

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

Data Warehouse

REVIEW


CODE REVIEW

HISTORY

Meets Specifications

Dear Enthusiastic Learner

Congratulations on meeting all the project specifications. 🎉👏

The work done demonstrates a good understanding of the concepts covered in the project. Continue with this hard work and good luck moving forward. I hope you get all of your answers in this review. If you have any further question then please reach out for mentors in knowledge based questions. 

Extra Materials

You might want to check out the following for more on the subject matter.

- [Top 8 Best Practices for High-Performance ETL Processing Using Amazon Redshift](#)
- [How a data warehouse was built using Amazon Redshift](#)
- [3 Ways to do Redshift ETL](#)
- [Data Warehousing on AWS](#)
- [2X Your Redshift Speed With 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.

Excellent implementation! 🎉 The script runs and connects to the Sparkify database successfully. DROP and CREATE tables have been implemented correctly. Nice use of IF EXISTS parameter in your drop table command, this clause is also useful when scripting, so the script doesn't fail if DROP TABLE runs against a nonexistent table. 🙌

Suggestions

- Here is a nice [discussion](#) about Amazon Redshift : drop table if exists.
- A good [documentation](#) for Amazon Redshift DROP TABLE.

Learning Materials

- Here is a nice [discussion](#) about Amazon Redshift : drop table if exists.
- A good [documentation](#) for Amazon Redshift DROP TABLE.

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

Nice work! Appropriate data types have been used on your staging tables. The staging tables need to be a copy of the source data and we do not want to apply any filtering or constraints at this stage. Good work avoiding any constraints in the staging tables. 🙌

Suggestions

Here is a very nice explanation about [SQL Constraints](#)
[SQL Keys and Constraints](#)
Different [types of constraints](#) in SQL.

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

Code for `DROP` statements in `sql_queries.py` are correctly implemented with the `IF EXISTS` clause.

Great job with your final table definitions. Every table has a primary key and all foreign keys have the NOT NULL constraint.

Learning Notes

- By default, the PRIMARY KEY constraint has the unique and not null constraint built into it, no need to specify it.
- You can also define a foreign key constraint to specify that the values in a column (or a group of columns) must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables using [REFERENCES](#).
- [SQL Keys and Constraints](#)

ETL

`INSERT` statements are correctly written i.e. wrong values/incorrect number of values are not inputted in any tables.

`INSERT` statements handle duplicate records where appropriate.

Both staging tables are used to insert data into the songplays table. They are correctly filtered (when required) before inserting data.

Great job with your `INSERT` statements and using `DISTINCT` to address duplicate entries.

Suggestions

To make this even better, 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?

Following resources will give you a clear understanding

- [Different types of JOINS](#)
- [When to use SQL JOINS](#)
- [SQL SELECT DISTINCT Statement](#)

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 `dwh.cfg` file is properly configured for IAM role and redshift database.

The `COPY` statements for `staging_events_copy` and `staging_songs_copy` have the correct S3 bucket path, IAM ROLE, and mapping.

Splendid! The script successfully runs, copying data into your staging tables and inserting data into your final tables in the redshift database. ✨

The script successfully runs, copying data into your staging tables and inserting data into your final tables in the Redshift database.

Learning Notes

[Top 4 ETL frameworks](#)

[Using Python script for data ETL](#)

Code Quality

Scripts have an intuitive, easy-to-follow structure with code separated into logical functions. Comments are used effectively and each function has a docstring. Naming for variables and functions follows the PEP8 style guidelines.

Excellent!! ✨

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 you submitted follows is easy to follow and is in accordance with PEP8 style guidelines

.### 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](#)
- [PEP-8 Tutorial: Code Standards in Python](#)

The README file includes a summary of the project including the purpose of the database and its analytical goals. It talks about how to run the Python scripts and an explanation of the files in the repository.

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 might be interested to know more about READMEs:

- [About READMEs](#)
- [How do you put Images on the README.md file](#)
- [Make a README](#)

Please go through these links and you can learn how you can do this step faster and easily!! But the way you have implemented it is also acceptable for this project!

Additional Links

- [Tutorial: Loading data from Amazon S3](#)
- [Using the COPY command to load from Amazon S3](#)

Only for Project Reviewers (No student work required)

This rubric will be ungraded. The reviewer will be providing a code review for the relevant files.

