

[◀ Return to Classroom](#)[DISCUSS ON STUDENT HUB](#)

Data Lake

REVIEW

CODE REVIEW 6

HISTORY


Meets Specifications

Dear Learner,

Congratulations 🎉 on passing the project! I am very pleased to see how far you have come with this project. 100 The effort that you have put in is indeed commendable 🙌 The implementation of the etl.py file was very precise with the necessary requirements. 🌟 ⭐ I am very pleased to have reviewed your project.

USEFUL RESOURCES

- [ETL scripts in Python](#)
- [Automating ETL with AWS](#)
- [Programming ETL scripts in AWS](#)
- [DOCSTRINGS](#)
- [Python Docstring](#)

You may use this resources in the future. Hope you have learnt a lot from this project. Stay Udacious 

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.

Excellent work and proper execution! The etl.py script runs successfully. It correctly reads song data, loads them from S3 partitions them and writes them correctly in the required parquet files. 👍

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.

You have done a brilliant job in correctly setting up the tables to be written in the parquet files. You made great use of the `PartitionBy()` argument and successfully partitioned the song table, time table and songplays table files as required.

Learning Materials

To know more about the difference between `PartitionBy()` and `repartition()` and its importance:

- Please check out this [link](#).
- Here is a [discussion](#) about Writing data to Parquet with partitions.
- [stackoverflow](#) on how to save a partitioned parquet file in Spark 2.1.

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

Fantastic! Correct columns with right data types are created for the tables and duplicates are dropped when required.

Suggestions:

You can use `DISTINCT` to address the duplicates as well.

Difference Between DISTINCT and DropDuplicate:

The key difference is that `DISTINCT` removes any row that has the same value in all columns while the `dropDuplicates()` function eliminates those that have the same value in selected columns.

Extra Materials

Here are some extra materials you can look into.

- [Drop duplicates by some condition](#)
- [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.

You have written an Impressive README file! It contains all the necessary information , instructions and docstrings were effectively used to explain each function. Please note that it is an essential part of documenting your code because it gives a polished and more professional look. Kudos to you! ★

Extra Materials

You may check some links below to know more about python code documentation:

- [Documenting Python Code: A Complete Guide(<https://realpython.com/documenting-python-code/>)
- [PEP 257 -- Docstring Conventions](#)

Python Docstrings

- [Different types of writing Docstrings](#)

You might be interested to know more about READMEs:

[About READMEs](#)

[Make a README](#)

[Mastering Markdown](#)

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

Good job! The code is clear, clean, and well-formatted. Variable names are indicative and single-line comments are used in a good manner. Functions have indicative names and are focused on specific objectives.

Suggestions

You may find this link helpful to [Check your code](#) for PEP8 requirements.

Here are some additional resource to know more about PEP8 style guidelines:

[PEP 8: Style Guide for Python Code](#)

[How to Write Beautiful Python Code With PEP 8](#)

 [DOWNLOAD PROJECT](#)

6

[CODE REVIEW COMMENTS](#)



[RETURN TO PATH](#)

Rate this review

START