

[Return to Classroom](#)

Data Warehouse

REVIEW

CODE REVIEW **1**

HISTORY

Meets Specifications

Congratulations!

Two things I liked in the project:

- How the code is structured and aligned as per PEP8 style, this is very important thing that is missed by lot of developers.
- Comments, they are in good amount to provide the user with context.

The project submission is of top quality, multiline docstring are effectively used, comments are provided. Code quality is also very good.

All the best and good read on [data warehouse](#)

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.

Good start.

- The script `create_tables.py` runs fine in the terminal without any error.
- At the end of the execution, databases and two staging tables along with 5 others tables were created on the redshift cluster.

You can also create a cluster using the code, you can revisit the IAC lesson

Further Readings

- [Drop Tables](#)
- [Create tables](#)



CREATE statements in `sql_queries.py` specify all columns for both the songs and logs staging tables with the right data types and conditions.

Well done, the queries for both the songs and logs staging tables are with the correct data types. Time stamp column is rightly set to timestamp and the numeric columns are correctly set using double/int.

I liked the use of the distkey and sortkey, it speeds up the performance of the system. They are not mandatory requirement for this project but they are used very frequently in real world scenarios.

[Learn more about them here](#)



CREATE statements in `sql_queries.py` specify all columns for each of the five tables with the right data types and conditions.

Good job

- The tables are created right with correct [data types](#)
- Primary keys are also used, [Redshift Primary key constraint is informational only; they are not enforced by Amazon Redshift](#).

ETL



The script, `etl.py`, runs in the terminal without errors. The script connects to the Sparkify redshift database, loads `log_data` and `song_data` into staging tables, and transforms them into the five tables.

ETL.py ran in the terminal without any issues or error.

At the end of the execution:

- The data was copied onto the staging tables
- The fact table songplays and dimension tables users, artists, time and songs, were correctly loaded with unique records.

Suggestion; You can make use of print statement to show the progress of the ety.py execution to the users

Great job



INSERT statements are correctly written for each table and handles duplicate records where appropriate. Both staging tables are used to insert data into the songplays table.

Excellent use of distinct keyword to filter out the duplicates from dimension tables, this ensures only unique values are taken forward.

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.

Readme file is detailed enough to meet the expectation .

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.

Good work

Multiline docstring is also made available for each function in the etl.py and create_tables.py file. This [article](#) explains why docstring are important



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 is indented nicely and organized into different functions,
- Optimum level of comments in the code that helps describe the flow of the code.
- I liked the way you have structured the sql_queries.py file, they are correctly aligned, easy on eyes.

You can learn more about

- [PEP](#)
- [DRY](#)

 [DOWNLOAD PROJECT](#)

1

[CODE REVIEW COMMENTS](#)



