Introducción

Imaginemos un escenario en el que tenemos un conjunto de datos masivo que contiene información de ventas de una cadena minorista con tiendas en múltiples regiones. El objetivo es analizar los datos para obtener información sobre el rendimiento de las ventas, el comportamiento de los clientes y la gestión del inventario.

Para poder trabajar con los datos, necesitaréis diseñar un proceso **ETL**.

EXTRACCIÓN de datos

En esta sección, vamos a describir los detalles de los datos que los estudiantes tendrán que ingerir para el proyecto. Cubriremos datos de diversas fuentes, como archivos CSV, bases de datos y Kafka.

CSV Data

Los datos CSV, que tendréis que crear vosotros, contienen información de ventas de diferentes tiendas en múltiples regiones. Cada fila representa una transacción de venta con varios atributos como date, store ID, product ID, quantity sold, and revenue generated. Los datos también tendréis que crearlos vosotros, de manera aleatoria con **Faker** y **Random** tiene que poner algunos **valores** como **nulos**, **vacíos**,**errores** de **formato**, etc

Formato del archivo

- **File Name:** sales_data.csv
- **Delimiter**: Comma (,)
- **Columns**:
    - Date (YYYY-MM-DD)
    - Store ID (Numeric)
    - Product ID (Alphanumeric)
    - Quantity Sold (Numeric)
    - Revenue (Numeric)

Database Data

La base de datos contiene información adicional sobre las tiendas, como su ubicación y datos demográficos. Estos datos se unirán a los de ventas para analizar las tendencias de ventas en función de factores geográficos. Los datos también tendréis que crearlos vosotros, de manera aleatoria con **Faker** y **Random** tiene que poner algunos **valores** como **nulos**, **vacíos**,**errores** de **formato**, etc

Detalles de BBDD

- **Database Type:** PostgreSQL
- **Host**: localhost
- **Port**: 5432
- **Database Name:** retail_db
- **Table Name**: Stores
- Columns:
    - store_id,
    - store_name,
    - location,
    - demographics

Kafka Data

Kafka transmite en tiempo real los datos de ventas generados por las transacciones en línea. Este flujo de datos se utilizará para el análisis en tiempo real con el fin de supervisar las ventas prácticamente en tiempo real, detectar anomalías y activar alertas para intervenir a tiempo. Los datos también tendréis que crearlos vosotros, de manera aleatoria con **Faker** y **Random** tiene que poner algunos **valores** como **nulos**, **vacíos**,**errores** de **formato**, etc

- **Topic Name**: sales_stream
- **Message Format:** JSON
- Attributes:
    - timestamp: UNIX timestamp (milliseconds)
    - store_id: Numeric
    - product_id: Alphanumeric
    - quantity_sold: Numeric

- revenue: Numeric

Data Lake

Un Data Lake es un repositorio centralizado que permite almacenar todos sus datos estructurados y no estructurados a cualquier escala. Los Data Lake almacenar datos en su formato nativo, lo que significa que no tiene que preprocesar o estructurar los datos antes de almacenarlos. Las características principales son:

1. **Esquema:** permiten almacenar datos en bruto sin imponer un esquema. El esquema se aplica cuando se leen los datos, lo que aporta flexibilidad en el análisis de datos.
2. **Escalabilidad**: están diseñados para escalar horizontalmente, lo que permite a las organizaciones almacenar y procesar cantidades masivas de datos de diversas fuentes sin cambios significativos en la infraestructura.
3. **Soporte para datos estructurados y no estructurados**: admiten

   tanto datos estructurados (como bases de datos relacionales) como datos no estructurados (como archivos de texto, imágenes, vídeos).
4. **Almacenamiento rentable**: suelen aprovechar soluciones de

   almacenamiento rentables, como el almacenamiento en la nube o los sistemas de archivos distribuidos, lo que los hace económicos para almacenar grandes volúmenes de datos.

