Deep Dive - Data Warehouse

IBM

Agenda

- Introduction & Motivation
- About This Project
- Project Walkthrough
- Q & A
- Feedback

Introduction

Introduction

Jay Teguh Wijaya is a data engineer with specialization in financial modeling for corporate valuations.

About the AWS Credit

The $25 credit is allocated for the entire Nanodegree. Each course requires only about $3 to $5 to complete, but you need to use it judiciously. For example, rather than loading the whole dataset, you may try to load a part of it for testing your work.

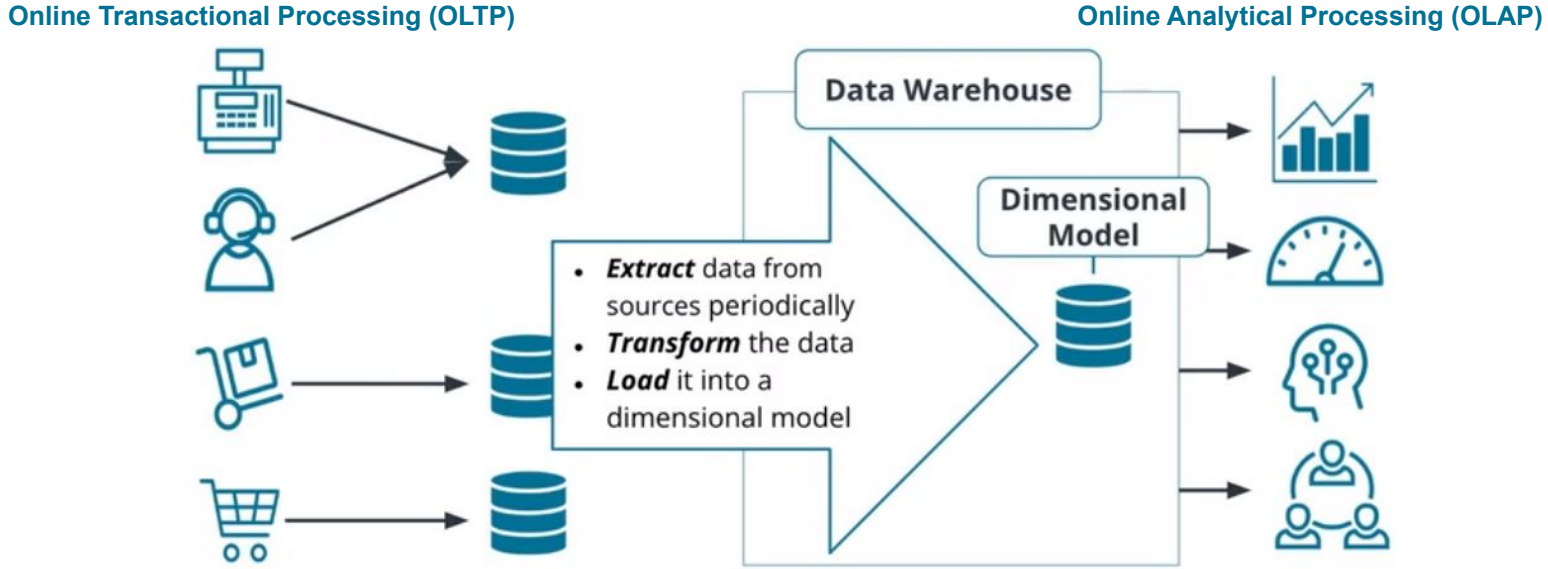
What is a Data Warehouse?

A data warehouse is a large store of data collected from a wide range of sources within a company and used to guide management decisions.

The concept of the data warehouse is different from a traditional database because it provides a large, centralized place to store data that is ready for analysis, rather than for transaction processing.

Trivia: The concept of data warehousing dates back to the late 1980s when IBM researchers Barry Devlin and Paul Murphy developed the "business data warehouse".

