UDACITY

< Volver al aula

Data Modeling with Postgres

REVISIÓN

REVISIÓN DE CÓDIGO

HISTORIAL

Cumple las especificaciones

Dear Student,
Excellent job submitting a great project 👍🏽
And congratulations you did very well.

Kindly go through the suggestions carefully and please take them positively and constructively as an opportunity to learn and grow and also improve the overall quality of your work. Learning in data science never stops and as you will see, there is not always one perfect way to do things but intuition and experimentation are very often required depending on the situation.

For further learning in Postgres you can refer to the following links -

- Postgres Best Practices
  https://blog.digitalocean.com/some-postgres-best-practices/
- Python PostgreSQL Tutorial Using Psycopg2
  https://pynative.com/python-postgresql-tutorial/
- Postgres Quick Start and Best Practices
  http://www.jancarloviray.com/blog/postgres-quick-start-and-best-practices/
- Python PostgreSQL CRUD Operations
  https://pynative.com/python-postgresql-insert-update-delete-table-data-to-perform-crud-operations/

Also If this review helped you in some way, Please consider rating it positively.
I wish you all the best for your future endeavors. Keep learning and stay udacious!

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.

You can implement `NOT NULL` constraint on the foreign keys in the CREATE statements. However, please be careful with this implementation when you implement the `INSERT` functions (see rubric "ETL script properly processes transformations in Python") since the dataset contains some null values.

You should run the tests under the `Sanity Tests` section at the end of the `test.ipynb` notebook to check your work for obvious errors.

Excellent work !
I see you have created dimension and fact tables as required in the template. Additionally, you have specified the PRIMARY KEYS and NOT NULL conditions for the appropriate columns in the tables. Your table are specifying the correct data type for all the columns.

- Having NOT NULLs ensures that the data in the critical columns of the table is not missing or corrupt. You can find use of NOT NULL in various queries in PostGres here - https://www.techonthenet.com/postgresql/is_not_null.php
- For Primary Key: The role of a Primary Key is to provide a unique identifier to each row in a table. That is why it cannot be null. The Primary Key is mandatory in order to define a Foreign Key relationship between a child table and a parent table which is the basic premise of relational databases. Read more about primary keys here - https://www.techopedia.com/definition/5547/primary-key
- Here's another link that highlights how How To Create, Remove, & Manage Tables in PostgreSQL on a Cloud Server - https://www.digitalocean.com/community/tutorials/how-to-create-remove-manage-tables-in-postgresql-on-a-cloud-server
- To improve further, you can look into how you can add FOREIGN KEYs in your fact table :
  -- https://www.w3resource.com/PostgreSQL/foreign-key-constraint.php
  -- https://softwareengineering.stackexchange.com/questions/375704/why-should-i-use-foreign-keys-in-database

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

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.

You should run the tests under the `Sanity Tests` section at the end of the `test.ipynb` notebook to check your work for obvious errors.

Good job !
The script 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.

While we've used ETL pipeline in our projects there is something called ELT pipeline as well.
Read here to understand more details about ETL/ELT pipeling -
https://www.ironsidegroup.com/2015/03/01/etl-vs-elt-whats-the-big-difference/

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.

You should run the tests under the `Sanity Tests` section at the end of the `test.ipynb` notebook to check your work for obvious errors.

PERFECT

- The SELECT statement is in place. You have correctly made the JOIN to extract and load the data to the fact table.
- The INSERT statements handle the ON CONFLICT scenarios as appropriate. The level of a user is being modified upon any UPDATE request which will ensure when a user changes the subscription paid of free it correctly gets updated. Rest are just ignored as most of the others are static data, which are very unlikely to change.
  Optionally you can read about how to take your project to next level by using COPY command here which explains fast-load-data-python-postgresql.

Here's a mini on point tutorial on the ON CONFLICT clause for postgresql -
http://www.postgresqltutorial.com/postgresql-upsert/

Code Quality

- Create a `README` file with the following information:
  - a summary of the project
  - how to run the Python scripts
  - an explanation of the files in the repository
- `DOCSTRING` statements have been added in each function in etl.py file to describe what each function does.

https://gist.github.com/PurpleBooth/109311bb0361f32d87a2

Detailed Readme file is important as it helps the user to present his/her case to viewer, Consider, your potential employer visits your github repo while evaluating you for the potential job, the readme file present your thought process behind the project and how you went about doing the project.

Impressive work !

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

Quite nice work.
It's nice to have document strings in all your functions so that it is clear to anyone looking at the function as to what it does and what are the parameters or what the function returns.

For document strings you might also check out the following -
This post regarding Docstrings vs Comments : https://stackoverflow.com/questions/19074745/docstrings-vs-comments
This Best of the Best Practices" (BOBP) guide to developing in Python :
https://gist.github.com/sloria/7001839

DESCARGAR PROYECTO

VOLVER A RUTA