Proyecto de Data Engineering: Extracción de Datos de Spotify

Introducción

En este proyecto, he desarrollado una solución de Data Engineering que extrae información de la API de Spotify, específicamente de la playlist "Top 50 Argentina". Utilizando Python, he implementado un proceso ETL (Extracción, Transformación y Carga) que permite almacenar los datos relevantes en una base de datos Redshift.

Objetivo

El objetivo principal de este proyecto es automatizar la extracción y carga de los tracks más populares de Argentina en Spotify, facilitando su análisis posterior. Además, el sistema incluye notificaciones por correo electrónico que informan sobre el éxito o fallo del proceso.

Tecnologías Utilizadas

- **Python**: Para la implementación del ETL.
- **psycopg2 y SQLAlchemy**: Para la conexión y manipulación de la base de datos PostgreSQL y Redshift.
- **Airflow**: Para la programación y gestión de tareas automatizadas.
- **JSON**: Para el almacenamiento de credenciales.

Detalles del Proyecto

1. Creación de la Base de Datos y Tablas

El primer paso en mi proyecto fue crear la base de datos spotify_db y la tabla PlaylistTracks. El código se encarga de verificar si la base de datos ya existe y, de no ser así, la crea. Luego, se crea la tabla con la estructura necesaria para almacenar información sobre los tracks.

Código: *DB_PlaylistTracks.py*

2. Proceso ETL

El proceso ETL se divide en varias funciones clave:

- Cargar Credenciales: Se cargan las credenciales desde un archivo JSON para conectarse a la API de Spotify y a Redshift.
- Obtener Token de Acceso: Se realiza una autenticación con la API de Spotify para obtener un token de acceso necesario para las solicitudes.
- Extraer Datos de la Playlist: Se consultan los datos de la playlist "Top 50 Argentina" utilizando el token de acceso.
- Conectar a Redshift y Cargar Datos: Se establecen conexiones con Redshift y se insertan los datos extraídos.

Código: *DB_PlaylistTracks.py*

Configuración de Airflow

Para la automatización, utilicé Airflow. Creé un DAG que se encarga de ejecutar el script ETL diariamente. Además, incluí funciones de callback que envían correos electrónicos para notificar si el proceso se completó con éxito o si hubo algún fallo.

Código*: Dags_PruebaMailSpotify.py*

Resultados

El sistema está diseñado para ejecutarse diariamente, garantizando que la información de la playlist se mantenga actualizada en la base de datos. Los correos electrónicos de notificación permiten un monitoreo constante del estado del proceso.

Conclusión

Este proyecto no solo demuestra la capacidad de automatizar el flujo de datos desde una API hacia una base de datos, sino que también proporciona un marco para futuras ampliaciones, como el análisis de tendencias en los datos extraídos. Estoy emocionado de continuar mejorando y expandiendo este proyecto.