Proyecto de Procesamiento de Datos

Juan Felipe Cardona Arango

24-10-2024

Metodología

La tarea se estructuró en cuatro fases, cada una dirigida al desarrollo de los requisitos específicos:

1. **Fase 1: Programación Lineal del Proceso de Datos**
2. **Fase 2: Reestructuración Basada en Programación Orientada a Objetos (POO)**
3. **Fase 3: Reestructuración basada en el framework Apache Spark**
4. **Fase 4: Arquitectura en AWS**

- **Objetivos**:
  - Limpieza de datos.
  - Combinación de datos según el modelo relacional.
  - Encapsular la lógica de carga, limpieza, combinación y almacenamiento.
  - Adaptar el código a PySpark.
  - Proponer un diagrama de arquitectura híbrida serverless para la transferencia de datos desde un entorno on-premise hacia la nube.

Fase 1: Programación Lineal del Proceso de Datos

Para abordar el problema, primero se realizó un análisis exploratorio del conjunto de datos utilizando Excel y el diagrama entidad-relación proporcionado. El análisis inicial mostró problemas de valores nulos, caracteres adicionales no deseados y formatos inconsistentes.

Para la tabla `film`, se desarrolló una función específica que:

- Procesa cada columna, eliminando caracteres no válidos según el tipo de dato esperado.
- Ajusta el formato en todas las tablas para asegurar una normalización adecuada.

Posteriormente, las tablas fueron unidas utilizando el método `merge` de Pandas, generando un DataFrame consolidado. Este proceso se realizó sin aplicar aún los principios de Programación Orientada a Objetos (POO).

Métrica de Autoconfianza: 4 (Confiado)

Fase 2: Reestructuración Basada en Programación Orientada a Objetos

El código fue reestructurado para encapsular la lógica en la clase `FilmDataProcessor`, con métodos específicos para cada fase del procesamiento:

- `__init__(self, file_path)`: Inicializa la clase con la ruta del archivo.
- `load_data(self)`: Carga las hojas del archivo Excel.
- `clean_data(self)`: Limpia datos en cada DataFrame.
- `combine_data(self)`: Une las tablas procesadas.
- `save_and_download(self, df_final, output_file_name)`: Guarda y descarga el DataFrame.

Métrica de Autoconfianza: 4 (Confiado)

Fase 3: Reestructuración Basada en Apache Spark

El código fue adaptado a PySpark y ejecutado en Google Colab, con una estructura dividida en tres secciones:

- Sección 1 y 2: Experimentación.
- Sección 3: Código final.

Métrica de Autoconfianza: 3 (Inseguro)

Fase 4: Arquitectura en AWS

Se propone un diagrama de arquitectura híbrida *serverless*, utilizando:

- **AWS DataSync**: Transferencia de datos hacia AWS S3.
- **AWS Glue**: Procesamiento de datos y catalogación de metadatos.
- **Amazon Athena y Amazon QuickSight**: Consultas y visualización de resultados.
- **Amazon SageMaker**: Modelos predictivos.
- **AWS IAM y AWS CloudWatch**: Seguridad y monitoreo continuo.

Nota: Para una arquitectura optimizada con Apache, se reemplazaría AWS Glue por AWS EMR.

Análisis del Dataset Procesado

Se realizaron las siguientes preguntas orientadoras para el análisis en Looker Studio:

- ¿Existen tendencias estacionales en las rentas?
- ¿Qué películas generan más ingresos?
- ¿Cuál es la duración típica de una renta y su impacto en el inventario?
- ¿Cuáles son las diferencias en rentas por tienda?
- ¿Qué géneros o características de películas atraen más rentas?

Para visualizar el análisis en Looker Studio: Informe Looker Studio.

Enlaces a Productos de Cada Fase

- **Código en Programación Lineal**: Enlace
- **Código en Programación Orientada a Objetos**: Enlace
- **Código en Apache Spark**: Enlace
- **Diagrama Arquitectura AWS**: Enlace