**LABORATORIO 5 – INTELIGENCIA DE NEGOCIOS – PROCESO ETL (EXTRACT, TRANSFORM, LOAD)**

**Integrantes del equipo:**

* Juan José Beltrán Ruiz: 201819446
* Santiago Bobadilla: 201820728

**Documentación del Proceso ETL**

A continuación, se muestra cada uno de los pasos de la guía, debidamente respaldado por una captura de pantalla:

Lo primero que se hace es crear el usuario Grupo8BI, en este caso, se puede ver desde la interfaz de PG Admin4 el usuario con sus detalles, en este caso, se ve que es el dueño de todas las tablas desarrolladas en el laboratorio. Adicionalmente, se pueden ver los permisos que se le otorgaron a este usuario:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 1: Creación del usuario propietario de la BD

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Ilustración 2: Privilegios del usuario Grupo8BI

Más adelante, se pueden ver las imágenes que sustentan el proceso de creación de la base de datos, teniendo en cuenta que ya se tiene el usuario que será propietario:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 3: Creación de la base de datos

Lo siguiente que se hace, modificar el archivo “. wslconfig”, el cual dirá cuánta RAM se le debe asignar a Docker:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 4: Modificación del archivo .wslconfig

Después de hacer la modificación de este archivo, despliega airflow, primero creando la imagen y posteriormente desplegándola, tal como se muestra en las siguientes imágenes en donde ven las salidas de consola para los dos comandos que se mostraban en la guía:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 5: Salida de consola al ejecutar el comando docker-compose up airflow-init

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración 6: salida de consola al ejecutar el comando docker compose up

Entonces, cuando se accede al puerto 8080 del host local se puede ver la siguiente interfaz:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 7: Interfaz de Airflow de inicio

Más adelante, se configura la conexión a la base de datos PostgreSQL que se creó con anterioridad:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 8: Configuración de la conexión con la base de datos

En este caso, el usuario propietario de la base de datos quedó registrado con la contraseña “MESSIMESSIMESSI”, por lo que debe ser introducida para así poder hacer la prueba correspondiente a la base de datos. Además de esto, para poder realizar la prueba, el servidor pide autenticarse con el usuario administrador de ariflow:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 9: Autenticación en airflow para prueba de conexión

Con todo esto en cuenta, se puede probar la conexión a la base de datos con éxito, tal como se muestra en la siguiente imagen:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 10: Muestra de conexión correcta con la base de datos

Después de haber probado la conexión a la base de datos, se comienzan a crear los archivos mencionados en la guía. En primera instancia, la creación de las tablas se pone igual a lo que decía la guía, pero se agrega la siguiente sentencia:

Esto porque si se quisiera ejecutar el grafo de forma repetida, entonces los valores en las tablas generarían problemas de llave primaria, pues la misma fila de datos se agregaría más de una vez. Posteriormente, se agrega el archivo “file\_util.py”, utilizado para parametrizar la carga y el almacenamiento de los datos dependiendo de la dimensión que se quiera agregar. Finalmente, se agrega el script “insert\_queries.py”, en donde, para la inserción de los datos en la base, se crearon las sentencias, tal como se puede ver en los archivos anexos a la entrega, tal como lo pedía el requerimiento.

Ya habiendo agregado las sentencias, se agrega el archivo “ELT.py”, que representa el grafo usado para efectos de este laboratorio, y este también se puede ver en el repositorio. Con esto ya creado, se puede ver que el grafo ya sale desplegado en el servidor, tal como se ve en la siguiente imagen:

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 11: grafo de dependencias de las tareas

Después de ejecutar el árbol por primera vez, se pudo ver que hubo errores de pequeños caracteres y a la hora de conectar la base de datos, no obstante, fueron resueltos y documentados de forma correcta, como se puede ver en la siguiente imagen en la que se tienen 6 ejecuciones de forma seguida correctas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 12: muestra de la ejecución correcta de todas las tareas presentes en el grafo

**DOCUMENTACIÓN DE LOS ERRORES MÁS IMPORTANTES EN LA EJECUCIÓN**

El error que más interrumpió el correcto desarrollo de la actividad fue la conexión a la base de datos, pues hubo una confusión a la hora de manejar las credenciales del usuario propietario de la base de datos. No obstante, fue resuelto oportunamente.

