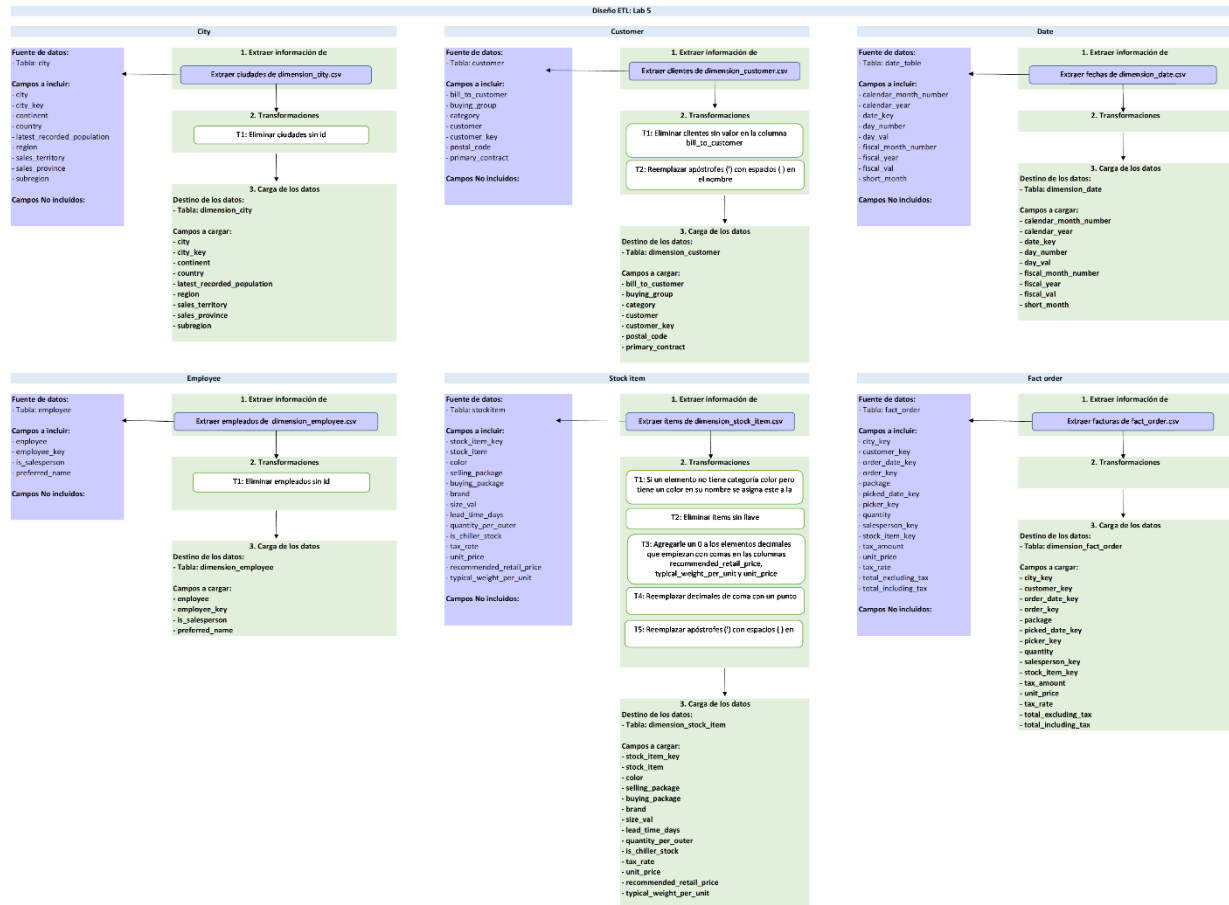


LAB 5 – Grupo 28

Enlace Repositorio en GitHub: [Lab5](#)

Enlace Excel con plantilla de diseño: [Excel](#)

