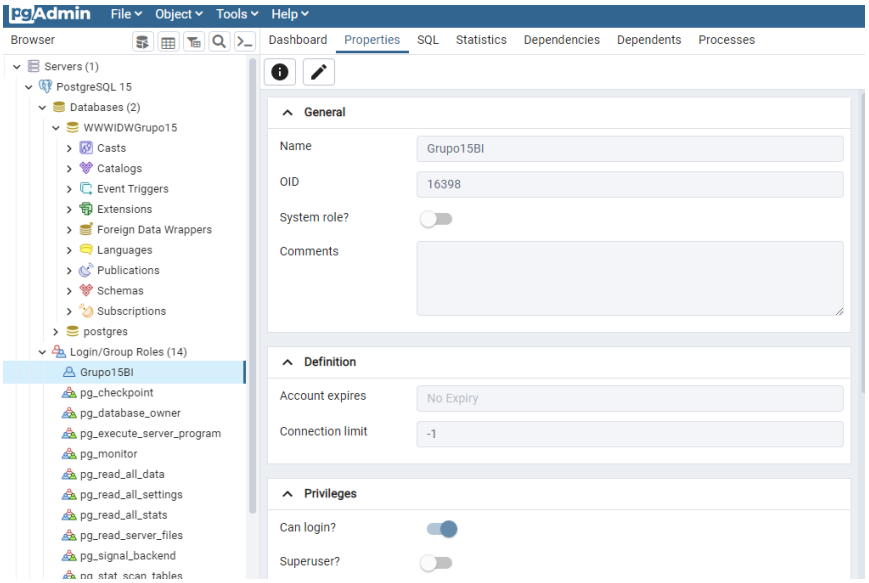
Laboratorio 5 – BI

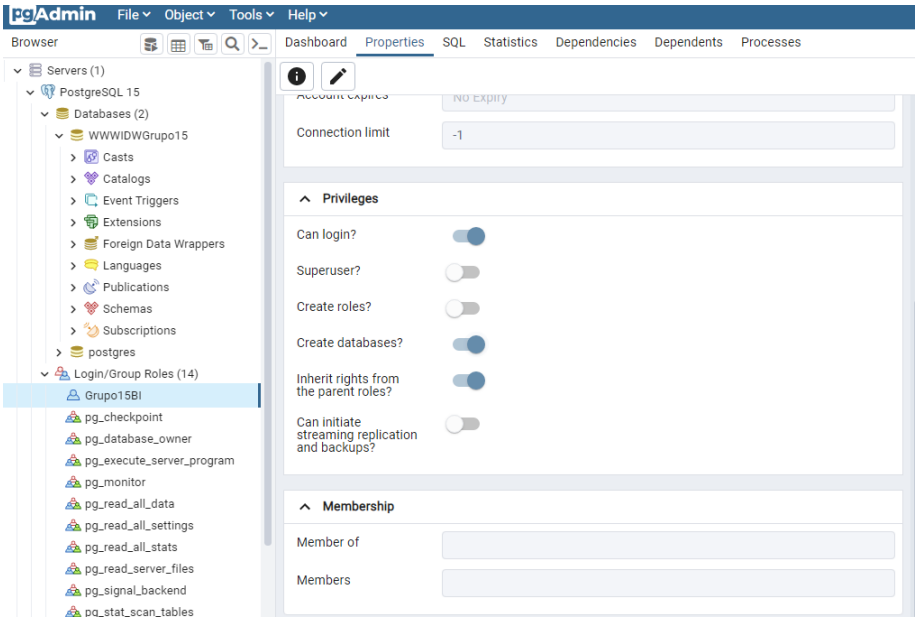
María Paula González Escallón
Jessica A. Robles Moreno
Juan Esteban Vergara

Descripción del proceso

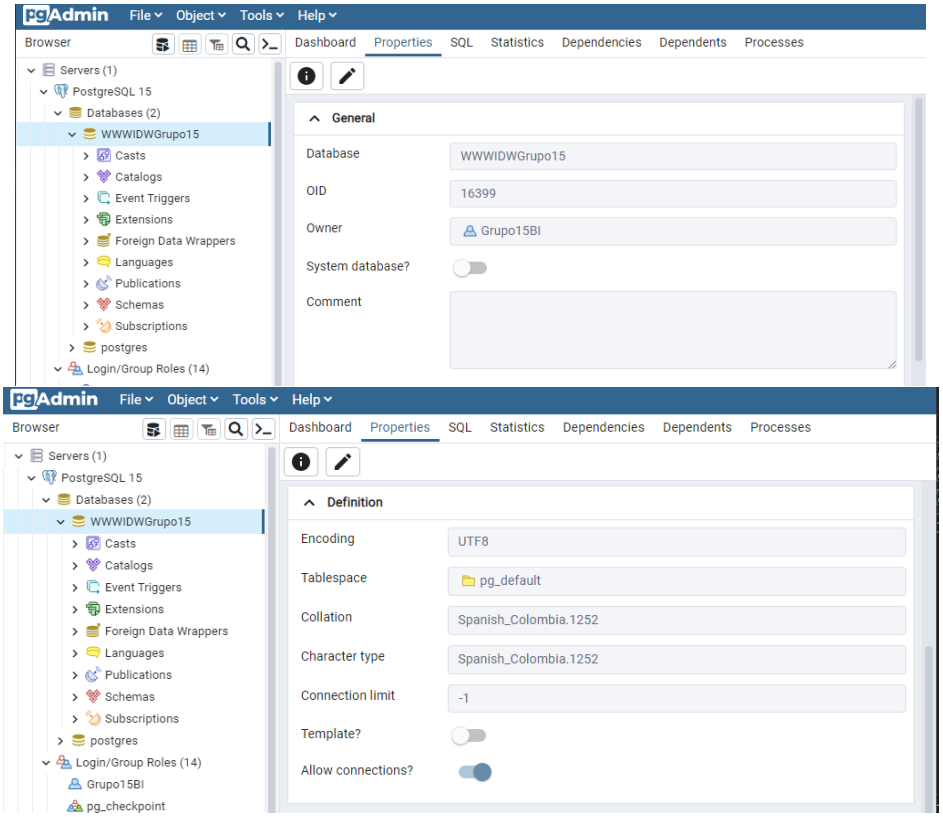
1. Creación de la base de datos

Para la creación de la base de datos se tuvo que abrir la aplicación de pgAdmin4. Seguido a esto, se creó el usuario, en nuestro caso el nombre es Grupo15BI y se dieron los permisos indicados.





Ya teniendo el usuario creado se creó la base de datos llamada “WWWIDWGrupo15” y se le asignó como dueño de esta base de datos el usuario anteriormente creado



2. Ejecutar Airflow entender su funcionamiento

Para poder ejecutar Airflow fue necesario comenzar la ejecución de Docker Desktop, al iniciar esta aplicación comienza la ejecución de Airflow mediante un contenedor de Docker, y para ir a la interfaz fue necesario dirigirnos a la dirección http://localhost:8080/. Para encontrar los archivos con los que trabaja esta aplicación fue necesario dirigirnos a la carpeta "Airflow-Docker" que se encuentra ubicada en los documentos de la máquina virtual asignada. Se revisó varias de las carpetas, las cuales ya se encontraban vacías debido a que aún no se habían hecho los archivos relacionados con la ejecución de los dags.