What is a Data Warehouse? (2)

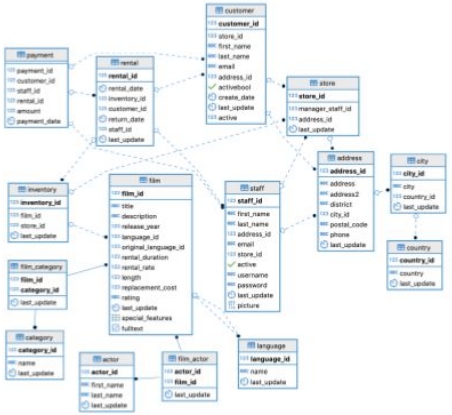


Classroom:
Introduction to Data Warehouses > [Data Warehouse Architecture]

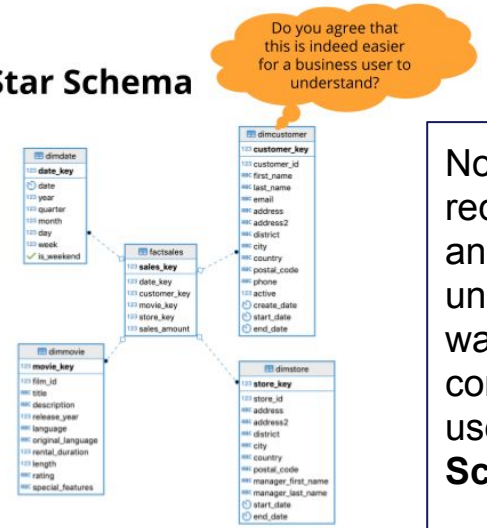
Dimensional Model

Note: 3rd Normal Form (or Normalized) schema maintains consistency across all tables, but most analytical queries require expensive JOINs.

3NF Schema



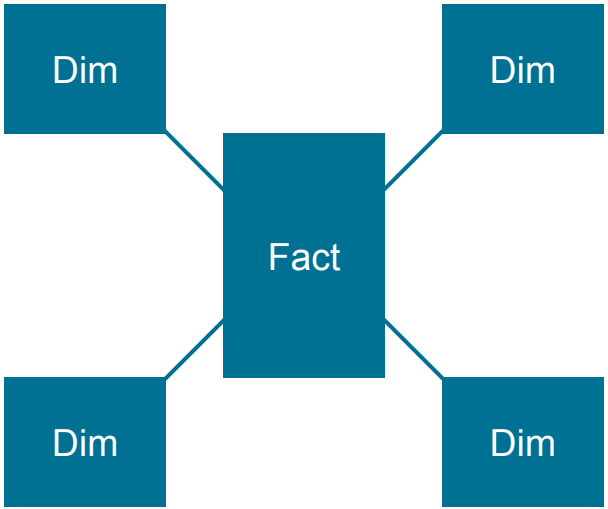
Star Schema



Note: Star Schema requires fewer joins and easier to understand, but if you want to maintain some consistency, you may use a **Snowflake Schema**.

Classroom:
Introduction to Data Warehouses > [ETL and Dimensional Modeling]

Star Schema



Fact tables:

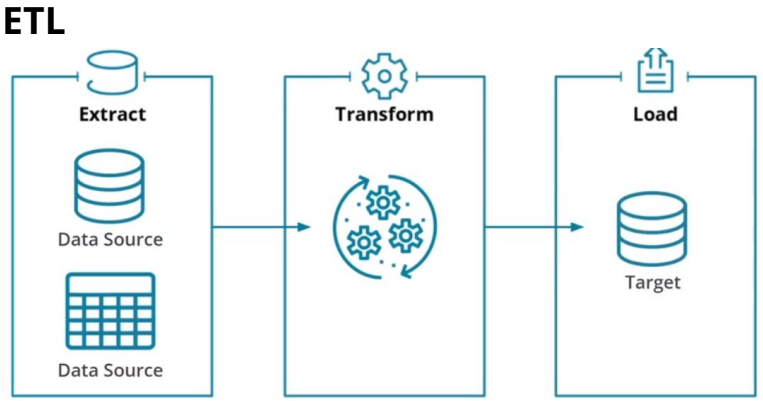
- Capture the data that's being measured.
- Have foreign key columns that point to dimension tables.
- *Often* store quantitative data.
- *Usually* have a key to the time dimension e.g. date.
- Ex: Orders, Transactions, Trips, Rentals, etc.

