UDACITY

< Return to Classroom

Data Lake

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hi there, it's Cláudio! Thanks for sending all the required files for the review process and for the all queries executing without any error.

Congratulations on completing the project! You should be very proud of your accomplishments in building an ETL pipeline for a star schema on a data lake! In the review, you'll find feedback and some tips to help you continue to improve on your project.

I hope you had enjoyed doing this project and put in practice important concepts from data foundations. I will leave my contact below in case you have any doubt about this review as well to stay connected.

That's all! Enjoy data foundations and keep it up the great work so far.

Thank you.
Cláudio

Email: cgimenest@uol.com.br
Linkedin: https://www.linkedin.com/in/claudiogimenestoledo/

ETL

The script, etl.py, runs in the terminal without errors. The script reads song_data and load_data from S3, transforms them to create five different tables, and writes them to partitioned parquet files in table directories on S3.

The script successfully runs, reading song and log files from S3 and transforming them into five tables written as parquet files in directories on S3.

Tip:

- For more advanced techniques and further reference - you might find this link useful:
  https://stackoverflow.com/questions/45082832/how-to-read-partitioned-parquet-files-from-s3-using-pyarrow-in-python
  https://arrow.apache.org/docs/python/parquet.html

Each of the five tables are written to parquet files in a separate analytics directory on S3. Each table has its own folder within the directory. Songs table files are partitioned by year and then artist. Time table files are partitioned by year and month. Songplays table files are partitioned by year and month.

Great job organizing your tables into well partitioned parquet files in separate directories for each of the five tables!

Tip:

- To keep yourself up to date with the new S3 features - you can also include as in your favourite list:
  https://www.simplilearn.com/tutorials/aws-tutorial/aws-s3

Each table includes the right columns and data types. Duplicates are addressed where appropriate.

Correct columns and datatypes are used for each table. Great job dropping duplicates where appropriate! 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! Consider adding some examples in your README to demonstrate the use of your data lake. 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!

Tip:

- Read Me files are so important for scalability and document purposes: This is a great tutorial that covers the majority of its aspects: https://www.makeareadme.com/
- Udacity also offers a great free course in this: https://www.udacity.com/course/writing-readmes--ud777

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

Code follows an organized structure and clear variable and function names are used.

Tip:

- If you are keen to understand more about this important style for the dev community, refer to this useful link: https://www.python.org/dev/peps/pep-0008/

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START