3. Perfilamiento de datos

Se revisaron los datos proporcionados por la empresa para el laboratorio. Entre estos nos podemos encontrar con seis archivos csv. Para poder revisar mejor su contenido y hacer una mejor limpieza de estos decidimos analizarlos desde dataframes, y así poder utilizar funciones de análisis que ofrece la librería pandas y poder mantener un registro de los cambios hechos y las visualizaciones de estos.

El primer archivo encontrado es fact_order.csv, este tiene un total de 1000 registros y 14 columnas. Este archivo contiene la información sobre la tabla de hechos principal, por ende, tiene referencias a llaves de otras dimensiones y tiene métricas sobre impuestos y otras informaciones de los paquetes. Se encontró que esta tabla no tiene registros nulos ni se encontraron inconsistencias en los datos, es por esto que no se le realizó ninguna limpieza.

El segundo archivo encontrado es dimension_city.csv, este tiene un total de 98 registros y 9 columnas. Este archivo contiene la información sobre la dimensión de la ciudad, por ende, tiene información sobre la ciudad y la región relacionadas a la orden. Se encontró que algunas columnas contenían registros nulos (6 de las 9 columnas tenían un único nulo) y no se encontraron inconsistencias en los datos, es por esto que solo fue necesario eliminar los registros que contuvieran nulos ya que estos no representan una gran cantidad sobre los registros totales. Después de hacer esto, el archivo quedó con un total de 97 registros.

El siguiente archivo encontrado es dimension_customer.csv, este tiene un total de 403 registros y 7 columnas. Este archivo contiene la información sobre la dimensión de los clientes, por ende, tiene información sobre el pago, él envío y la categoría de cada cliente. Se encontró que algunas columnas contenían registros nulos (5 de sus 7 columnas tienen un único nulo) y se encontraron inconsistencias entre los datos. Para esto fue necesario eliminar los registros que contuvieran nulos ya que estos no representan una gran cantidad sobre los registros totales. Después de hacer esto, el archivo quedó con un total de 402 registros. También se evidenció que hay datos con apostrofes que pueden ser un problema cuando se quieran ingresar en queries a través de sql, para manejar esto se eliminaron todas las apostrofes en el dataframe.

El cuarto archivo encontrado es dimension_date.csv, este tiene un total de 1461 registros y 8 columnas. Este archivo contiene la información sobre la dimensión de las fechas, por ende, tiene información sobre el día, mes y año relacionados a la orden. Se encontró que no había columnas que contuvieran registros nulos y no se encontraron inconsistencias entre los datos, es por esto que no fue necesario eliminar o modificar los registros.

El siguiente archivo encontrado es dimension_employee.csv, este tiene un total de 213 registros y 4 columnas. Este archivo contiene la información sobre la dimensión de los empleados. Se encontró que una columna contenía un registro nulo y no se encontraron inconsistencias entre los datos, es por esto que solo fue necesario eliminar los registros que contuvieran nulos ya que estos no representan una gran cantidad sobre los registros totales. Después de hacer esto, el archivo quedó con un total de 212 registros.

El sexto archivo encontrado es dimension_stock_item.csv, este tiene un total de 672 registros y 14 columnas. Este archivo contiene la información sobre la dimensión de los ítems que contiene las cantidades y características de estos. Se encontró que habían varias columnas con registros nulos, pero debido a que eran una gran cantidad de datos con valores nulos, especialmente en la columna Brand, la cual solo tenía 67 registros no nulos, y como no se podían remplazar estos valores con algún valor, se asignó este valor como unknown. También se encontró que habían ciertos registros que contenían apostrofes, y estos generan problemas al momento de ingresarlos en la base de datos a través de queries, es por esto que también fue necesario eliminarlas.

1. Implementación ETL

Para comenzar este proceso fue necesario crear una conexión en Airflow. En esta conexión se hace referencia a la base de datos anteriormente creada en postgres para que se puedan acceder a estos datos.

Acto seguido se comenzó la implementación de los dags. Para esto fue necesario crear 4 archivos python en la carpeta dags dentro de la carpeta "Airflow-Docker" mencionada anteriormente. Se creo una carpeta utils en la cual se crearon 3 archivos:

1. crear_tablas.py: para el contenido de este archivo solo fue necesario copiar la información con las sentencias utilizadas para la creación de las tablas en la base de datos de postgres, no fue necesario hacerle ninguna modificación.
2. file_util.py: este archivo es el encargado del manejo de los datos, la descarga, limpieza y carga de estos. Para crearlo fue necesario copiar el código que ya nos proporcionaba el tutorial que incluía las funciones para hacer la carga y descarga de los archivos csv. De esta parte solo fue necesario cambiar la dirección del link de descarga de los archivos con el número de nuestro grupo. Sin embargo, en el código proporcionado aún no se había hecho la limpieza de los datos, fue necesario que nosotros la desarrolláramos. Para hacer la limpieza se hizo un procedimiento similar al que se hizo en el notebook utilizado para el perfilamiento de datos, se cargaron los csv a dataframes y desde aquí se hicieron las modificaciones indicadas anteriormente.
3. insert_queries.py: este archivo es encargado de crear las sentencias utilizadas para la carga de los datos del csv a la base de datos. El código proporcionado en el enunciado solo contenía la

función para poblar la tabla de las ciudades, nosotros tuvimos que implementar el resto de las funciones. Para hacer esto hicimos las funciones teniendo en cuenta los parámetros con los cuales se crearon las tablas, su tipo y el nombre de la columna del csv. Las sentencias fueron hechas de manera similar a como se realizó en la función dada como ejemplo, lo único que fue necesario tener en cuenta fue que el tipo de columna fuera ingresado de tal manera que pueda ser ingresado con éxito a la base de datos y que este concuerde con el tipo de dato indicado al momento de crear las tablas. Es por esto que para ingresar las fechas fue necesario pasarlas de tipo string a tipo fecha con ayuda de la función TO_DATE en sql y fue necesario ingresar los números decimales con separadores de "." En vez de ",". El resto de los datos no fue necesario hacerles modificaciones.

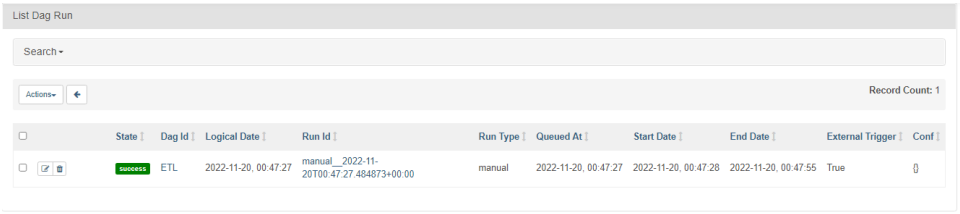
Ya para la implementación del DAG se creó un archivo en la carpeta dags llamado ETL.py, e.e archivo es el encargado de manejar el proceso del dag. Se encarga de crear un orden de ejecución y de hacer llamado a las funciones implementadas en los archivos dentro de la carpeta utils para el manejo de los datos, creación de la base de datos y poblar todas las tablas. Este archivo fue suministrado en su totalidad en el enunciado y no fue necesario hacer ningún cambio.

