I would encourage you to use images, like screenshots of your final tables, and adding more details about your project, such as the dataset and how you cleaned the data.
- You need to write a little bit about the Sparkify database.
  - Write about the fact table and dimension tables. From project instruction, take some ideas.

[illegible]

- Say a little about how you have solved it if you have any errors or issues.

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

`Nice work here! Your scripts follow the PEP8 style guidelines`, and the various functions are distinct and separated with proper function names. For example, the indentations within functions and CREATE statements are correct.
You have `added docstring` in the create_tables.py file. In programming and in Python, we follow the docstring and PEP8 convention for comments
For more information on pep8, please check this link
Here is the screenshot from your create_tables.py file:

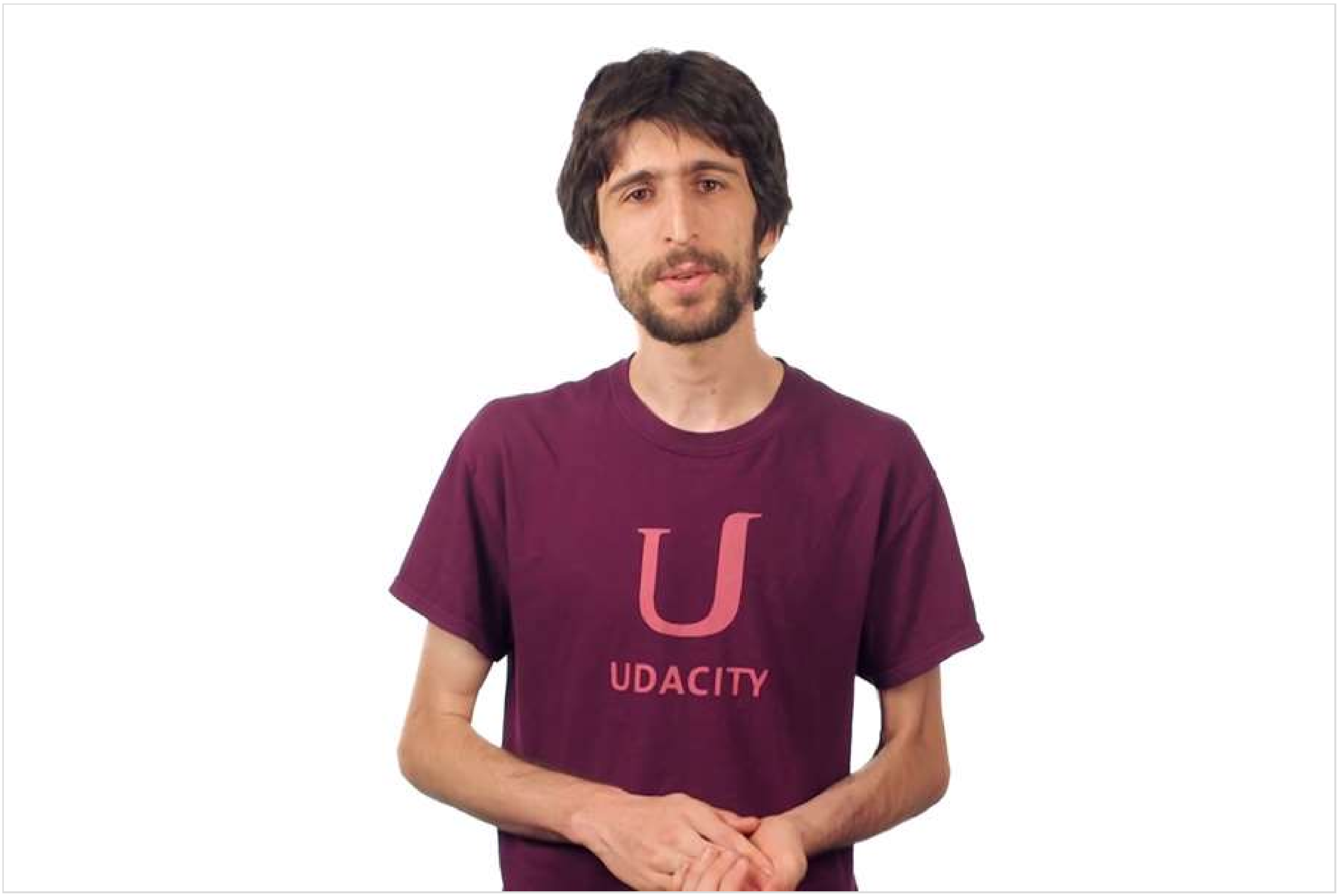


```
def main():
    """
    - Drops (if exists) and Creates the sparkify database.

    - Establishes connection with the sparkify database and gets
    cursor to it.

    - Drops all the tables.

    - Creates all tables needed.

    - Finally, closes the connection.
    """
    cur, conn = create_database()
```

RESUBMIT

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›



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

START