Dimension tables:

- Provide the context of business events, dates, times, etc.
- *Usually* Contain descriptive, textual data.
- Ex: Customers, Employees, Stores, Vehicles, etc.

Classroom:
Introduction to Data Warehouses > **ETL and Dimensional Modeling**

ETL and ELT



In ELT, we just switch the Transform and Load processes.

ETL or ELT?

Let's say we have this scenario:

- CSV files are located in an S3 bucket
- A COPY query is done to pull data from S3 to Amazon Redshift into staging tables
- INSERT queries are run to get the appropriate data from the staging tables to the final data model.

This is an ELT process. The ETL equivalent would be if the data is transformed before it's loaded into Redshift.

ELT is preferable in most modern architectures, but ETL may still be preferred in cases where it's important to transform the data before it's loaded into the data warehouse (for instance, to ensure data quality or to comply with data privacy regulations).

About This Project

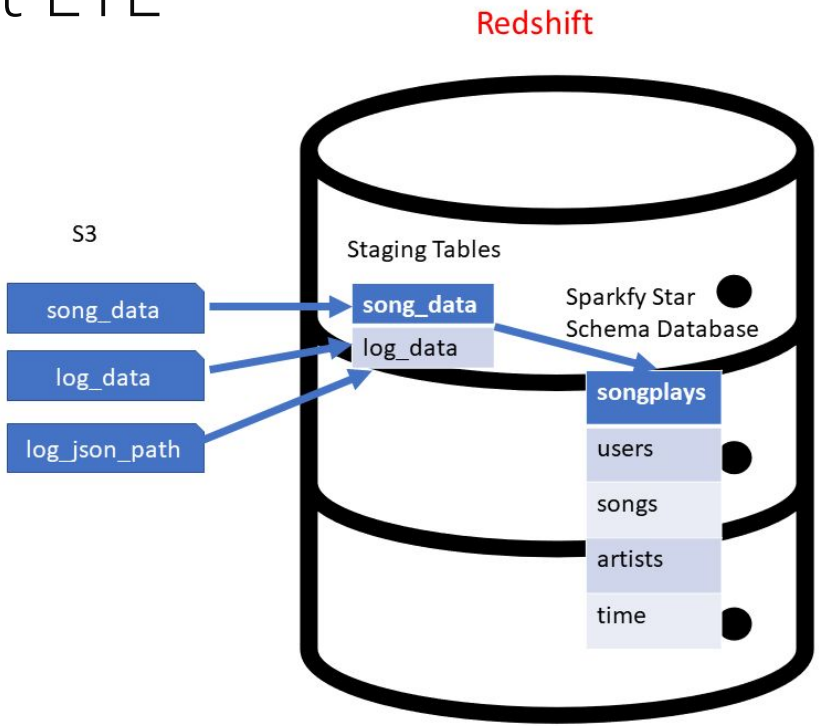
Project Description

A music streaming startup, Sparkify, has grown their user base and song database and want to move their processes and data onto the cloud. Their data resides in S3, in a directory of JSON logs on user activity on the app, as well as a directory with JSON metadata on the songs in their app.

As their data engineer, you are tasked with building an ETL pipeline that extracts their data from S3, stages them in Redshift, and transforms data into a set of dimensional tables for their analytics team to continue finding insights into what songs their users are listening to.

Sparkify S3 to Redshift ETL

Note: Sometimes ELT is called ETL since the latter sounds more familiar. Additionally, the boundaries between ETL and ELT have become somewhat blurred, with many data integration tools offering flexible options that can support both ETL and ELT workflows.



⚠️ AWS Costs

Amazon Redshift accumulates hourly costs after you have set it up, even when it is not running any query. You will use dc2.large node type with 2 nodes. Each node costs $0.25 per hour, so that will be $0.5/hour in total.