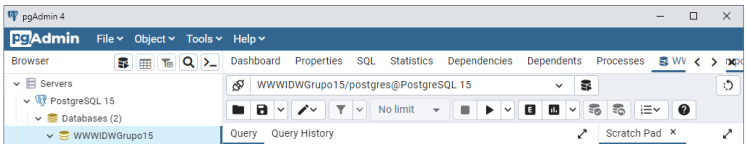
Después de tener todos los archivos necesarios para la ejecución del DAG se ejecutó este en la interfaz de Airflow. La primera vez que se ejecutó aún no se habían hecho todas las modificaciones a las sentencias para que los datos ingresados correspondieran a el tipo de dato de la columna de la tabla en la base de datos, es por esto que al principio la ejecución no fue exitosa. Nos dimos cuenta de nuestros errores a través de los logs generados por Airflow e hicimos las modificaciones necesarias. Cuando se volvió a ejecutar el proceso se generó un nuevo error debido a que no se eliminaron los registros en las bases de datos, y los registros que se estaban cargando contenían llaves primarias ya utilizadas, y por ende, no podían ser cargados. Para solucionar este problema se eliminaron los datos a través de sentencias sql en la base de datos (este archivo se encuentra adjunto en la entrega). Ya cuando se solucionó este problema se pudo ejecutar el dag de manera exitosa.

Resultados

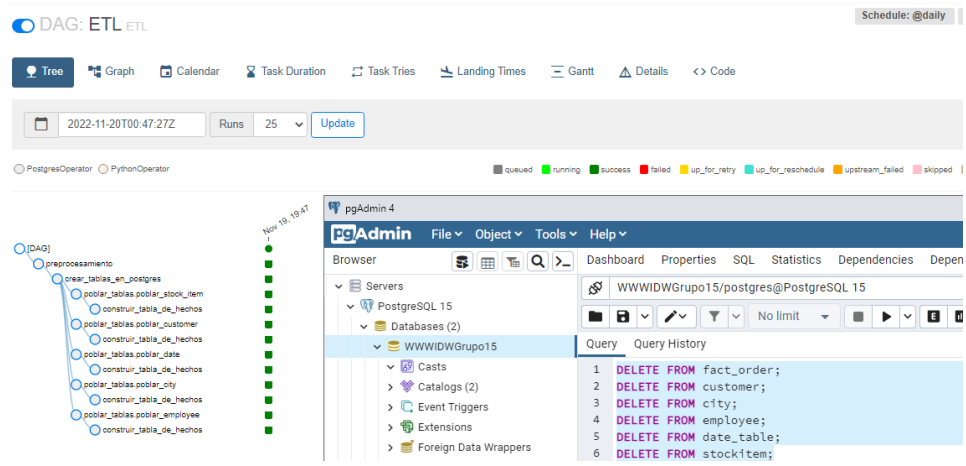
Finalmente, después de hacer el procedimiento indicado en el tutorial del enunciado y de realizar las correcciones se pudieron observar los siguientes resultados demostrando la correcta ejecución del dag.

1. Estado exitoso de la ejecución:





2. Árbol de ejecución exitoso:



3. Grafo de ejecución exitoso:

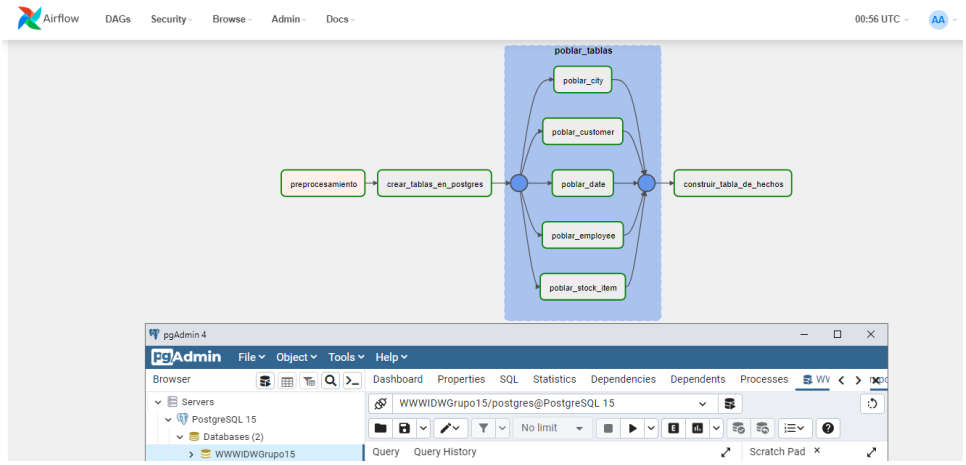
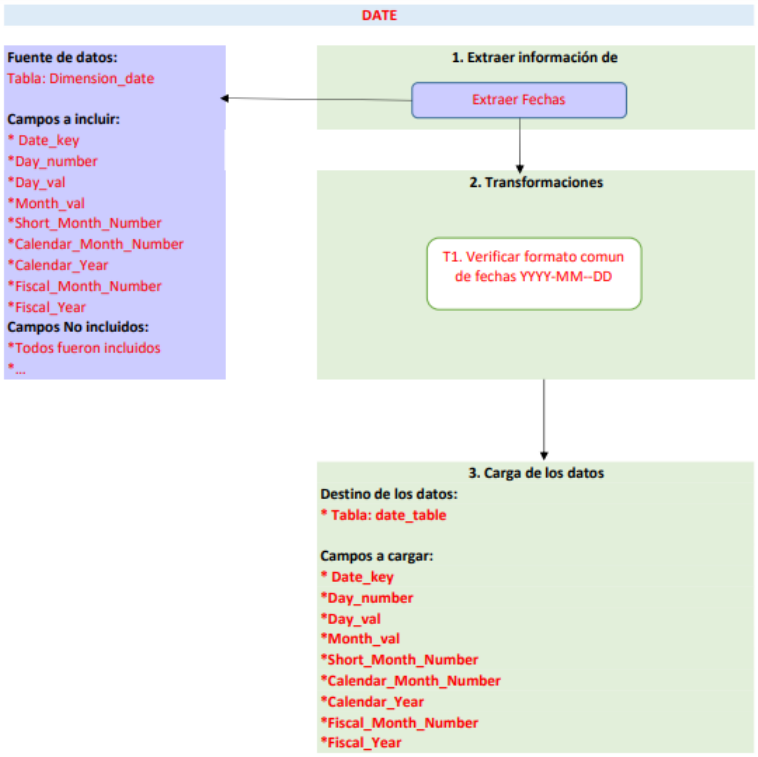
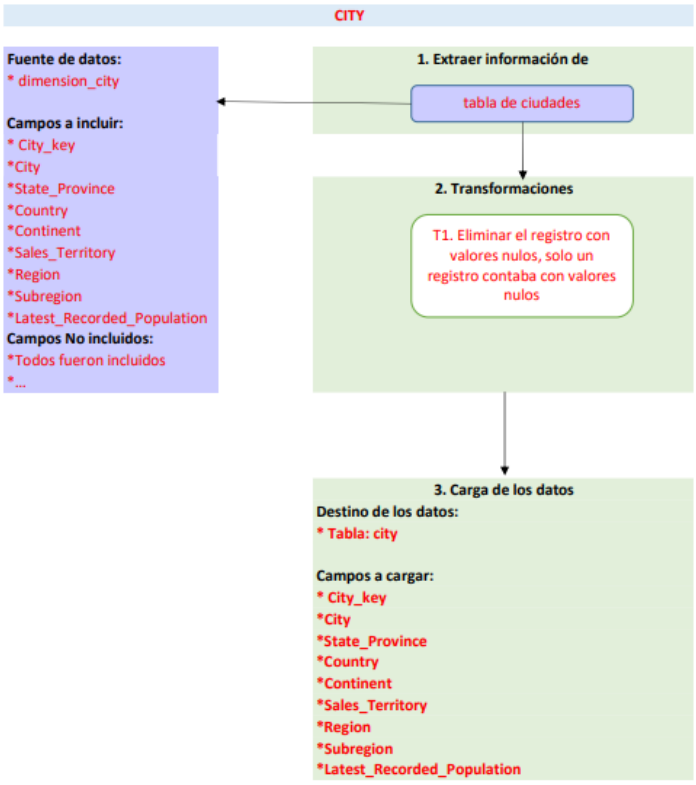


