

Propuesta de Solucion para el Challenge de Data Engineer SSR

Por Julian Falvo

Introduccion

Este documento presenta una propuesta de solucion para el challenge propuesto, el cual consiste en analizar un conjunto de datos de ventas y generar diversos reportes. Se utilizara Spark ya que es una herramienta de procesamiento de datos a gran escala, para llevar a cabo las transformaciones y calculos necesarios.

Comprension del Problema

El objetivo principal es procesar un conjunto de datos en formato Parquet, que representan ventas de un marketplace, y realizar las siguientes tareas:

- **Lectura y Validacion de Datos:** Leer los archivos Parquet correspondientes a los primeros 7 días del mes, validar la integridad de los datos y detectar posibles faltantes.
- **Calculo de Métricas:** Calcular la facturacion total por vendedor y el total de unidades vendidas por producto.
- **Generacion de Ranking:** Crear un ranking de vendedores basado en su facturacion total.
- **Almacenamiento:** Guardar los resultados en formato CSV y Parquet particionado.

Propuesta de Solucion

1. Lectura y Validacion de Datos

- **Lectura:** Utilizar PySpark para leer los archivos Parquet correspondientes a cada día del mes.
- **Union:** Unir los DataFrames resultantes en un único DataFrame, manejando de forma adecuada las columnas faltantes.
- **Validacion:** Verificar la existencia de todos los días esperados y registrar los faltantes.

2. Calculo de Métricas

- **Facturacion Total por Vendedor:** Agrupar los datos por vendedor y calcular la suma de la facturacion total.
- **Unidades Vendidas por Producto:** Agrupar los datos por producto y calcular la suma de las unidades vendidas.

3. Generacion de Ranking

- **Ranking:** Utilizar la funcion rank() de PySpark para asignar un ranking a cada vendedor basado en su facturacion total.
- **Almacenamiento:** Guardar el ranking en un archivo CSV.

4. Almacenamiento Particionado

- **Particionamiento:** Particionar el DataFrame final por año, mes y vendedor para mejorar la eficiencia de las consultas y facilitar la escalabilidad.
- **Escritura:** Guardar el DataFrame particionado en formato Parquet.

Explicacion

El código proporcionado implementa la solución de manera clara y concisa. Los comentarios explican cada paso del proceso:

- **Lectura y Union:** Se leen los archivos Parquet y se unen en un único DataFrame.
- **Validacion:** Se compara la lista de días esperados con los encontrados y se registran los faltantes.
- **Calculo de Métricas:** Se calculan la facturación total por vendedor y las unidades vendidas por producto utilizando las funciones groupBy y agg.
- **Ranking:** Se asigna un ranking a cada vendedor utilizando la función rank() y se guarda en un archivo CSV.
- **Particionamiento y Escritura:** Se particiona el DataFrame por año, mes y vendedor y se guarda en formato Parquet.



Subida de datos de Facturacion de ventas a Azure.

Importar Librerías y Configurar:

Se importan las librerías necesarias para interactuar con Azure Table Storage (`azure.data.tables`) y manejar archivos (`os`).

Se carga un archivo de configuración `.env` que contiene las credenciales de acceso a la cuenta de almacenamiento de Azure (nombre de cuenta y clave). Estas credenciales se almacenan de forma segura para evitar exponerlas directamente en el código.

Establecer Conexión a Azure Table Storage:

Se crea una cadena de conexión utilizando las credenciales obtenidas del archivo `.env`. Esta cadena de conexión es esencial para establecer una comunicación segura con la cuenta de almacenamiento.

Se crea un cliente de tabla (`table_service_client`) utilizando la cadena de conexión. Este cliente servirá como punto de entrada para realizar operaciones en la tabla.

Se obtiene un cliente específico para la tabla "FacturacionVentas" (`table_client`).

Procesar Datos desde compressedData:

Existe un proceso previo para cargar los datos desde el archivo comprimido `compressedData`. Este proceso se parsea usando `spark` para facilitar el manejo.

Los datos procesados se almacenan en un `DataFrame` llamado `dfUnida`.