Ejemplos:

- HDFS
- Amazon S3 (Simple Storage Service)
- Plataforma de datos Cloudera (CDP)
- Google Cloud Storage (GCS)
- Microsoft Azure Data Lake Storage (ADLS)

Los datos obtenidos en la fase de extracción se tendrán que almacenar en un Data Lake, disponéis 2 herramientas: S3 a través de LocalStack o través de AWS. Mi recomendación es que uséis LocalStack y luego, cuando funcione, intentéis montar el clúster en AWS.

LocalStack

LocalStack es un **emulador de servicios en la nube** que se ejecuta en un único contenedor en su portátil o en su entorno CI. Con LocalStack, puede ejecutar sus aplicaciones de AWS o Lambdas completamente en su máquina local sin conectarse a un proveedor de nube remoto.

LocalStack admite un número cada vez mayor de servicios de AWS, como AWS Lambda, S3, Dynamodb, Kinesis, SQS, SNS, etc. La versión Pro de LocalStack soporta APIs adicionales y características avanzadas. Puede encontrar una lista completa de las API compatibles en nuestra página Feature Coverage.

LocalStack también proporciona funciones adicionales para facilitarle la vida como desarrollador en la nube. Consulte las Guías del usuario de LocalStack para obtener más información.

La imagen se encuentra aquí.

TRANSFORMACIÓN de datos

Dentro del **Data Lake** será necesario tratar los valores que falten utilizando **SPARK**, eliminar los duplicados y convertir los tipos de datos según sea necesario.

Todos los datos del Data lake tendrán además de las columnas de datos 2 añadidas:

- **Tratados**: si han sido o no tratados en el proceso de transformación. Cuando se termine este proceso deben quedar
- **Fecha Inserción:** fecha UTC de la inserción.

Tratamiento de los valores perdidos

Los valores perdidos pueden afectar significativamente a los resultados del análisis. Será necesario aplicar estrategias para tratar eficazmente los valores perdidos, como:

- **Imputación**: Rellenar los valores perdidos con un valor calculado o estimado (por ejemplo, media, mediana, moda).
- **Supresión**: Eliminar filas o columnas con valores omitidos, en función de la importancia de los datos omitidos.

Eliminación de duplicados

Los registros duplicados pueden distorsionar los resultados de los análisis y dar lugar a conclusiones inexactas. Será necesario aplicar métodos para identificar y eliminar duplicados, como, por ejemplo

- **Identificar filas duplicadas:** Comprobar si hay filas con valores idénticos en todas las columnas.
- **Eliminar filas duplicadas:** Mantener sólo una instancia de cada fila duplicada, eliminando el resto.

Considera los duplicados basándose en un subconjunto de columnas en lugar de en todas las columnas.

Conversión de tipos de datos

Los tipos de datos deben ser coherentes y apropiados para el análisis. Será necesario convertir los tipos de datos según sea necesario, por ejemplo

- **Datos numéricos:** Convertir cadenas que representan valores numéricos en tipos de datos numéricos (por ejemplo, enteros, flotantes).
- **Datos de fecha y hora:** Convertir representaciones de cadenas de fechas y horas en objetos datetime para facilitar su manipulación.

Detección y tratamiento de valores atípicos

Implantar métodos para detectar y tratar los valores atípicos, que pueden afectar significativamente a los resultados de los análisis.

LOAD: Data Warehouse

Un **Data Warehouse** es un depósito centralizado que almacena grandes volúmenes de datos estructurados y no estructurados procedentes de diversas fuentes de una organización. Está diseñado para apoyar los procesos analíticos y de elaboración de informes, permitiendo a los usuarios tomar decisiones de negocio informadas basadas en datos históricos y actuales.

Las principales características de un almacén de datos son

1. **Integración**: Los datos procedentes de fuentes dispares de toda la organización se consolidan y estandarizan dentro del almacén de datos.
2. **Orientación temática**: Los datos de un almacén de datos se organizan en torno a áreas temáticas o dominios empresariales específicos, como ventas, marketing, finanzas u operaciones.
3. **Variable en el tiempo**: Los datos de un almacén de datos suelen incluir información histórica, lo que permite a los usuarios analizar tendencias y patrones a lo largo del tiempo.
4. **No volátiles**: Una vez que los datos se cargan en el almacén de datos, rara vez, o nunca, se alteran o borran.