To save costs:

- Pause the cluster when you no longer need it. Paused clusters do not incur any cost.
- Use Udacity's S3 paths instead of uploading the datafiles to your S3 bucket.
- The `song_data` dataset is huge! Use a section of that dataset instead.

Your data paths should then be:

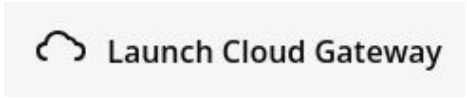
- `s3://udacity-dend/song_data/A/A/A`
- `s3://udacity-dend/log_data`
- `s3://udacity-dend/log_json_path.json`

Project Walkthrough

How to Launch The Cloud Gateway

Click on the **Launch Cloud Gateway** button on the bottom-left part of the Udacity classroom page, then **Open Cloud Console**.

Note: If you get permission error while working in AWS, refresh your Udacity classroom page, then re-access the console.

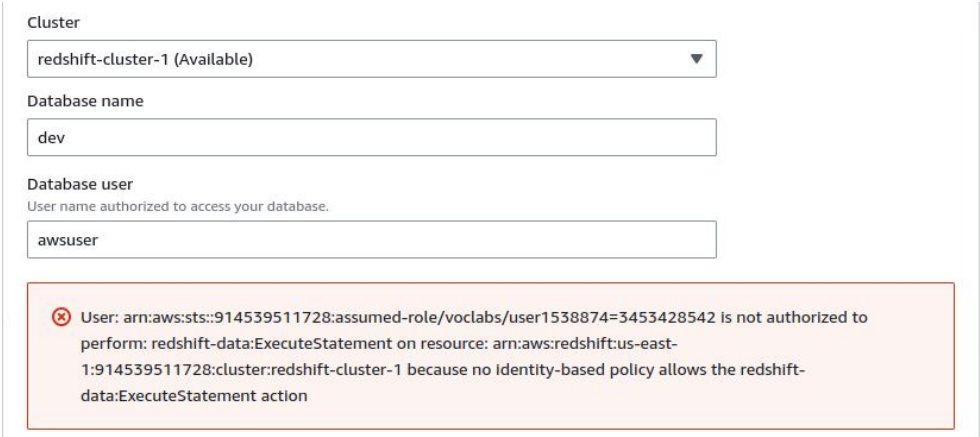


Step 1. Create a Redshift Cluster

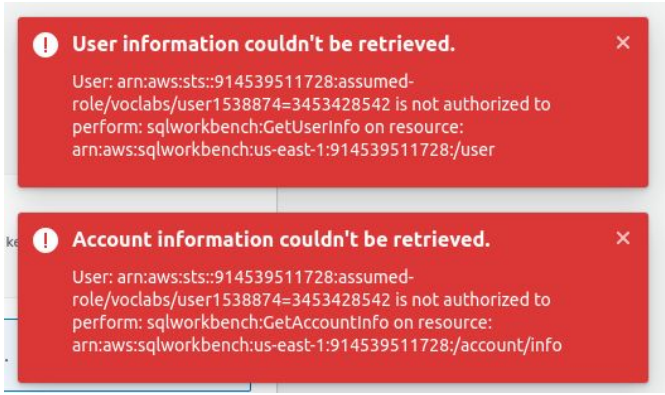
This step is pretty straightforward. Follow all the steps starting from the lesson **AWS Data Warehouse Technologies** > **Redshift Exercise: Create an IAM Role** and you should have a Redshift cluster that you can connect to from an external Python script (from the Udacity Workspace, for instance).