Cargar Datos en Azure Table:

Se itera sobre cada fila del `DataFrame` `dfUnida`.

Por cada fila, se crea un diccionario que representa una entidad en Azure Table. Las claves del diccionario corresponden a las columnas del `DataFrame` y los valores son los datos de la fila.

Se utilizan las columnas `sellerId` y `id` como `PartitionKey` y `RowKey`, respectivamente. Estas claves son fundamentales para organizar y acceder a las entidades en la tabla.

Se intenta crear la entidad en la tabla utilizando el método `create_entity` del cliente de tabla.

Si la entidad ya existe (es decir, si una entidad con la misma `PartitionKey` y `RowKey` ya está en la tabla), se captura una excepción `ResourceExistsError`. En este caso, se puede optar por actualizar la entidad existente o simplemente ignorar el error.

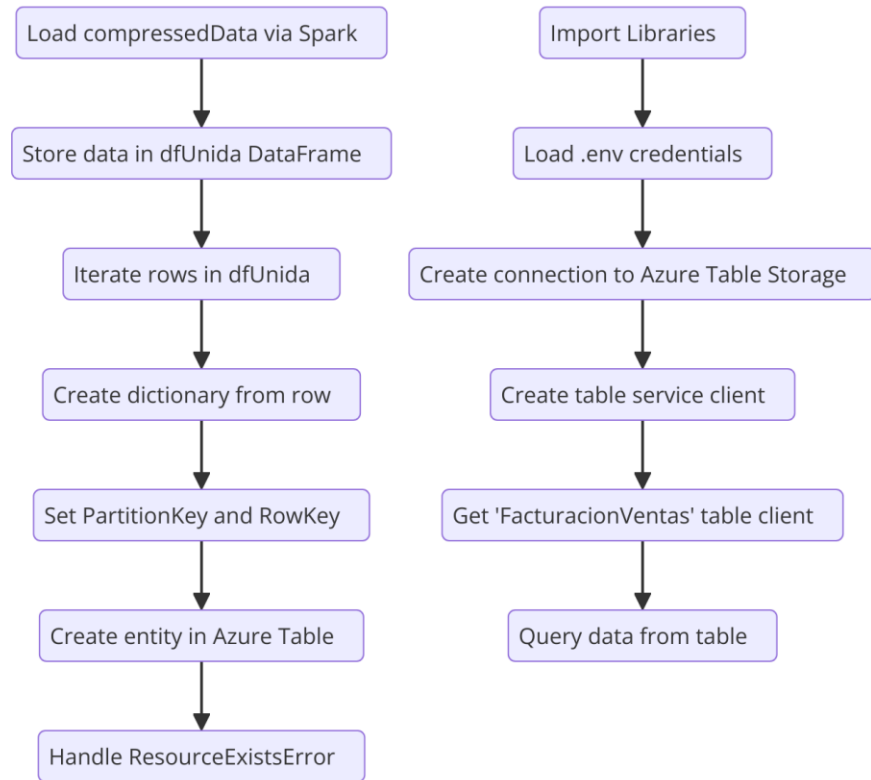
Consultar Datos en Azure Table :

Se incluyen ejemplos de cómo consultar datos en la tabla utilizando diferentes filtros:















Consultar todas las entidades.

Consultar entidades con un `PartitionKey` específico.

Consultar entidades con un filtro más complejo (combinando PartitionKey y un rango de valores para otra propiedad).



¿Qué tipo de datos se necesita para cada campo para copiar correctamente a una Azure Table?

Nombre de propiedad	Tipo	Valor	
PartitionKey	String	sellerId	
RowKey	String	id	
currency	String	Especifique un valor para mantener la propiedad	 
price	Double	Especifique un valor para mantener la propiedad	 
sales	Int64	Especifique un valor para mantener la propiedad	 
category	String	Especifique un valor para mantener la propiedad	 
title	String	Especifique un valor para mantener la propiedad	 
GMV	Double	Especifique un valor para mantener la propiedad	 
Agregar propiedad			