Ejemplos:

1. **Amazon Redshift**: Un servicio de almacén de datos en la nube a escala de petabytes y totalmente gestionado, ofrecido por Amazon Web Services (AWS).
2. **Snowflake**: Otra solución de almacenamiento de datos en la nube que ofrece características como escalado automático, separación de almacenamiento y computación, y soporte para datos semiestructurados.

3. **Google BigQuery**: Un almacén de datos sin servidor y altamente escalable ofrecido por Google Cloud Platform (GCP).
4. **Almacén de datos SQL de Microsoft Azure**: Un servicio de almacén de datos totalmente gestionado y altamente escalable en la plataforma en la nube Azure.

Nuestra BBDD será **Amazon Redshift** a través de **LocalStack o** usar los servicios de AWS. Mi recomendación es que uséis LocalStack y luego, cuando funcione, intentéis montar el clúster en AWS.

Obtendremos los datos del Data Lake y los almacenaremos en cuatro tablas que permitirán responder a las preguntas dadas en los 4 tipos de preguntas analíticas:

- Análisis de ventas
- Análisis geográfico
- Análisis demográfico
- Análisis temporal

IMPORTANTE: Guardar los datos de manera que sean eficientes las consultas que tendréis que realizar en el siguiente apartado, por lo que **antes de crearlas ver las preguntas**, tendréis que agrupar los datos de manera distinta, **todas** las tablas tendrán **fecha de inserción**. Por cada grupo de preguntas, una tabla distinta.

Data analytics

Utiliza **Apache Spark** para responder a las siguientes preguntas analíticas en el **Data Lake**, además realice las mismas consultas utilizando **python** en su elección de **Data Warehouse**.

a. Análisis de ventas:

- ¿Qué tienda tiene los mayores ingresos totales?
- ¿Cuáles son los ingresos totales generados en una fecha concreta?
- ¿Qué producto tiene la mayor cantidad vendida?

b. Análisis geográfico:

- ¿Cuáles son las regiones con mejores resultados en función de los ingresos?
- ¿Existe alguna correlación entre la ubicación de la tienda y el rendimiento de las ventas?

c. Análisis demográfico:

- ¿Cómo varía el rendimiento de las ventas entre los distintos grupos demográficos?
- ¿Existen productos específicos preferidos por determinados grupos demográficos?

d. Análisis temporal:

- ¿Cómo varía el rendimiento de las ventas a lo largo del tiempo (diariamente, semanalmente, mensualmente)?
- ¿Existen tendencias estacionales en las ventas?

Estructura del proyecto

Para este proyecto tendréis que seguir la siguiente estructura:

data_bda/: Esta carpeta contiene todos los datos necesarios para el análisis.

- **json/**: Aquí se almacenan los archivos json.
- **text/:** Aquí se almacenan los archivos txt.
- **csv/:** Aquí se almacenan los archivos csv.

data_generation/: Esta carpeta contiene todos los archivos python que generen los archivos anteriores, **tienen que generarlos en la carpeta especificada.**

apps_bda/: Esta carpeta contiene el resto de archivos py.

- **data_integration.py**: Almacena datos en s3
- **data_transformation.py**: Organiza/une/limpia los datos.
- **data_load.py**: Guarda los datos en el Data Warehouse.
- **data_analysis.py**: documento donde se realiza el análisis de datos.

Si existe algún archivo/carpeta más que sea necesario **comentarlo** con el profesor.

Ejercicio Extra

A través de una herramienta de planificación de eventos, modifica tu solución para que el proceso sea cíclico y continuo a través de la librería https://docs.python.org/es/3.8/library/sched.html

1. **Generación de datos:** cada 30 segundos, kafka contínuo.
2. **Extracción de datos:** cada minuto, suponemos que los datos son siempre "nuevos".
3. **Transformación:** cada 30 segundos, **solo los datos no tratados y nuevos**. Se puede hacer de muchas maneras, por ejemplo, **consultando** la **última fecha de inserción** en el Data Warehouse.
4. **Carga de datos :** cada minuto