**Data Engineering with AWS**

Udacity Nanodegrees

**Author:** Nguyen Viet Manh – ManhNV12

# **Overview**

This course includes 4 main contents and 4 projects corresponding to each content:

* **Data Modeling**
  + Project 1: Project: Data Modeling with Apache Cassandra
* **Cloud Data Warehouses**
  + Project 2: Data Warehouses
* **Spark and Data Lakes**
  + Project 3: STEDI Human Balance Analytics
* **Automate Data Pipelines**
  + Project 4: Data Pipelines

Fsofter will master the AWS Data Engineering skills necessary to level up your tech career. Learn data engineering concepts like designing data models, building data warehouses and data lakes, automating data pipelines, and managing massive datasets.

# **Prerequisite**

I recommend that you have intermediate Python, intermediate SQL, AWS, and command line skills.

There are no software and version requirements to complete this Nanodegree program. All coursework and projects can be completed via Student Workspaces in the Udacity online classroom. But in project 4, when you work with AirFlow you can install Airflow in your local env coz in this project on Student Workspaces often gets error with them.

# **Attention**

When joining you will be provided AWS account. In this account, all AWS services are a pay-as-you-go service. Udacity has set a budget for each student to complete their course work. Please understand that these credits are limited and available for you to use judiciously. The budget for this entire course is **$25**. Although, I find about $10 sufficient for most to complete this course. After completing or submitting project, you should shut-down services to save the money.

# **Project 01: Data Modeling with Cassandra**

**Overview:**

A start-up called Sparkify wants to analyze the data they've been collecting on songs and user activity on their new music streaming app. The analysis team is particularly interested in understanding what songs users are listening to. Currently, there is no easy way to query the data to generate the results, since the data reside in a directory of CSV files on user activity on the app.

They'd like a data engineer to create an Apache Cassandra database which can create queries on song play data to answer the questions and wish to bring you on the project. Your role is to create a database for this analysis. You'll be able to test your database by running queries given to you by the analytics team from Sparkify to create the results.

In this project, you'll apply what you've learned on data modeling with Apache Cassandra and complete an ETL pipeline using Python. To complete the project, you will need to model your data by creating tables in Apache Cassandra to run queries. You are provided with part of the ETL pipeline that transfers data from a set of CSV files within a directory to create a streamlined CSV file to model and insert data into Apache Cassandra tables.

Udacity have provided you with a project template that takes care of all the imports and provides a structure for ETL pipeline you'd need to process this data. This project can finish with Udacity Workspaces.

**INSTRUCTION:**

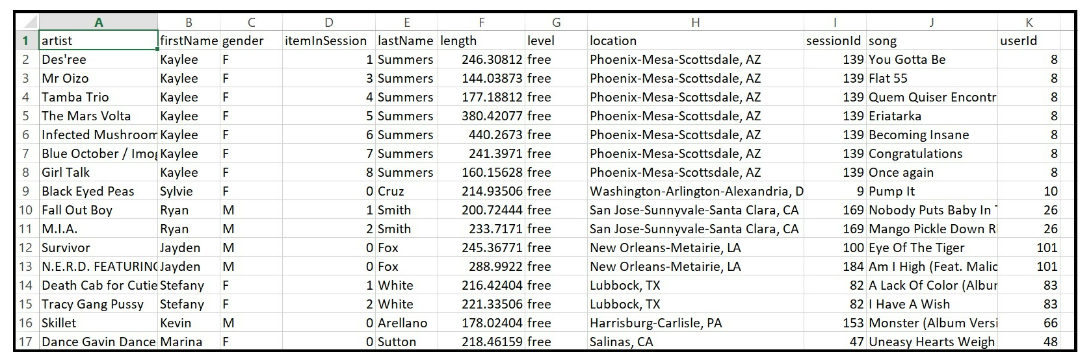
You must complete 3 querys according to project template instruction:

1. Give me the artist, song title and song's length in the music app history that was heard during sessionId = 338, and itemInSession = 4

2. Give me only the following: name of artist, song (sorted by itemInSession) and user (first and last name) for userid = 10, sessionid = 182

