Data Lake

REVIEW | CODE REVIEW 2 | HISTORY

Meets Specifications

Congratulations on completing the Data Lake Project. You have demonstrated good understanding of Spark and data lakes to build an ETL pipeline for a data lake hosted on S3. In the review, you'll find some feedback to help you continue to improve on your project.

A few articles of interest:
TOP FIVE DIFFERENCES BETWEEN DATA LAKES AND DATA WAREHOUSES
https://www.blue-granite.com/blog/bid/402596/top-five-differences-between-data-lakes-and-data-warehouses

What Is A Data Lake? A Super-Simple Explanation For Anyone
https://www.forbes.com/sites/bernardmarr/2018/08/27/what-is-a-data-lake-a-super-simple-explanation-for-anyone/#4f61b70d76e0

When Should We Load Relational Data to a Data Lake?
https://www.sqlchick.com/entries/2018/11/13/when-should-we-load-relational-data-to-a-data-lake

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 ran without any other errors. Well done!

✓ **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.**

All the tables, except the `songplays` are correctly partitioned as per specification and written to parquet files.

- The `songplays` table files should be partitioned by `year` and `month`.
  Join with the time table, include year and month column and then partition the songplays table by year and month.

Here's an example with spark sql:

```
    songplays_table = spark.sql("SELECT e.start_time,\
                e.userId, e.level, s.song_id,\
                s.artist_id, e.sessionId, e.userAgent, \
                t.year, t.month \
            FROM events_table e \
            JOIN songs_table s ON e.song = s.title AND e.length = s.duration \
            JOIN artists_table a ON e.artist = a.artist_name AND a.artist_id = s.artist_id \
            JOIN time_table t ON e.start_time = t.start_time ")

    # write songplays table to parquet files partitioned by year and month
    songplays_table.write.partitionBy(['year', 'month']).parquet(output_data + "songplays_tabl
e.parquet")
```

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

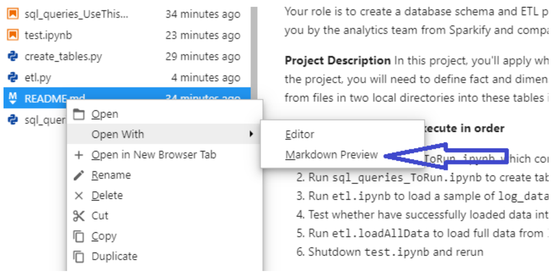
- Nice job handling duplicate records.

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.**

Please add more content to README. A nice README is great way to showcase your project to potential employers. Following sections provide better readability:

- Introduction - purpose of project, What is Sparkify, how this project is going to help it.
- Database schema design and ETL process
- Files in repository
- How to run the python scripts
- Add a screenshot or an image showing how the fact and dimension tables are connected.
- Properly use the markdown language in your README, for headings, lists, subsections. Refer:
  https://guides.github.com/features/mastering-markdown/
- You can use an online markdown editor like https://dillinger.io/
- Refer good READMEs on web:
  https://github.com/matiassingers/awesome-readme
  https://bulldogjob.com/news/449-how-to-write-a-good-readme-for-your-github-project
  https://medium.com/@meakaakka/a-beginners-guide-to-writing-a-kickass-readme-7ac01da88ab3
- You can preview your README in your Udacity workplace as below:



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

The code mostly follows the PEP8 style guide. Nice job with docstrings. The docstrings are important in describing what a function does. It's not just going to help you understand and maintain your code. It will also make you a better job candidate.

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ›

RETURN TO PATH