⚠️ Step 2. Create a new IAM User

On your Redshift console, Open up Query editor and then click on the **Connect to database** button, or open up Query editor v2. You'll get the following errors, respectively:



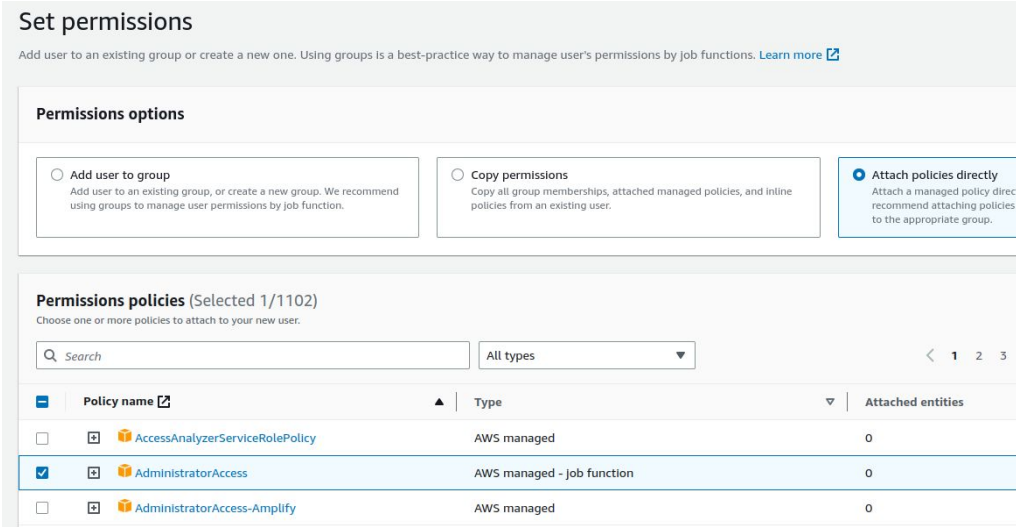
or



⚠️ Step 2. Create a new IAM User (2)

What causes the error was that the federated user account given by Udacity does not have access to programmatically connect to the Redshift cluster or query the tables from the Query Editor.

Hence, create a new user with console access and set its permission to AdministratorAccess (not best practice, but makes things simpler).

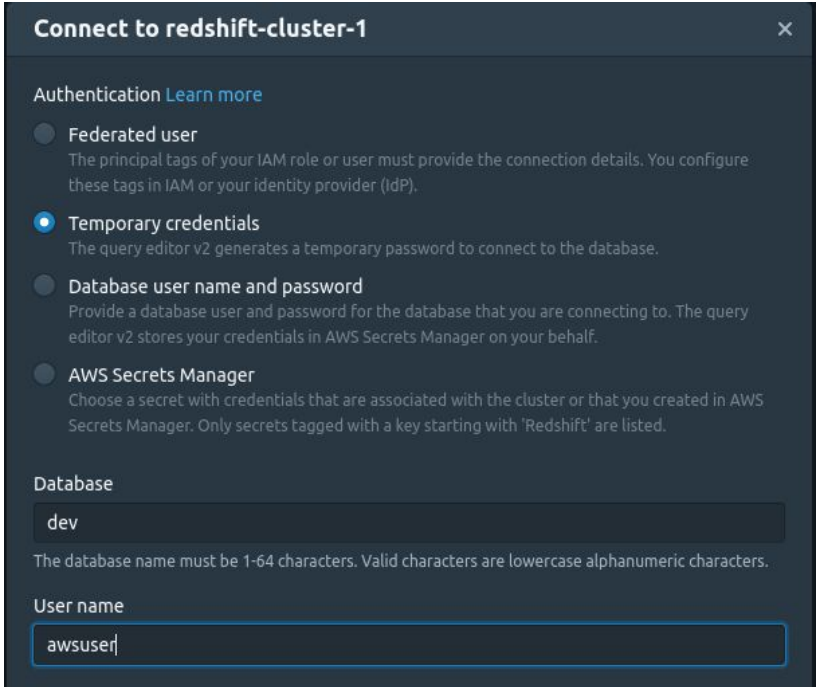


⚠️ Step 2. Create a new IAM User (3)

Save the csv that contains the credentials somewhere safe, log out from AWS, and then use the **Console sign-in URL** to login to AWS with that new user account.