Diagrama de alto nivel

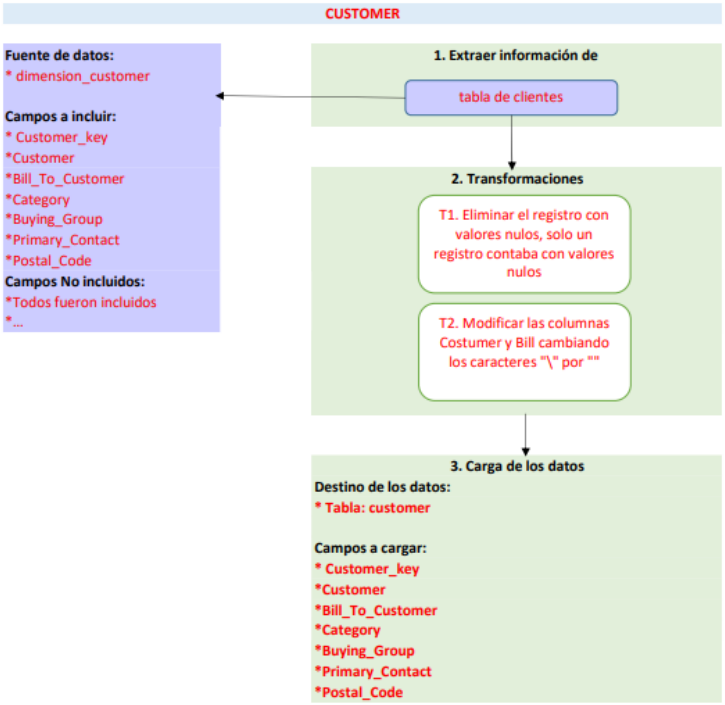
1. Diagrama de alto nivel de fechas:



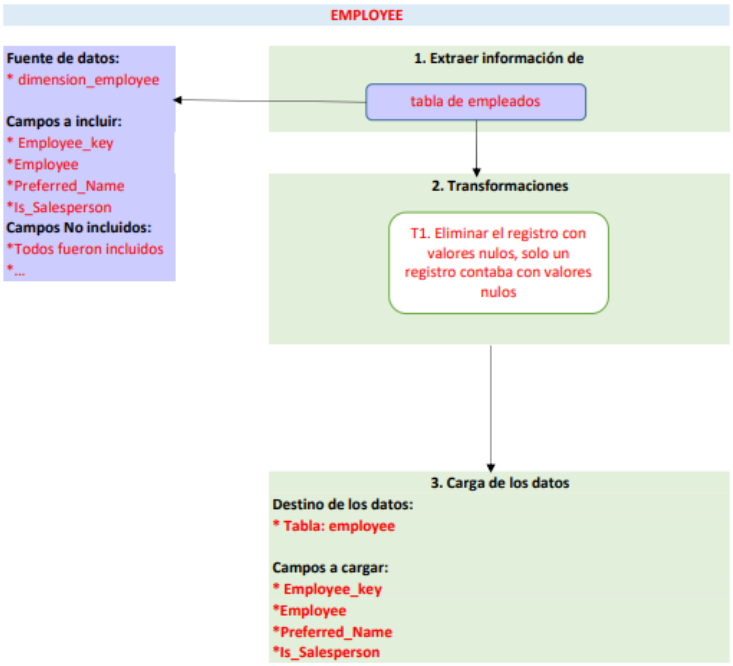
2. Diagrama de alto nivel de ciudades:



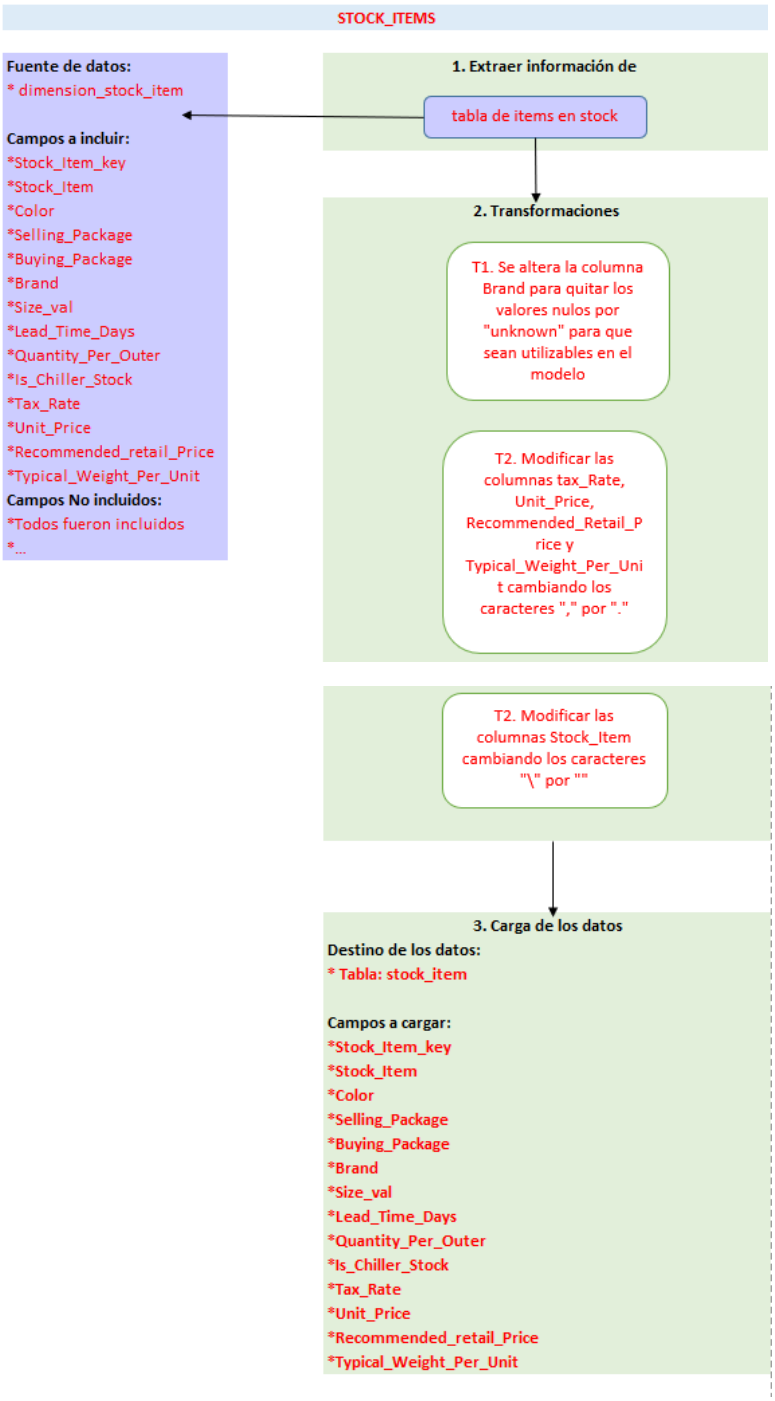
3. Diagrama de alto nivel de customer:



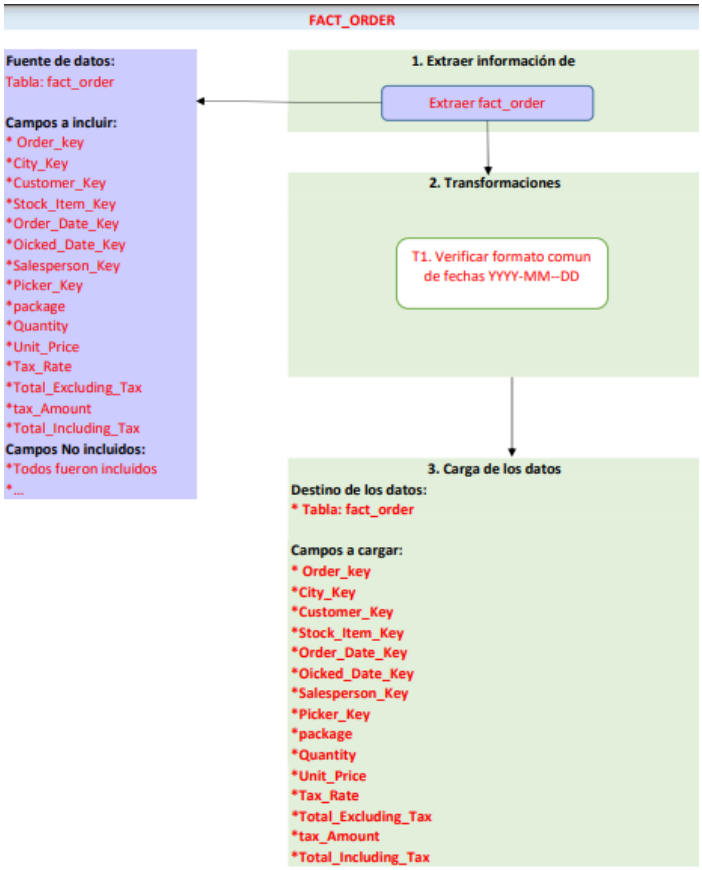
4. Diagrama de alto nivel de empleados:



5. Diagrama de alto nivel de items:



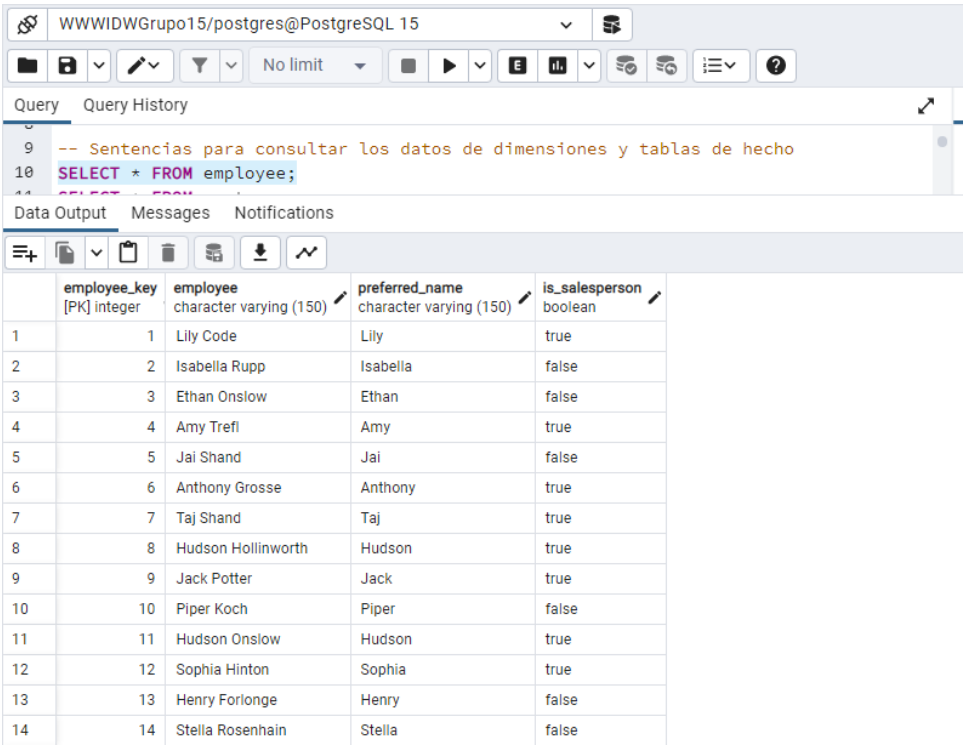
6. Diagrama de alto nivel de ordenes:



Consultas

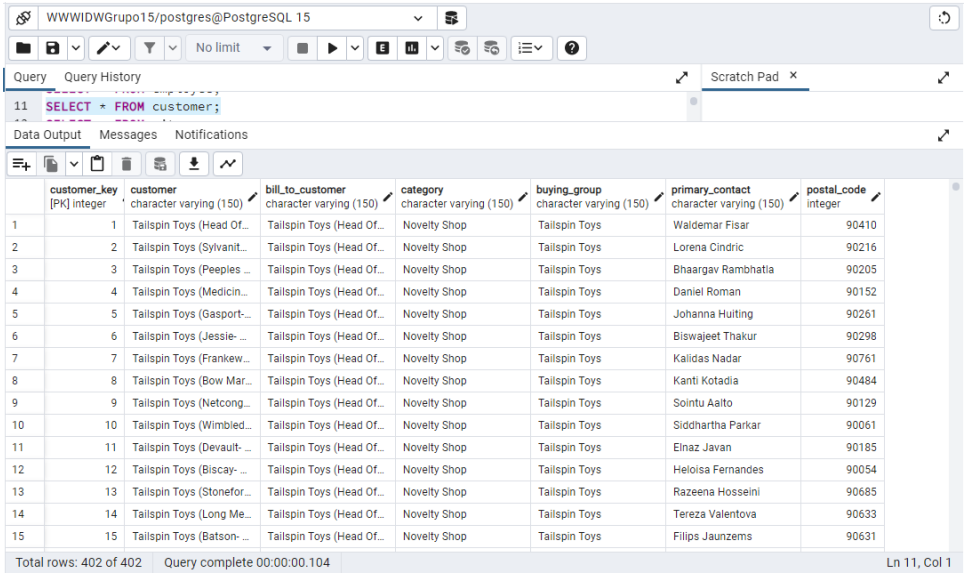
Para verificar que las tablas se poblaran de manera correcta se ejecutaron las siguientes sentencias en la base de datos después de la ejecución del dag. Los resultados de estas consultas se encuentran adjuntados en la entrega para una mejor visualización.