3. Give me every user name (first and last) in my music app history who listened to the song 'All Hands Against His Own'

**Data:**



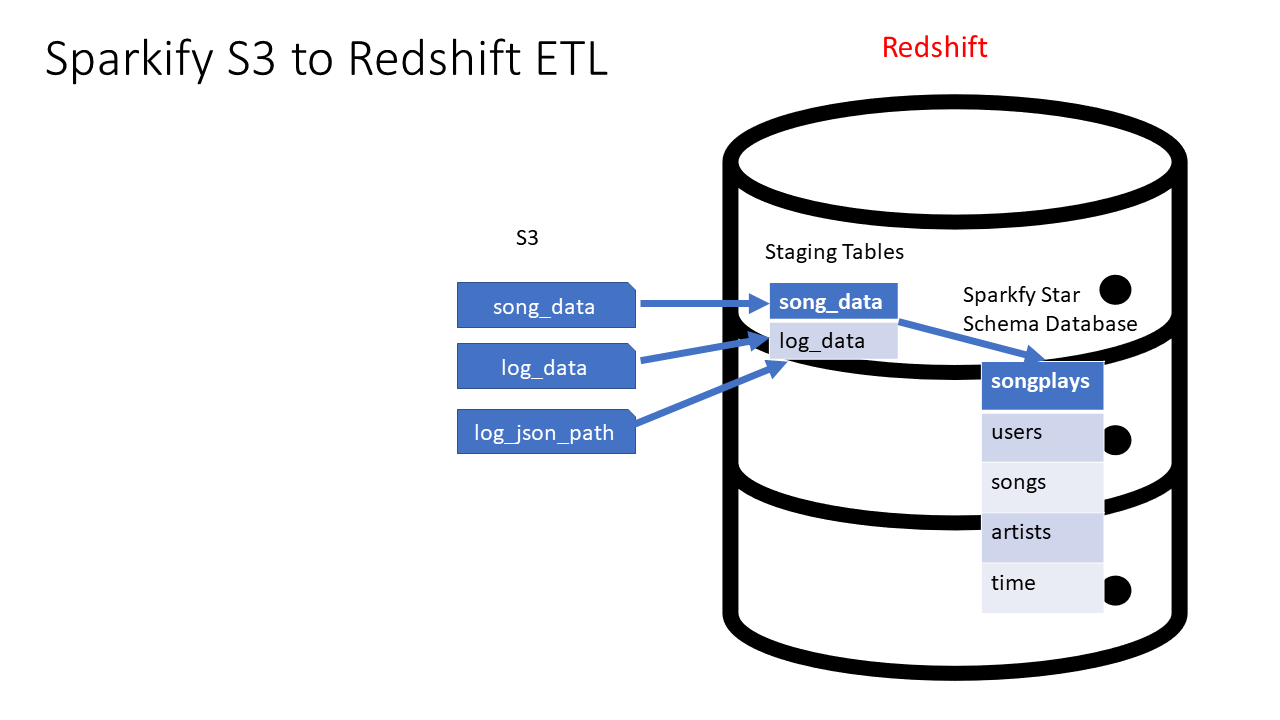
Completing this project is quite simple. You just need to use the SELECT statement to query with the WHERE condition. What I want to note here is getting the correct field names in the database, and when submitting, you need to delete all the TODO lines.

# **Project 02: Project: Data Warehouse**

**Overview:**

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.

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.

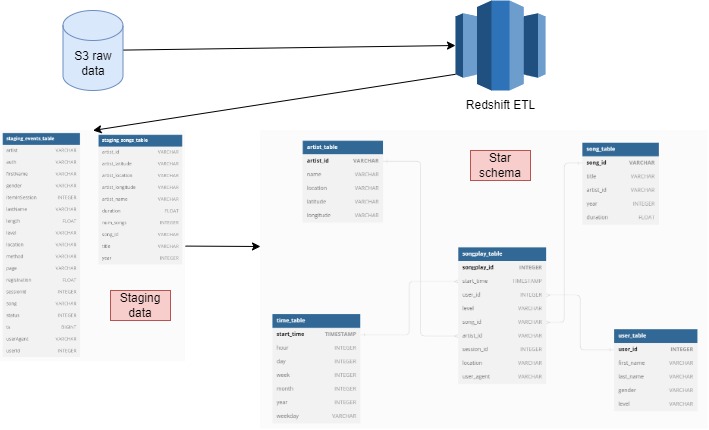


In this project, you'll apply what you've learned on data warehouses and AWS to build an ETL pipeline for a database hosted on Redshift. To complete the project, you will need to load data from S3 to staging tables on Redshift and execute SQL statements that create the analytics tables from these staging tables.