⚠️ Step 2. Create a new IAM User (4)

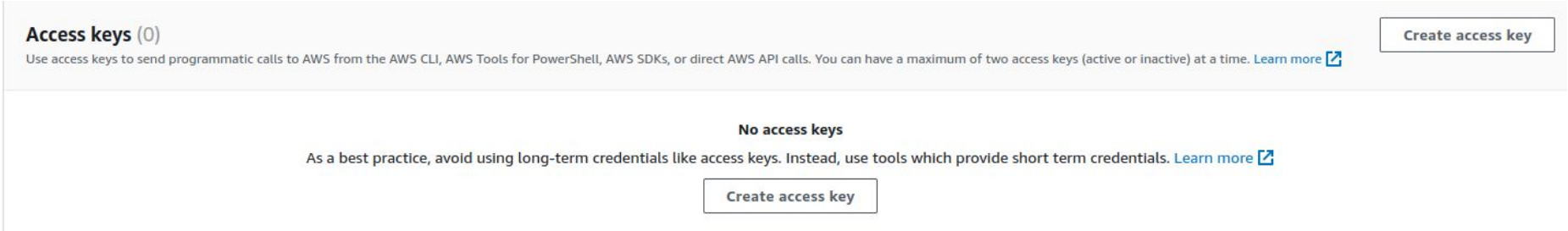
You may now access either Query Editor or Query Editor v2. To connect to your Redshift cluster, pick **Temporary Credentials** and set the Database and Username you've used when setting up the Redshift cluster (if you forgot, you may view them on the Redshift cluster page).



Step 3. Get AWS Credentials

From the new user's IAM console, go to **Users**, then click on the **Security credentials** tab. From there, click on **Create access key**. Then pick the Command Line Interface (CLI) and complete the steps.

Store the AWS credentials somewhere safe. You will use the Access key and Secret access key in your Python scripts.



Step 4. CREATE TABLE Queries

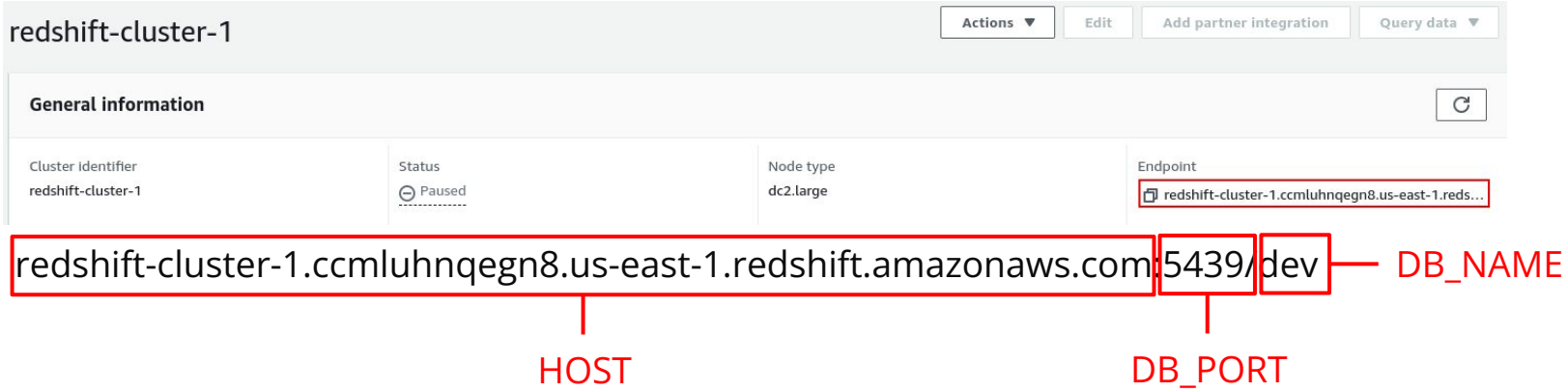
Take a look at sample row values on page Project: **Data Warehouse** > **Project Datasets** and then complete the queries in `staging_events_table_create` and `staging_songs_table_create` variables.

Also, create the CREATE TABLE queries for your final data model according to the specifications in the **rubric** and **Project Instructions** page.