1. Visualización de la tabla empleados:



	employee_key [PK] integer	employee character varying (150)	preferred_name character varying (150)	is_salesperson boolean
1	1	Lily Code	Lily	true
2	2	Isabella Rupp	Isabella	false
3	3	Ethan Onslow	Ethan	false
4	4	Amy Trefl	Amy	true
5	5	Jai Shand	Jai	false
6	6	Anthony Grosse	Anthony	true
7	7	Taj Shand	Taj	true
8	8	Hudson Hollinworth	Hudson	true
9	9	Jack Potter	Jack	true
10	10	Piper Koch	Piper	false
11	11	Hudson Onslow	Hudson	true
12	12	Sophia Hinton	Sophia	true
13	13	Henry Forlonge	Henry	false
14	14	Stella Rosenhain	Stella	false

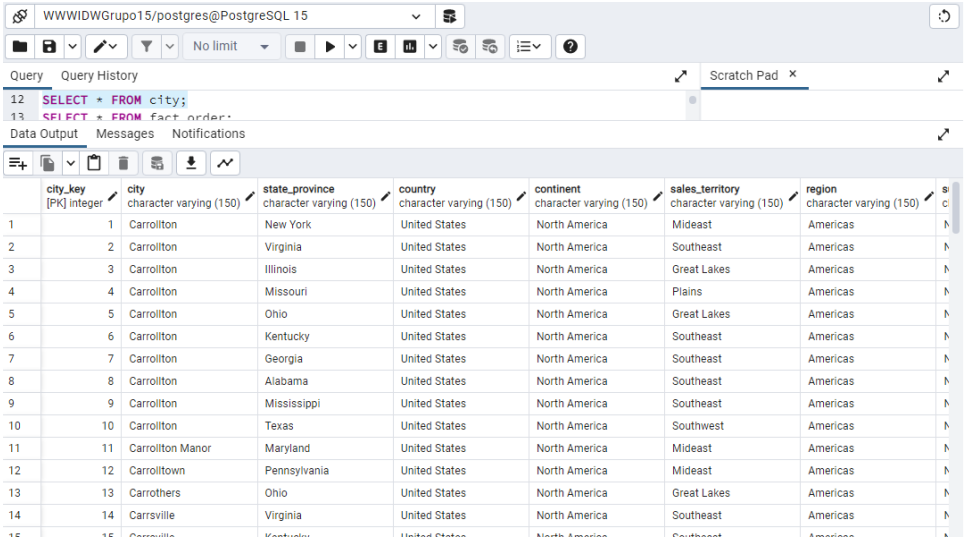
2. Visualización de la tabla customer:



	customer_key [PK] integer	customer character varying (150)	bill_to_customer character varying (150)	category character varying (150)	buying_group character varying (150)	primary_contact character varying (150)	postal_code integer
1	1	Tailspin Toys (Head Of...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Waldemar Fisar	90410
2	2	Tailspin Toys (Sylvanit...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Lorena Cindric	90216
3	3	Tailspin Toys (Peeples ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Bhaargav Rambhatla	90205
4	4	Tailspin Toys (Medicin...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Daniel Roman	90152
5	5	Tailspin Toys (Gasport-...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Johanna Huiting	90261
6	6	Tailspin Toys (Jessie- ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Biswajeet Thakur	90298
7	7	Tailspin Toys (Frankew...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Kalidas Nadar	90761
8	8	Tailspin Toys (Bow Mar...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Kanti Kotadia	90484
9	9	Tailspin Toys (Netcong...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Sointu Aalto	90129
10	10	Tailspin Toys (Wimbled...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Siddhartha Parkar	90061
11	11	Tailspin Toys (Devault- ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Elnaz Javan	90185
12	12	Tailspin Toys (Biscay- ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Heloisa Fernandes	90054
13	13	Tailspin Toys (Stonefor...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Razeena Hosseini	90685
14	14	Tailspin Toys (Long Me...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Tereza Valentova	90633
15	15	Tailspin Toys (Batson- ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Filips Jaunzems	90631

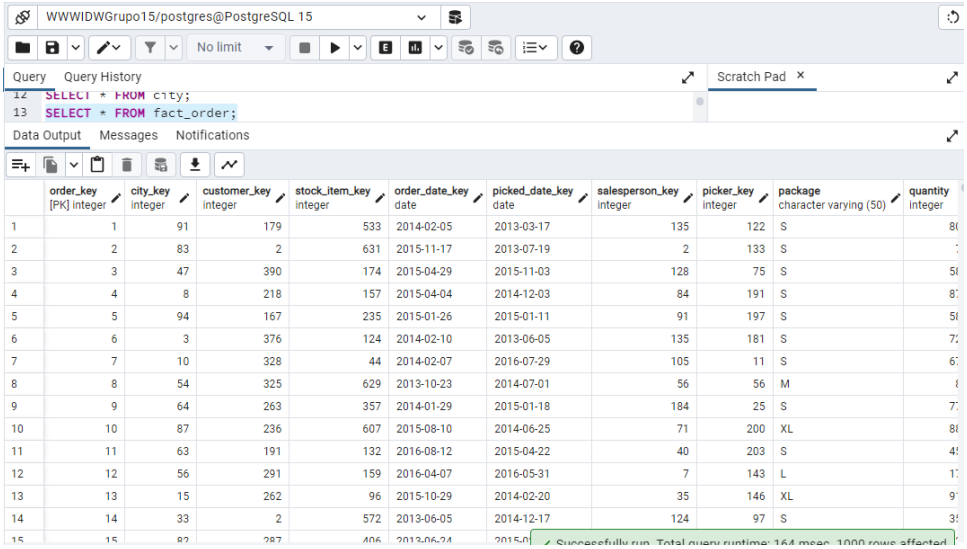
Total rows: 402 of 402 Query complete 00:00:00.104 Ln 11, Col 1