***NOTE:*** *In this project, you must use provided AWS account to use Redshift on AWS Cloud, therefore you should shutdown services after using to avoid AWS money limit.*

**INSTRUCTION:**

* Staging tables
  + **staging\_events:** stores data extracted from JSON logs on user activity. Columns: artist, auth, firstName, gender, itemInSession, lastName, length, level, location, method, page, registration, sessionId, song, status, ts, userAgent, userId
  + **staging\_songs:** stores data extracted from JSON metadata on the songs in the app. Columns: num\_songs, artist\_id, artist\_latitude, artist\_longitude, artist\_location, artist\_name, song\_id, title, duration, year
* Analytical tables
  + Fact Table
    - **songplays:** records in event data associated with song plays i.e. records with page NextSong. Columns: songplay\_id, start\_time, user\_id, level, song\_id, artist\_id, session\_id, location, user\_agent
  + Dimension Tables
    - **users:** users in the app. Columns: user\_id, first\_name, last\_name, gender, level
    - **songs:** songs in music database. Columns: song\_id, title, artist\_id, year, duration
    - **artists:** artists in music database. Columns: artist\_id, name, location, latitude, longitude
    - **time:** timestamps of records in songplays broken down into specific units. Columns: start\_time, hour, day, week, month, year, weekday
* ETL pipeline
  + create\_tables.py will drop all existing tables and create tables a per the queries mentioned in sql\_queries.py.
  + etl.py copy data from s3 to staging table and then populate fact and dimension tables.
* main.ipynb for running complete project flow including setting up aws resources, running aforementioned etl pipeline and cleaning up resources.
* I have done project with my design as following image:



***NOTE:*** *You should draw your design image to be possible pass when submit project*

# **Project 03: STEDI Human Balance Analytics**

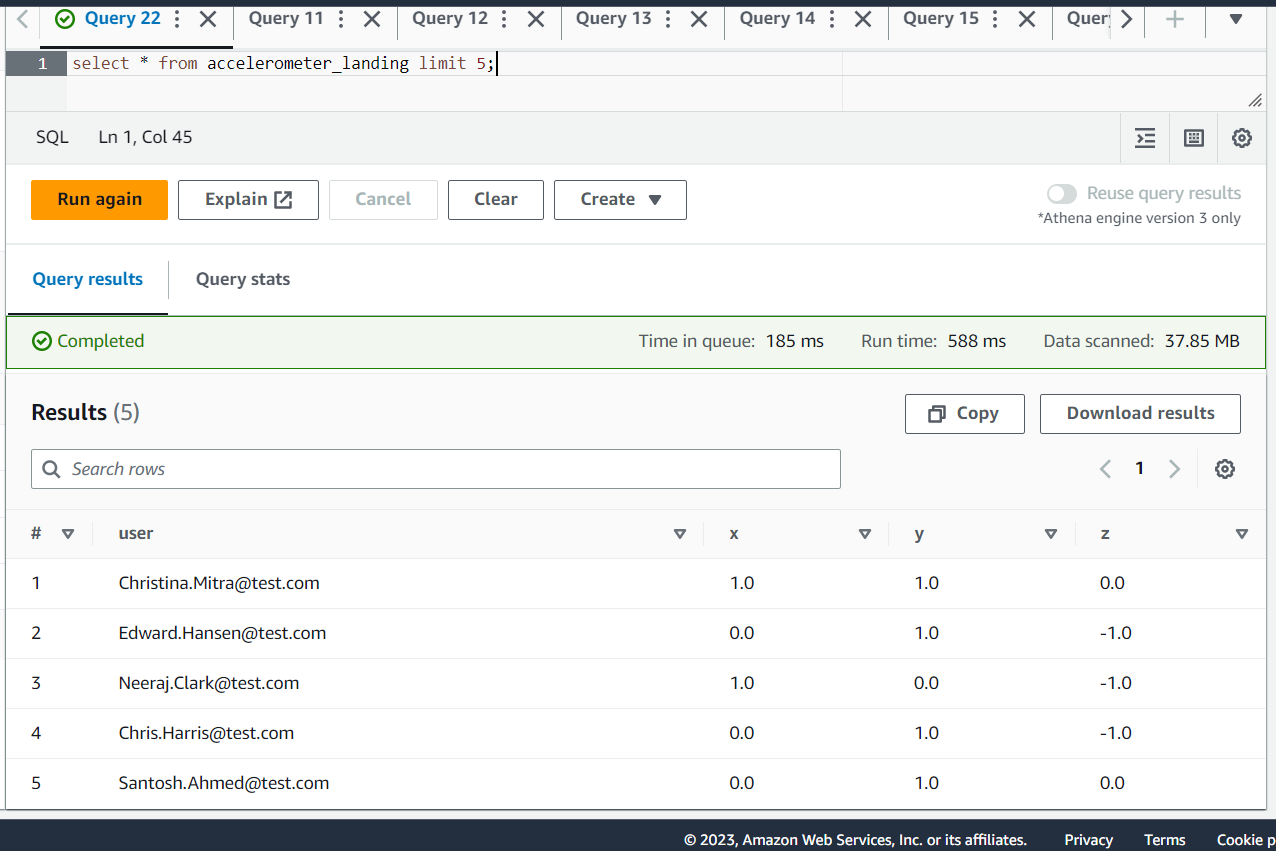
**Overview**

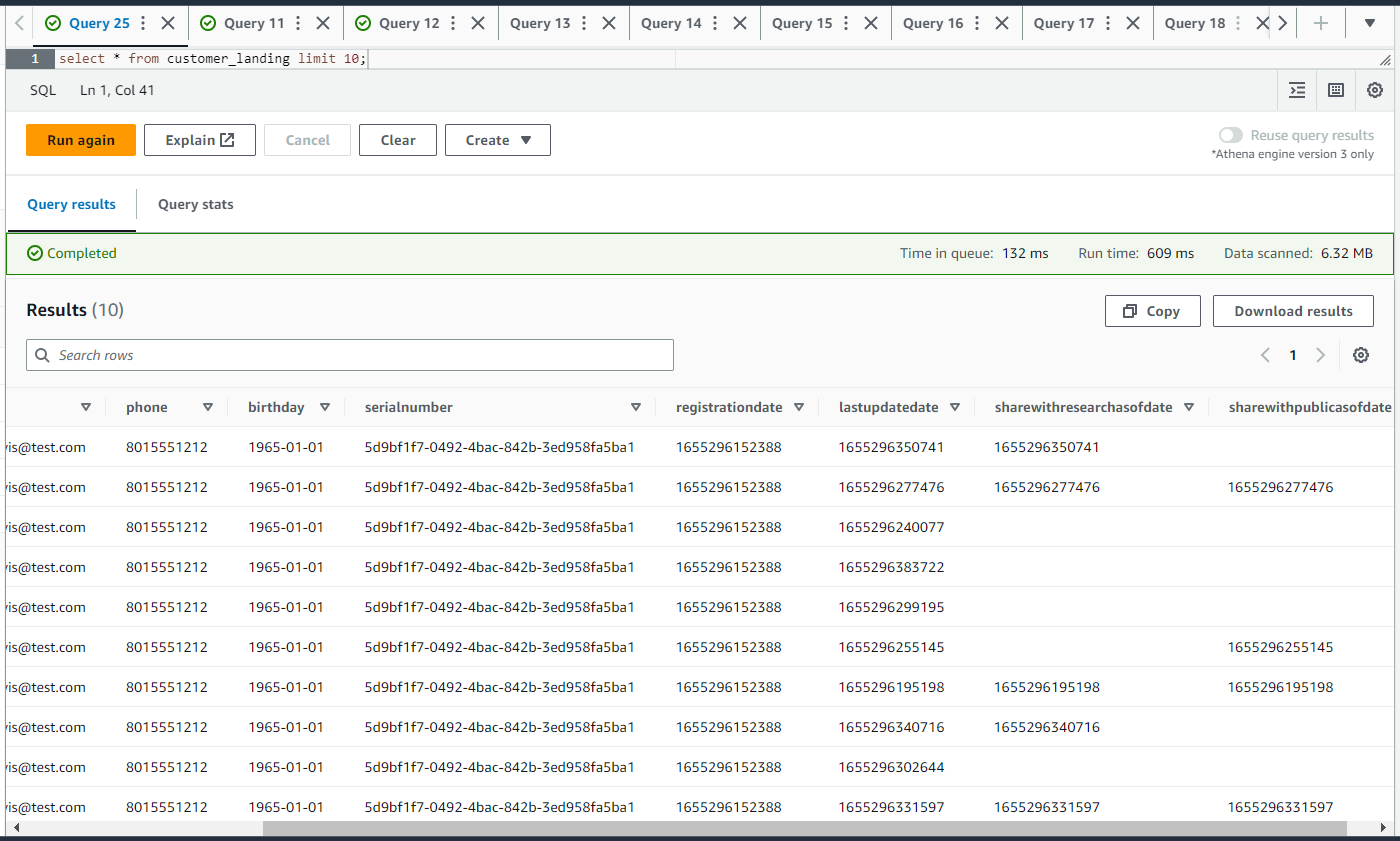
Spark and Human Balance. As you have learned in this course Spark and AWS Glue allow you to process data from multiple sources, categorize the data, and curate it to be queried in the future for multiple purposes. In this project you will directly use the skills you have used, including some of the code you have already written. You will go beyond that to write additional AWS Glue jobs to create curated step trainer data that can be used for machine learning.