- All data types should be reasonably correct.
- The ordering of fields of `staging_events` should follow the `log_json_path.json` file.
- Final tables should have PRIMARY KEY constraints.
- Some fields in the final tables should have NOT NULL constraints.
- Use `IDENTITY(0,1)` for the data type of the songplays ID field.

Step 5. Test-run create_tables.py

Set up the `dwh.cfg` file with everything you've created so far. The **General information** panel of your Redshift cluster has the information you need to complete the `[CLUSTER]` settings. The value for DB_USER can be found in the **Database configurations** panel.



⚠️Cost-Saving Maneuver

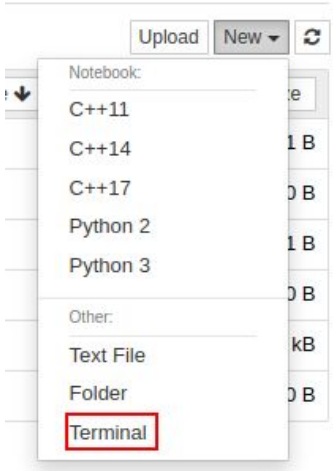
As previously mentioned, please update your paths in the `dwh.cfg` file to:

- `s3://udacity-dend/song_data/A/A/A`
- `s3://udacity-dend/log_data`
- `s3://udacity-dend/log_json_path.json`

The first path ensures that you do not use the entire songs data, which would be massive.

Step 5. Test-run create_tables.py (2)

Now we are ready to test-run the code! Open up a terminal by clicking on **New** > **Terminal**:



and then type in:

```
home root$ python create_tables.py
```

To see if the tables were created successfully, open up the Query Editor page and check the list of tables there.

Step 6. COPY Queries

Here is an example of a working COPY query for `staging_events`:

```
COPY staging_events
FROM {}
iam_role '{}'
region 'us-west-2'
TIMEFORMAT AS 'epochmillisecs'
JSON {};
```

The `TIMEFORMAT` option is if the `ts` field's data type is set to `timestamp`, to ensure the values that are in milliseconds are converted into timestamp objects.

Note: Load the configuration values from the pwd.cfg file rather than hard-coding them. Use the configparser module for this.

Step 7. Test-run load_staging_tables() function

Before working on the INSERT queries, you may want to check if the staging tables are filled correctly. Using the Terminal from the Workspace, you may run the `etl.py` script and then query the staging tables from the Query Editor.

Step 8. INSERT Queries

Complete the INSERT queries in sql_queries.py. Take note of the following requirements in the **rubric** and **Project Instructions** page.

- Duplicate records must be handled properly. More on this on the next slide.
- The `songplays` table should keep only records with page == `NextSong`.

Step 9. Checking for Duplicates

Run the etl.py script again, and use the following query to check for duplicates:

```
SELECT user_id, COUNT(*) as count FROM users GROUP BY user_id ORDER BY count DESC;
```

Good results - no duplicates

Result 1 (96)

user_id	count
18	1
51	1
27	1
61	1
29	1

Acceptable results - some duplicates

Result 1 (96)

user_id	count
15	2
49	2
85	2
80	2
29	2

Bad results - Many duplicates

Result 1 (96)

user_id	count
49	689
80	665
97	557
15	463
44	397

Check out the discussion in this Knowledge Thread (read my answer, in particular).

Step 10. Polishing Up

Lastly, don't forget to accomplish the last two specifications in the **rubric**:

- Create a README file that contains a summary of the project, how to run the Python scripts, and an explanation of the files in the repository.
- Add docstrings to all of your functions.

```
def insert_tables(cur, conn):
    """ Run queries to insert data from staging tables to analytical tables in the Redshift cluster.

    Args:
        cur(psycopg2.cursor): Cursor object from Psycopg2's connection.
        conn(psycopg2.connection): Connection object of Psycopg2.
    Returns: None
    """
```

Q&A

Session Feedback Form

https://airtable.com/shr7WGplyaZIWq8pE

Tips (again)

1. Check the Project Rubric often.

https://review.udacity.com/#!/rubrics/2475/view

2. Have a quick question? Shoot an email to teguhwpurwanto@gmail.com.

Thank you