3. Visualización de la tabla de ciudades:



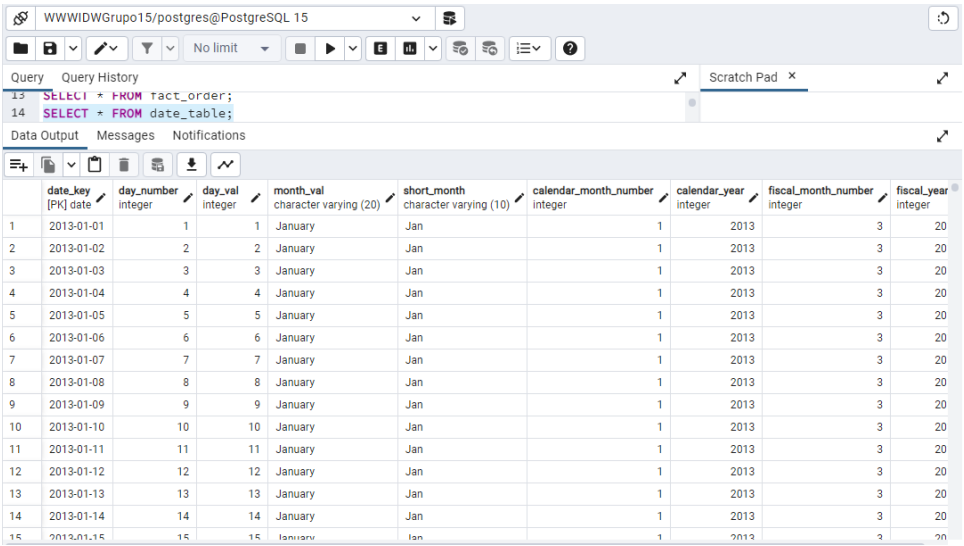
	city_key [PK] integer	city character varying (150)	state_province character varying (150)	country character varying (150)	continent character varying (150)	sales_territory character varying (150)	region character varying (150)
1	1	Carrollton	New York	United States	North America	Mideast	Americas
2	2	Carrollton	Virginia	United States	North America	Southeast	Americas
3	3	Carrollton	Illinois	United States	North America	Great Lakes	Americas
4	4	Carrollton	Missouri	United States	North America	Plains	Americas
5	5	Carrollton	Ohio	United States	North America	Great Lakes	Americas
6	6	Carrollton	Kentucky	United States	North America	Southeast	Americas
7	7	Carrollton	Georgia	United States	North America	Southeast	Americas
8	8	Carrollton	Alabama	United States	North America	Southeast	Americas
9	9	Carrollton	Mississippi	United States	North America	Southeast	Americas
10	10	Carrollton	Texas	United States	North America	Southwest	Americas
11	11	Carrollton Manor	Maryland	United States	North America	Mideast	Americas
12	12	Carrolltown	Pennsylvania	United States	North America	Mideast	Americas
13	13	Carrothers	Ohio	United States	North America	Great Lakes	Americas
14	14	Carrsville	Virginia	United States	North America	Southeast	Americas

4. Visualización de la tabla de hechos:



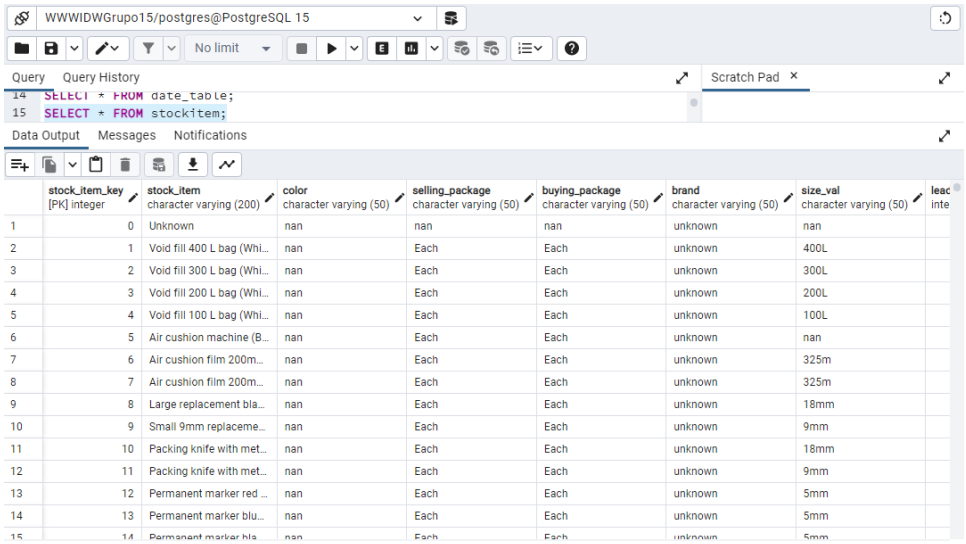
	order_key [PK] integer	city_key integer	customer_key integer	stock_item_key integer	order_date_key date	picked_date_key date	salesperson_key integer	picker_key integer	package character varying (50)	quantity integer
1	1	91	179	533	2014-02-05	2013-03-17	135	122	S	8
2	2	83	2	631	2015-11-17	2013-07-19	2	133	S	1
3	3	47	390	174	2015-04-29	2015-11-03	128	75	S	5
4	4	8	218	157	2015-04-04	2014-12-03	84	191	S	8
5	5	94	167	235	2015-01-26	2015-01-11	91	197	S	5
6	6	3	376	124	2014-02-10	2013-06-05	135	181	S	7
7	7	10	328	44	2014-02-07	2016-07-29	105	11	S	6
8	8	54	325	629	2013-10-23	2014-07-01	56	56	M	
9	9	64	263	357	2014-01-29	2015-01-18	184	25	S	7
10	10	87	236	607	2015-08-10	2014-06-25	71	200	XL	8
11	11	63	191	132	2016-08-12	2015-04-22	40	203	S	4
12	12	56	291	159	2016-04-07	2016-05-31	7	143	L	1
13	13	15	262	96	2015-10-29	2014-02-20	35	146	XL	9
14	14	33	2	572	2013-06-05	2014-12-17	124	97	S	3

5. Visualización de la tabla de fechas:



	date_key [PK] date	day_number integer	day_val integer	month_val character varying (20)	short_month character varying (10)	calendar_month_number integer	calendar_year integer	fiscal_month_number integer	fiscal_year integer
1	2013-01-01	1	1	January	Jan	1	2013	3	20
2	2013-01-02	2	2	January	Jan	1	2013	3	20
3	2013-01-03	3	3	January	Jan	1	2013	3	20
4	2013-01-04	4	4	January	Jan	1	2013	3	20
5	2013-01-05	5	5	January	Jan	1	2013	3	20
6	2013-01-06	6	6	January	Jan	1	2013	3	20
7	2013-01-07	7	7	January	Jan	1	2013	3	20
8	2013-01-08	8	8	January	Jan	1	2013	3	20
9	2013-01-09	9	9	January	Jan	1	2013	3	20
10	2013-01-10	10	10	January	Jan	1	2013	3	20
11	2013-01-11	11	11	January	Jan	1	2013	3	20
12	2013-01-12	12	12	January	Jan	1	2013	3	20
13	2013-01-13	13	13	January	Jan	1	2013	3	20
14	2013-01-14	14	14	January	Jan	1	2013	3	20