As a data engineer on the STEDI Step Trainer team, you'll need to extract the data produced by the STEDI Step Trainer sensors and the mobile app, and curate them into a data lakehouse solution on AWS so that Data Scientists can train the learning model.

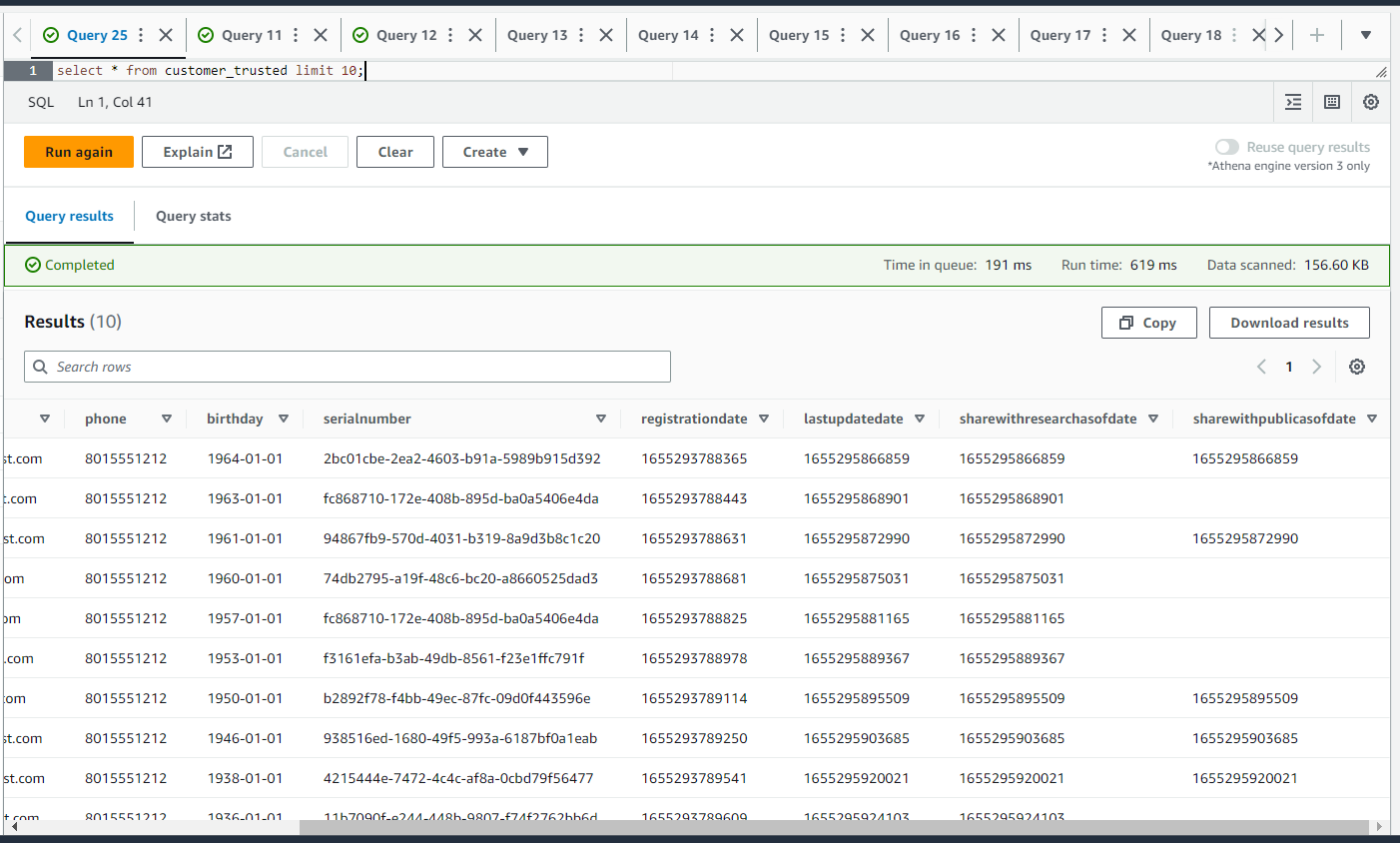
**INSTRUCTION:**

* **Landing zone** 
  + After downloading the dataset provided by STEDI, you need create an S3 bucket and uploaded them to your AWS with your bucket name. Then you open AWS Athena to create 2 sql query scripts to query data from landing zone.
    - accelerometer\_landing
    - sqlcustomer\_landing.sql
    - By running 2 above sql scripts, you will get the results of landing data as below:





* **Trusted zone** 
  + Following the instructions in the project, you must create 3 glue jobs to process data from the Landing zone to the Trusted zone with 3 scripts:
    - customer\_landing\_to\_trusted.py: This is a python script using Spark that sanitizes the Customer data from the Website (Landing Zone) and only stores the Customer Records who agreed to share their data for research purposes (Trusted Zone) (shareWithResearchAsOfDate!=0)
    - accelerometer\_landing\_to\_trusted.py: This is a python script using Spark that sanitizes the Accelerometer data from the Mobile App (Landing Zone) - and only stores Accelerometer Readings from customers who agreed to share their data for research purposes (Trusted Zone) (Join accelerometer landing with customer trusted data on clause user==email)
    - step\_trainer\_landing\_to\_trusted.py: This is a python script using Spark that reads the Step Trainer IoT data stream (S3) for customers who have accelerometer data and have agreed to share their data for research (customers\_curated) (Join step trainer landing with customer curated data on clause serialNumber == customer\_serial Number)
    - Athena customer\_trusted query result:



* **Curated zone** 
  + Finally, you must create 2 glue jobs to process the data from the Trusted zone to the Curated zone:
    - customer\_trusted\_to\_curated.py: This is a python script using Spark that sanitizes the Customer data (Trusted Zone) that only includes customers who have accelerometer data and have agreed to share their data for research (Join customer trusted with accelerometer landing data on clause email==user)
    - step\_trainer\_trusted\_to\_machine\_learning\_curated.py: This is a Python script using Spark that has each of the Step Trainer readings, and the associated accelerometer reading data for the same timestamp, but only for customers who have agreed to share their data (Join step trainer trusted with accelerometer trusted on clause sensorReadingTime==timeStamp)

# **Project 04: Data Pipelines with Airflow**

**Overview**

A music streaming company, Sparkify, has decided that it is time to introduce more automation and monitoring to their data warehouse ETL pipelines and come to the conclusion that the best tool to achieve this is Apache Airflow.

