UDACITY

Return to Classroom

Data Warehouse

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations on completing the **Data Warehousing** project. Overall you have done a really good job with this project. There were no errors with the scripts. The SQL statements are well-formatted making it easy to understand. See below for some suggestions on revisions that can be used to improve it further.

Here are some resources to learn further:

- Top 8 Best Practices for High-Performance ETL Processing Using Amazon Redshift
  https://aws.amazon.com/blogs/big-data/top-8-best-practices-for-high-performance-etl-processing-using-amazon-redshift/
- How I built a data warehouse using Amazon Redshift and AWS services in record time
  https://aws.amazon.com/blogs/big-data/how-i-built-a-data-warehouse-using-amazon-redshift-and-aws-services-in-record-time/
- Redshift ETL: 3 Ways to load data into AWS Redshift
  https://panoply.io/data-warehouse-guide/redshift-etl/
- 2X Your Redshift Speed With Sortkeys and Distkeys
  https://www.sisense.com/blog/double-your-redshift-performance-with-the-right-sortkeys-and-distkeys/

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 job. The script ran without any errors creating tables, dropping them if it already exists.

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

- Staging tables are created as specified.
- Appropriate data types and columns used
- No constraints applied.

The staging tables should be an exact copy of data from the json or csv files. It is better not to apply any type of constraints or filtering at this stage.

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

- The final tables are created with appropriate columns and data types.
- PRIMARY KEYs are specified for the final tables.
- Please specify NOT NULLs where appropriate.

Please note that Redshift does not enforce unique, primary-key, and foreign-key constraints. Even though they are informational only, the query optimizer uses those constraints to generate more efficient query plans.
Ref:
https://docs.aws.amazon.com/redshift/latest/dg/c_best-practices-defining-constraints.html
http://www.sqlhaven.com/amazon-redshift-what-you-need-to-think-before-defining-primary-key/

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.

- The ETL script ran without errors, updating the final tables.
- Nice job using the 's3://udacity-dend/log_json_path.json' as JSON format while copying **events data**.

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.

- Nice job using SELECT DISTINCT to filter duplicate entries.
- Rewrite the `songplays_insert` query so that the records are filtered based **song title, artist name** and **song duration**.

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

Good job with the well-structured README. A nice README is a great way to showcase your project to potential employers. Suggestions:

- Include an ER Diagram to show how the different tables are connected.
  You can make use of online tools like https://www.lucidchart.com/
- Refer good READMEs:
  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

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.
[Required]

- Please add docstrings to each function in `etl.py` and `create_tables.py`
  Example:

```
def load_staging_tables(cur, conn):
  """
  Just write briefly what the function does, not how it does.
  """
```

  See more details and examples from here: https://www.pythonforbeginners.com/basics/python-docstrings

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
- The SQL statements are well-formatted.

Here's an excellent guide on writing beautiful Python code with PEP8: https://realpython.com/python-pep8/
Well-formatted SQL statements are important as well. Here are some references:

https://www.sqlstyle.guide/
https://gist.github.com/fredbenenson/7bb92718e19138c20591

DOWNLOAD PROJECT

RETURN TO PATH