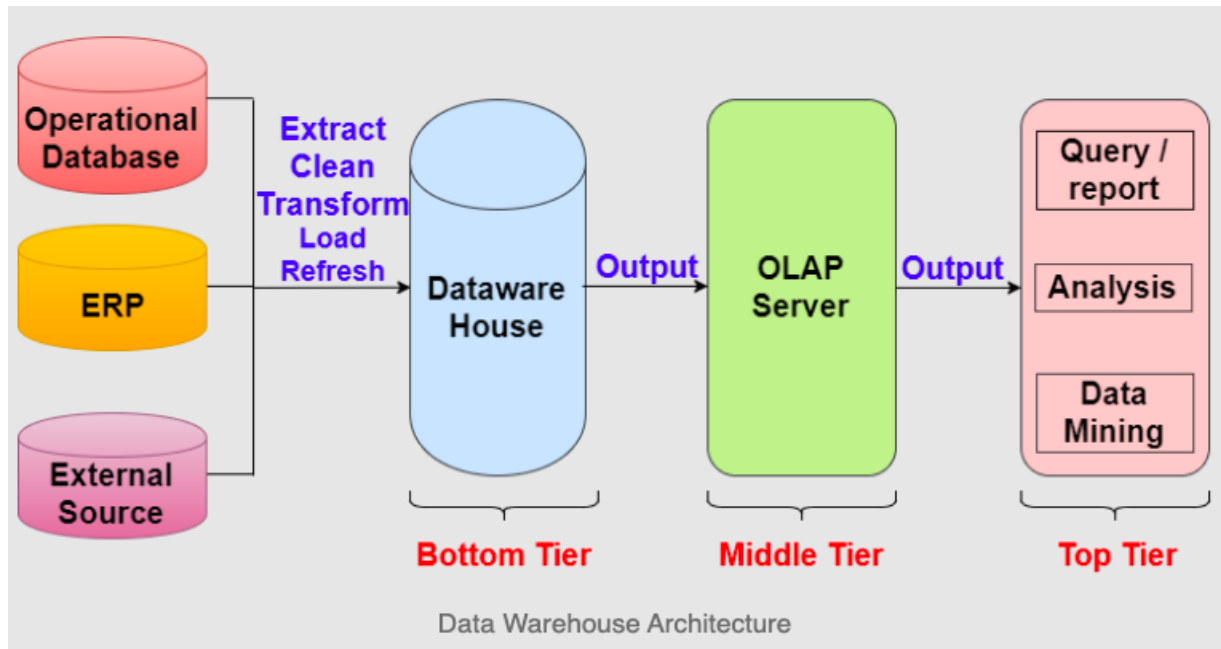
This rubric will be ungraded. The reviewer will brief the students about the concepts learned in this

section of the Nanodegree program.

What are data warehouses?

A system that transforms data (that was entered into a database during operational processes) into a form that supports efficient analytical processes.

The simplest [image](#) that can explain this is shown below:



A data warehouse is simply where data is stored. As the name suggests, warehouse for data! A nice example for this can be checked using this [link](#).

How do you create a redshift cluster without using the GUI - by using IAC?

- Infrastructure as Code(IAC) can help avoid cloud deployment inconsistencies, increase developer productivity, and lower costs. Check out the following posts about IaC:
 - [6 best practices to get the most out of IaC](#)
 - [15 Infrastructure as Code tools you can use to automate your deployments](#)

– [What is AWS CloudFormation and how can it help your IaC efforts?](<http://www.thorntech.com/2018/05/whatisawscloudformation/>)

– [How AWS CloudFormation Works (and How to Create a Virtual Private Cloud with it)](<https://www.thorntech.com/2018/06/createvpcwithcloudformation/>)

- How do you perform ETL from S3 buckets to Redshift Database?
 - When using a load design with staging tables, the ETL flow looks something more like this:
 - 1.) Delete existing data in the staging table(s)
 - 2.) Extract the data from the source
 - 3.) Load this source data into the staging table(s)
 - 4.) Perform relational updates (SQL) to cleanse or apply business rules to the data, repeating this transformation stage as necessary
 - 5.) Load the transformed data from the staging table(s) into the final destination table(s)
 - This load design pattern has more steps than the traditional ETL process, but it also brings additional flexibility as well. By loading the data first into staging tables, you'll be able to use the database engine for things that it already does well. For example, joining two sets of data together for validation or lookup purposes can be done in most every ETL tool, but this is the type of task that the database engine does exceptionally well.

type or task that the database engine does exceptionally well.

- You may check this [blog](#) about Why Data Warehouses Should Stage Source Data.

How do you perform ETL from S3 buckets to Redshift Database?

This [Tutorial: Loading data from Amazon S3](#) will provide detailed steps on how to do that.

This rubric is ungraded. If the learner has asked a question pertaining to the implementation of the project, the reviewer will provide an answer along with links to any helpful resources.

1. How to create an upsert behavior with redshift ?

Instead of using UPSERT you can use many other commands from the staging table.

Follow this [link](#)

2. For the table that we used to use ON CONFLICT DO NOTHING in PostgreSQL in assignment 1, how to get the same behavior with the redshift ?

Redshift is mostly compatible with PostgreSQL, so you can use any of the PostgreSQL clients for Python to query Redshift databases from Python. Please check the below references :

[Accessing AWS Redshift with Python](#)

[Access your data in Amazon Redshift and PostgreSQL with Python](#)

[Connecting Redshift using Python](#)

You don't need to write code to create the redshift cluster in etl.py or create_tables.py scripts as far as the project is concerned. You can create the cluster and add the required credentials/parameters in the config file.

3. What is the purpose of staging table when and why I will need it ?

Staging tables will be treated as intermediate tables, we will use them for some processing, like join tables and convert the timestamp, before inserting them into the final source tables. That is why we need to use Redshift to do the processing.

 [DOWNLOAD PROJECT](#)