Taller: Procesamiento y Análisis de Datos para una Aerolínea

Consigna para el Taller de Alumnos

Título: Análisis de Retrasos de Vuelos y Satisfacción de Clientes

Descripción:

Eres parte del equipo de análisis de datos de una aerolínea que quiere identificar cómo los retrasos en los vuelos afectan la satisfacción de los pasajeros. Dispones de dos conjuntos de datos clave:

Datos de Entrada

1. `**flights.csv**`: Contiene información sobre los vuelos realizados, como el número de vuelo, origen, destino, y retrasos en minutos.

2. `**feedback.csv**`: Contiene opiniones de los pasajeros sobre su experiencia, incluyendo una calificación de 1 a 5.

Objetivo:

Desarrollar un proceso automatizado para combinar los datos de los archivos proporcionados y generar un informe que calcule el **retraso promedio por vuelo** y su **impacto en la calificación promedio de los pasajeros**. Este informe deberá almacenarse en una base de datos **Amazon Redshift** y cumplir con las siguientes condiciones:

1. **Ejecución por evento:**
    - Crear el bucket S3 s3-training-activity-02.
    - Crear el bucket S3 s3-training-activity-03.
    - La inserción de datos en Redshift no debe realizarse manualmente desde la consola o AWS CLI. En su lugar, se **debe configurar un trigger** en AWS Lambda para que el proceso se active automáticamente cuando se cree un objeto en el bucket de S3 **s3-training-activity-02**
    - El evento deberá ser de tipo **s3:ObjectCreated:*.**

2. **Generación y escritura de datos procesados en S3 y Redshift:**
    - El DataFrame resultante (merged_df), producto de la combinación de los datos, debe ser escrito en Redshfit en la tabla **flight_feedback_summary.**
    - El DataFrame resultante (merged_df), producto de la combinación de los datos, debe ser escrito en el bucket de S3 en formato CSV, el nombre del fichero es **flight_feedback_summary_lab**. El bucket de destino es s3-training-activity-03 con la ruta: s3://s3-training-activity-03/staging/ flight_feedback_summary_lab.csv

- Este fichero debe **excluir el índice** para mantener la estructura limpia y manejable.
- Este fichero debe incluir las cabeceras.

Código:
Este código debes copiarlo en tu lambda:

```
import os
import time
import boto3
import pandas as pd
import awswrangler as wr

# Variables de entorno
S3_BUCKET = os.environ["S3_BUCKET"]
FLIGHTS_KEY = os.environ["FLIGHTS_KEY"]
FEEDBACK_KEY = os.environ["FEEDBACK_KEY"]
REDSHIFT_CLUSTER = os.environ["REDSHIFT_CLUSTER"]
REDSHIFT_DATABASE = os.environ["REDSHIFT_DATABASE"]
REDSHIFT_USER = os.environ["REDSHIFT_USER"]
REDSHIFT_PASSWORD = os.environ["REDSHIFT_PASSWORD"]
REDSHIFT_TABLE = os.environ["REDSHIFT_TABLE"]
REDSHIFT_SCHEMA = os.environ.get("REDSHIFT_SCHEMA", "public")
MAX_RETRIES = int(os.environ.get("MAX_RETRIES", 10))
RETRY_DELAY = int(os.environ.get("RETRY_DELAY", 10))
S3_BUCKET_TARGET = os.environ["S3_BUCKET_TARGET"]


def wait_for_files(s3_client, bucket, keys, max_retries, retry_delay):
    """
    Espera hasta que los archivos especificados existan en el bucket de S3.
    """
    retries = 0
    while retries < max_retries:
        existing_files = [
            key for key in keys if check_file_exists(s3_client, bucket, key)
        ]
        if len(existing_files) == len(keys):
            print("Todos los archivos están disponibles.")
            return True
        print(
            f"Archivos faltantes: {set(keys) - set(existing_files)}. Reintento {retries + 1}/{max_retries}..."
        )
        retries += 1
        time.sleep(retry_delay)
    print("Tiempo de espera agotado. No se encontraron todos los archivos.")
    return False
```

```
def check_file_exists(s3_client, bucket, key):
    """
    Verifica si un archivo existe en un bucket de S3.
    """
    try:
        s3_client.head_object(Bucket=bucket, Key=key)
        return True
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "404":
            return False


def lambda_handler(event, context):
    s3_client = boto3.client("s3")

    # Esperar a que los archivos existan
    keys = [FLIGHTS_KEY, FEEDBACK_KEY]
    if not wait_for_files(s3_client, S3_BUCKET, keys, MAX_RETRIES, RETRY_DELAY):
        return {
            "statusCode": 404,
            "body": "Archivos faltantes en S3. No se pudo completar el proceso",
        }

    try:
        # Rutas de los archivos
        flights_path = f"s3://{S3_BUCKET}/{FLIGHTS_KEY}"
        feedback_path = f"s3://{S3_BUCKET}/{FEEDBACK_KEY}"

        # Leer datos desde S3
        flights_df = wr.s3.read_csv(flights_path)
        feedback_df = wr.s3.read_csv(feedback_path)

        # Calcular retraso promedio y calificación promedio
        delay_avg = (
            flights_df.groupby("flight_number")["delay_minutes"]
            .mean()
            .reset_index(name="average_delay")
        )
        rating_avg = (
            feedback_df.groupby("flight_number")["rating"]
            .mean()
            .reset_index(name="average_rating")
        )

        # Combinar métricas
        merged_df = pd.merge(flights_df, delay_avg, on="flight_number", how="left")
        merged_df = pd.merge(merged_df, rating_avg, on="flight_number", how="left")

        con = wr.redshift.connect_temp(
```

```
            cluster_identifier=REDSHIFT_CLUSTER, database="dev", user="awsuser"
        )

        wr.redshift.copy(
            df=merged_df,
            path=f"s3://{S3_BUCKET}/processed/temp",
            con=con,
            schema=REDSHIFT_SCHEMA,
            table=REDSHIFT_TABLE,
            iam_role=os.environ["IAM_ROLE"],
            mode="overwrite",
        )

        return {
            "statusCode": 200,
            "body": "Archivos procesados exitosamente. Datos combinados generados.",
        }
    except Exception as e:
        return {"statusCode": 500, "body": f"Error durante el procesamiento: {str(e)}"}
```

Variables:

Variable	Valor
S3_BUCKET	s3-training-activity-02
FLIGHTS_KEY	flights-01.csv
FEEDBACK_KEY	feedback-02.csv
REDSHIFT_CLUSTER	redshift-cluster-training
REDSHIFT_DATABASE	dev
REDSHIFT_USER	awsuser
REDSHIFT_PASSWORD	
REDSHIFT_TABLE	flight_feedback_summary
REDSHIFT_SCHEMA	public
MAX_RETRIES	30
RETRY_DELAY	30
S3_BUCKET_TARGET	s3-training-activity-03
IAM ROLE	arn:aws:iam::933263644347:role/myRedshiftRole