You must create high grade data pipelines that are dynamic and built from reusable tasks, can be monitored, and allow easy backfills. They have also noted that the data quality plays a big part when analyses are executed on top the data warehouse and want to run tests against their datasets after the ETL steps have been executed to catch any discrepancies in the datasets.

The source data resides in S3 and needs to be processed in Sparkify's data warehouse in Amazon Redshift. The source datasets consist of JSON logs that tell about user activity in the application and JSON metadata about the songs the users listen to.

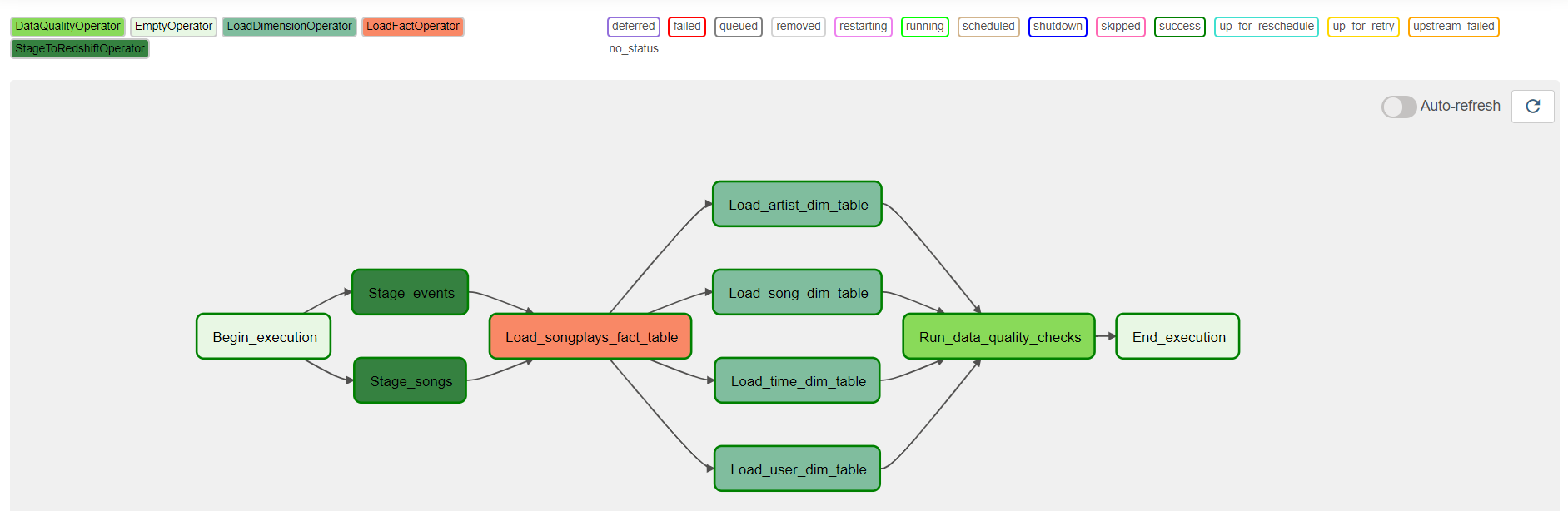
In this project, you will create my own custom operators to perform tasks such as staging the data, filling the data warehouse, and running checks on the data as the final step.

