‹ Return to "Data Engineering Nanodegree" in the classroom

DISCUSS ON STUDENT HUB ›

Data Lake

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Congratulations on completing the project! You should be very proud of your accomplishments in building an ETL pipeline in Apache Spark

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.

✓ **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!

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

Great job specifying the columns and datatypes for each table! Distinct keyword does help in eliminating duplicates but, It is better to use drop duplicates api.. On top of that drop duplicate provide you more control.. 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?

Extra Materials

- Deduplicating and Collapsing Records in Spark DataFrames
- Drop duplicates by some condition
- Stackoverflow: Spark SQL DataFrame - distinct() vs dropDuplicates()

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!

✓ **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 l and clear variable and function names are used.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

☆☆☆☆☆