6. Visualización de la tabla items:



	stock_item_key [PK] integer	stock_item character varying (200)	color character varying (50)	selling_package character varying (50)	buying_package character varying (50)	brand character varying (50)	size_val character varying (50)	leac inte
1	0	Unknown	nan	nan	nan	unknown	nan	
2	1	Void fill 400 L bag (Whi...	nan	Each	Each	unknown	400L	
3	2	Void fill 300 L bag (Whi...	nan	Each	Each	unknown	300L	
4	3	Void fill 200 L bag (Whi...	nan	Each	Each	unknown	200L	
5	4	Void fill 100 L bag (Whi...	nan	Each	Each	unknown	100L	
6	5	Air cushion machine (B...	nan	Each	Each	unknown	nan	
7	6	Air cushion film 200m...	nan	Each	Each	unknown	325m	
8	7	Air cushion film 200m...	nan	Each	Each	unknown	325m	
9	8	Large replacement bla...	nan	Each	Each	unknown	18mm	
10	9	Small 9mm replaceme...	nan	Each	Each	unknown	9mm	
11	10	Packing knife with met...	nan	Each	Each	unknown	18mm	
12	11	Packing knife with met...	nan	Each	Each	unknown	9mm	
13	12	Permanent marker red ...	nan	Each	Each	unknown	5mm	
14	13	Permanent marker blu...	nan	Each	Each	unknown	5mm	
15	14	Permanent marker bla...	nan	Each	Each	unknown	5mm	

Preguntas

1. Explique a fondo los siguientes conceptos de airflow: Task, Operator, DAG.
   - Task

     En Airflow, una tarea es la unidad básica de ejecución. Las tareas son organizadas en DAG's. Existen 3 tipos básicos de tareas: Operators, Sensors y TaskFlow-decorated. La clave para usar las tareas es definir como se relacionan entre ellas. De forma predeterminada, una tarea se ejecutará después de que sus tareas antecesoras se hayan ejecutado exitosamente.(Tasks — Airflow Documentation, s. f.)

   - Operator

     Los operadores son plantillas de tareas predefinidas. Además, determinan que es lo que se ejecuta en una tarea. (Operators — Airflow Documentation, s. f.)(Using Operators — Airflow Documentation, s. f.)

   - DAG

     Es un gráfico acíclico dirigido, es decir, un grafo dirigido sin ciclos. En Airflow es el concepto principal, el cual recopila tareas para manifestar en que orden deben ejecutarse, así como decidir que tareas dependen de otras. (DAGs — Airflow Documentation, s. f.)

2. ¿Por qué para la columna de día se utiliza el nombre "day_val" y no "day"?

   Se utiliza este nombre para no generar confusiones. Primero para hacer referencia que el día va a ser un valor, un entero. Segundo para que se sepa que este no hace referencia a una dimensión de la fecha, sino que hace referencia a un valor. De esta manera es menos probable que se generen confusiones al momento de visualizar los datos o la estructura de la base de datos

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

   Para saber las columnas que van a ser utilizadas en las filas es necesario ver la información que se tiene. Para el caso del proyecto es necesario ver las columnas que existen en cada archivo csv, y al momento de decidir que columnas se van a usar es

necesario evaluar cuales de las columnas proporcionadas son importantes para el negocio y si es necesario agregar más. Sin embargo, en nuestro caso la información de las columnas que van a ser usadas en la base de datos ya eran proporcionadas a través del archivo crear_tablas.py. En este archivo se encontraban las sentencias para la creación de las tablas en la base de datos y sus respectivas columnas.

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

   El flujo de ejecución es necesario para mantener el orden de ejecución de las tareas. Además, esto evitará que se ejecuten tareas, las cuales sus antecesoras hayan salido sin éxito, evitando errores de ejecución que pueden ser altamente perjudiciales.

5. ¿Qué ajustes habría que hacer a este proceso de ETL si se trata de un ETL incremental, donde previamente hay datos cargados en la bodega de datos?

   Si se trata de un ETL incremental en donde ya existen datos previamente cargados, lo que se debe hacer es comparar los datos que ya están en la base de datos con los datos que queremos introducir, de esta manera se asegura que solo se cargarán los datos que no se encuentran en la base de datos. Al tener menos datos que agregar se reducen los tiempos de ejecución, se reduce el riesgo de fallos en la carga, el rendimiento se vuelve consistente y se mantienen los datos históricos.

   Existen 2 métodos para la implementación de un ETL incremental, estos métodos son:

   - Comparación del cambio destino: Este método compara, por medio de las filas, los registros nuevos con los datos que no sufrieron cambios. Requiere traer todos los datos para monitorear los cambios. Sin embargo, tiene menos suposiciones que el método de comparación del cambio de fuente.
   - Comparación del cambio de fuente: Este método extrae únicamente los nuevos datos de donde se sacará la información.

   (Agarwal, 2022)

6. Al revisar lo entregado por el grupo previo de consultores, se evidencia que no se está trabajando de forma apropiada con las llaves primarias, ya que se tienen las del sistema transaccional como PK, es así como se decide mantener esas llaves y renombrarlas con el sufijo _T y crear las propias de la bodega de datos con el sufijo _DWH. Estas últimas son consecutivos. ¿Qué se debe hacer para que este cambio sea efectivo? muestre un ejemplo para una dimensión y su efecto en la tabla de hechos.

   Si se quiere hacer este cambio es necesario modificar la base de datos para que las tablas reconozcan que se tiene una nueva llave primaria y hacer las modificaciones correspondientes a las llaves anteriores para cambiarles el nombre y que estas no sean primarias en su respectiva tabla. También es necesario hacer modificaciones a otras dimensiones que hagan referencia a la llave primaria y asignarles esta nueva referencia. Acto seguido se debe modificar las consultas y las sentencias para el ingreso de datos con los cambios definidos. Por ejemplo: para la dimensión de empleados a tabla tendria los valores de Employee_DWH como primary key, Employee_T, employee, preferred_name y is_salesperson.

7. ¿A nivel de la dimensión Fecha, se detecta que la llave primaria no es un entero que represente la fecha (AAAAMMDD), sino un atributo de tipo DATE, cómo corregirían este error?

   Nos damos cuenta de que el valor de la llave primaria de la dimensión fecha es de tipo DATE en vez de tipo string al momento de ingresarla a la base de datos, ya que en esta se definió que el valor de la llave primaria es de tipo DATE. Para solucionar esto es necesario pasar este dato a un atributo DATE mediante la instrucción TO_DATE en las sentencias en sql.

Referencias

DAGs — Airflow Documentation. (s. f.). https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html

Tasks — Airflow Documentation. (s. f.). https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html

Operators — Airflow Documentation. (s. f.). https://airflow.apache.org/docs/apache-airflow/stable/concepts/operators.html

Using Operators — Airflow Documentation. (s. f.). https://airflow.apache.org/docs/apache-airflow/stable/howto/operator/index.html

Agarwal, S. (2022, 4 julio). *ETL Incremental Loading 101: A Comprehensive Guide*. Learn | Hevo. https://hevodata.com/learn/etl-incremental/