Data Warehousing

In this project, you will build an automated data pipeline using AWS services. In this project, you will load CSV files into an **S3 bucket**, set up an **AWS Lambda** function to read the files and insert the data into a **Redshift data warehouse** following a specified schema. You will also interact with both the **data** and **metadata** in the warehouse.

***DISCLAIMER**: Students should be aware that this exercise relies on AWS Free tier and an AWS Redshift Free Trial. If either has exhausted on your AWS account, then this exercise will incur charges on your account**

Requirements

Part 1: Setting Up the AWS Environment

1. Sign Up for AWS Free Tier
    - Sign up for the AWS Free tier, if you haven't made an account yet
    - Sign Up for AWS Redshift Free Trial. Use the link here for additional details
        - The free trial includes $300 credits for Redshift. Lambda, S3, and other services also have free-tier eligibility under AWS Free tier.
2. Create an S3 Bucket

- From the AWS Console, search for **S3** and **create a new S3 bucket**.
- Name the bucket something unique, like ***datawarehouse-bucket-username***.
- Set the region (***make sure this is the same region you'll use for Redshift and Lambda***) and keep the default settings.
- Enable versioning if you want to track file uploads over time.

3. Create an AWS Redshift Cluster
   - Navigate to **Amazon Redshift** in the AWS console.
   - **Create a Redshift cluster** as outlined in the previous lesson, using the dc2.large node type for free-tier eligibility.
   - Configure the cluster, ensuring you note down the **JDBC endpoint**.
   - Set up the database (*e.g., name it* ***datawarehouse***), and keep track of your **admin credentials** (*username and password*).
   - Amazon's documentation has a walkthrough on how to create and work with Redshift Warehouses. Check out the additional instructions **here**.

Part 2: Automating Data Ingestion with AWS Lambda

4. Use **Lambda** to automate the process of reading the CSV files from S3 and loading them into the Redshift data warehouse. (*Note: This task can also be done using AWS Glue instead of a Lambda function. Try and recreate this project using Glue for additional practice*)
   - Go to the AWS Lambda console and **create a new Lambda function**.
   - Choose a **Python 3.8+** runtime and give your function a name *(e.g.,* ***s3_to_redshift***).
   - Attach the necessary **IAM roles**:
     - The role should have permissions to **read from S3** and **write to Redshift**.
     - Attach policies such as **AmazonS3FullAccess** and **AmazonRedshiftFullAccess**.

        - You may also need **AmazonVPCFullAccess** if your Redshift cluster is inside a VPC (Virtual Private Cloud).
5. Modify the Lambda Function
    - Open the included file called `**CleaningLambda.py**`. This file contains the lambda code that will read csv files and push them to our redshift data warehouse. Adjust the conn parameters to match your database
        - dbname='**your-db-name**'
        - user='**your-db-user**'
        - password='**your-db-password**'
        - host='**your-redshift-endpoint**'
6. Once you've modified the code, save the file and convert it to a **ZIP file**:
    - Name the ZIP file (e.g., **lambda_redshift.zip**).
7. Set the **Handler**
    - In the Handler field, set the handler to match the name of your Python file (if it's named **CleaningLambda.py**, set the handler as **CleaningLambda.lambda_handler**).
8. Add an S3 Trigger
    - Scroll down to the Designer section and click **Add Trigger**.
    - Select **S3** as the trigger source.
    - Set the following details:
        - **Bucket**: Choose the S3 bucket you created earlier (e.g., **datawarehouse-bucket-username**).
        - **Event Type**: Select All object create events (this will trigger Lambda whenever a new file is uploaded to the bucket).
        - **Prefix/Suffix**: Optionally, set filters to only trigger when certain files (like CSVs) are uploaded.
9. **Save and Deploy**: After uploading the code and configuring the trigger, click Deploy to save the Lambda function.

Part 3: Working with Data and Metadata

10. Setting Up the Tables in Redshift:
    - Manually **create the primary data tables** in Redshift (**customers, orders**, etc.) before running the Lambda function.
        - Example:

```
CREATE TABLE customers (
    customer_id VARCHAR(50),
    customer_name VARCHAR(100),
    email VARCHAR(100)  );


CREATE TABLE orders (
    order_id VARCHAR(50),
    customer_id VARCHAR(50),
    order_amount INT,
    order_date DATE  );
```

    - Manually **create a metadata table** in Redshift to store information about the data loads (*e.g., when the data was loaded, how many rows were inserted, and which file was used*).
11. Upload Sample CSV Files to S3
    - Upload the `**customer_date.csv**` and `**order_data.csv**` files to the created **S3 bucket from step 2**. This should trigger your lambda function.
12. Querying Data and Metadata to answer the below questions (*Note: Redshift is based off of PostgreSQL. This closely resembles MySQL commands with some minor differences. Double check PostgreSQL commands **here***)
    - Find the **total sales by customer**
    - Find when the **last file was loaded**

Congratulations! This is a simple version of a Data Warehouse