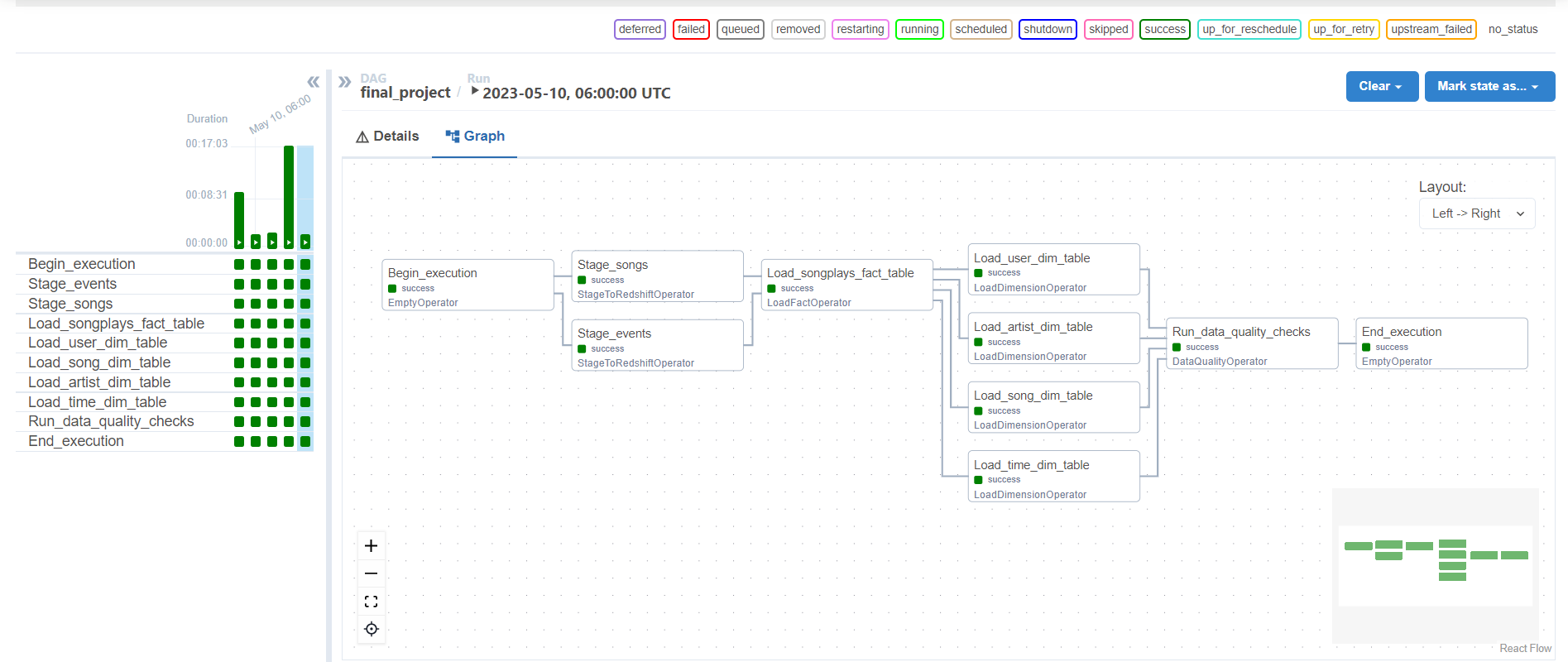
**INSTRUCTION:**

* **Dataset:** 
  + For this project, you’ll be working with two datasets. Here are the s3 links for each:
    - Log data: s3://udacity-dend/log\_data
    - Song data: s3://udacity-dend/song\_data

***Note:*** *Because in this project, Udacity Workspace have an error with Airflow so you can install Airflow on your laptop, and donwload the provided template at project template package*

* **Coding:** 
  + First, I created IAM user and add aws\_credentials into Airflow
  + Then, following exercises in course you must create and configured AWS Redshift Serverless into Airflow
  + This is my instruction for you:
    - Begin\_execution: operator for start DAG
    - Stage\_events, Stage\_songs: operators for loading data from log json data to staging table
    - Load\_songplays\_fact\_table, Load\_user\_dim\_table, Load\_song\_dim\_table, Load\_artist \_dim\_table, Load\_time\_dim\_table: operators for loading data from fact tables into dim tables. Code to insert data into table is written in helpers/sql\_queries.py
    - Run\_data\_quality\_checks: operator for checking NULL data in table on Redshift
    - End\_execution: operator for end DAG
* After create this pipeline, in Airflow your pipeline must be as following:





# **Support Channel**

* **If you meet some errors relate this Nanodegree, you can reach out Udacity Support**:
  + Content: Knowledge Portal Udacity/Mentor Help/ Udacity GPT
  + Technical problems: <https://udacityenterprise.zendesk.com/hc/en-us/requests/new>
  + Data Engineering with AWS
* **Contact Point**: DanPNL (HN+OB)/ ThaoNT39 (ĐN+ Hue +QN)/ VyTT4 (HCM+CT)