**CONSULTAS SOBRE LA BASE DE DATOS**

A continuación, se muestran las consultas sobre las diferentes bases de datos desde PostgreSQL, para mostrar que las bases de datos quedaron bien cargadas:

* Dimensión ciudad:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración 13: consulta de la dimensión ciudad

* Dimensión cliente:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración 14: Consulta dimensión cliente

* Dimensión fecha:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 15: Consulta dimensión fecha

* Dimensión empleado:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración 16: Consulta dimensión empleado

* Tabla de hechos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 17: consulta sobre la tabla de hechos

* Dimensión ítem:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración 18: Consulta sobre la dimensión de ítem de compra

**PREGUNTAS**

1. Explique a fondo los siguientes conceptos de airflow: Task, Operator, DAG.
   * Task: cuando se está trabajando sobre airflow se están manejando pipelines sobre grafos acíclicos dirigidos. Recordando la definición de un pipeline, corresponde a una serie de tareas que se ejecutan para llegar de unos datos brutos a un resultado final; entonces, en este caso, se tiene la primera fase fuera del ETL que es la limpieza de datos que se hace mediante Python Notebooks desde Google. Más adelante, se tienen las diferentes partes que se ejecutan para ir transformando los datos de simplemente tablas a un datamart. Estos pasos son los mismos “tasks”, pues son las pequeñas tareas que se van ejecutando para tener la transformación, que en este caso son la creación de la base, de las tablas y posteriormente hacer las sentencias SQL para poblarlas.
   * Operator: tal como se puede ver en la documentación de Airflow, un operador es una plantilla para realizar cualquier tarea, para las cuales hay diferentes funcionalidades. La usada en este laboratorio fue la “PostgresOperator”, en la cual se hacen transacciones sobre una base de datos PostgreSQL. Otros ejemplos presentes en la documentación son “smooth”, “slack\_operator”, entre otros.
   * DAG: DAG, o “Directed Acyclic Graph” por sus siglas en inglés es un grafo sin ciclos no dirigido. Esta estructura de datos es utilizada para describir el proceso de dependencias entre las diferentes tareas que se realizan en el pipeline. Es muy importante usar esta estructura porque se deben tener en cuenta las dependencias entre las mismas, para evitar realizar tareas que todavía no tienen lo anterior, como no poblar las bases de datos sin haberlas creado. Este tipo de dependencias son ampliamente utilizadas en las cadenas de producción en las que los productos necesitan pasar por muchos procesos que tienen una secuencia lógica.
2. ¿Por qué se utiliza el comando “IF NOT EXISTS” en la sentencia de creación de tablas, en el contexto del proceso de ETL?

Porque el proceso ETL se puede utilizar para diferentes datos, entonces, se pueden disparar las tareas del grafo más de una vez. En caso de que se ejecuten más de una vez y la sentencia no tenga esas partes, va a haber un fallo porque va a intentar crear una base de datos ya existente, lo cual generará una excepción en el modelo SQL.

1. ¿Por qué para la columna de día se utiliza el nombre “day\_val” y no “day”?

Porque pueden generarse ambigüedades a la hora de trabajar con las consultas, dado es mucho más sencillo manejar consultas con los números a manejar los días de la semana. Además, es mucho más fácil manejar las consultas de los campos de fecha, tales como tener el total de ventas de los primeros 15 o últimos 10 días del mes. Si se tuviera un campo de texto con esta información, sería supremamente difícil diferenciar en qué época del mes se creó cada una de las ventas, lo cual es perder información valiosa para el negocio.

1. ¿De dónde se obtiene la información sobre las columnas que hay que crear en la tabla?

Para el grupo, la pregunta puede tener dos interpretaciones, la primera busca resolver de dónde se determina qué información irá almacenada en la base de datos, y la otra, respecto a dónde se encuentra la información en el desarrollo particular del laboratorio.

Para la primera interpretación, las columnas o atributos de las diferentes dimensiones salen del negocio, dependiendo de qué información sea relevante para la empresa o según los acuerdos de nivel de servicio. Por ejemplo, para un sistema de facturación se deben tener en cuenta dimensiones como la fecha, el lugar, el vendedor y el cliente para hacer un seguimiento detallado de los indicadores importantes para el negocio. Estos hechos se irán almacenando según la granularidad que defina el negocio, que para este caso es una factura de compra, de donde se puede obtener toda esta información. En caso de que no se cuente con toda la información relevante para el negocio, se debe replantear el sistema de recepción de información de las cajas de las tiendas, con el fin de tener información más detallada acerca de las transacciones que se realizan día a día.

Respecto al segundo enfoque, se puede obtener la información de las columnas mediante el script dado de crear tablas, en donde se puede ver claramente el tipo de datos esperado.

1. ¿Por qué es necesario un flujo de ejecución de las tareas en Airflow?

Por lo que se mencionaba anteriormente, hay un orden de jerarquía entre las tareas y una dependencia estricta que debe ser cumplida a cabalidad. Para efectos de este laboratorio, se debe primero conectar la base de datos, luego crear las tablas y finalmente poblarlas una a una para después poder llenar la tabla de hechos. En caso de que no se haga en ese orden se tendrán problemas en los pasos del proceso.

**DIAGRAMA DE SOLUCIÓN ETL:**

Diagram

Description automatically generated