Plantilla de Diseño ETL



Limpieza de datos

- **Preprocesamiento dimension_city.csv:** Al revisar el archivo pudimos ver que existe una fila completa de nulos, así que decidimos eliminarla pues no aporta nada al modelo ni al negocio. Se eliminó la columna de los IDs de las filas.
- **Preprocesamiento dimension_customer.csv:** Al revisar el archivo pudimos ver que existe una fila completa de nulos, así que decidimos eliminarla pues no aporta nada al modelo ni al negocio. Se quitó el carácter ' para evitar errores en la base de datos.
- **Preprocesamiento dimension_date.csv:** No hubo necesidad de hacer transformación
- **Preprocesamiento dimension_employee.csv:** Se eliminaron las filas que contenían nulos.
- **Preprocesamiento dimension_stock_item.csv:** Existen algunos elementos que tienen un color en el nombre pero en el campo de Color hay nulos, entonces para aquellos que tenían color en el nombre se llenó el campo de color con el color que estaba en el nombre. Se eliminó la fila que solo tenía nulos. Existían numeros que no estaban completos sino que comenzaban con , esto hacía conflicto con las queries de inserción así que decidimos añadirles un 0 al principio y cambiar la , por el .

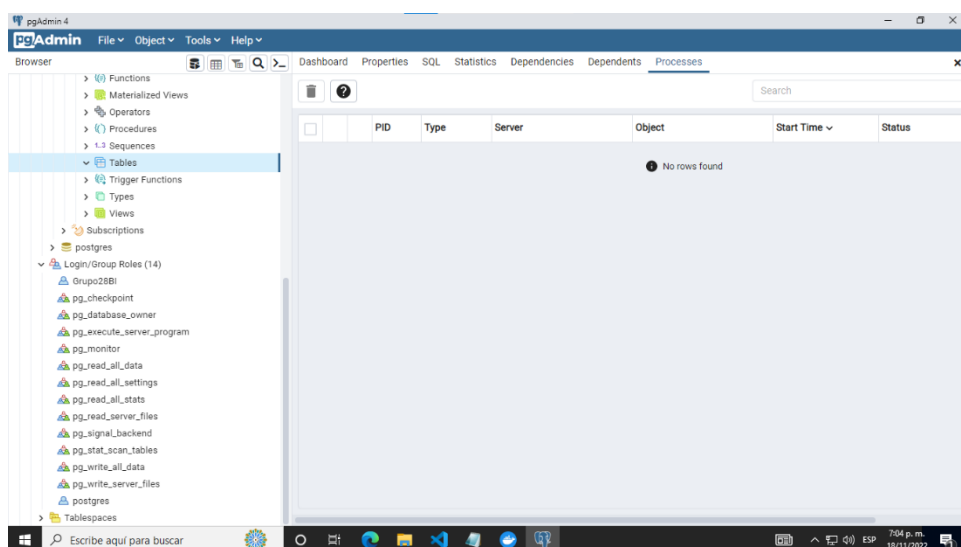
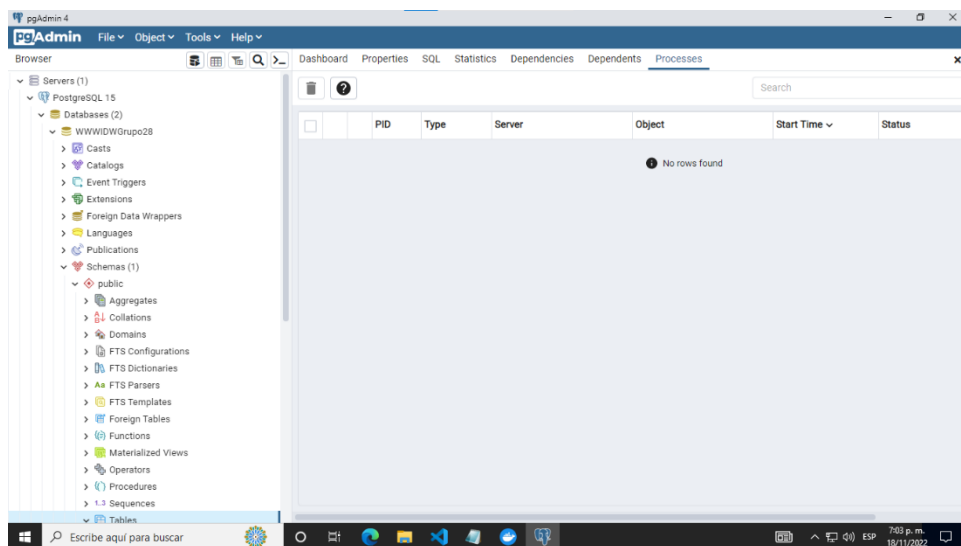
- **Preprocesamiento fact_order.csv:** No fue necesaria ninguna transformación.

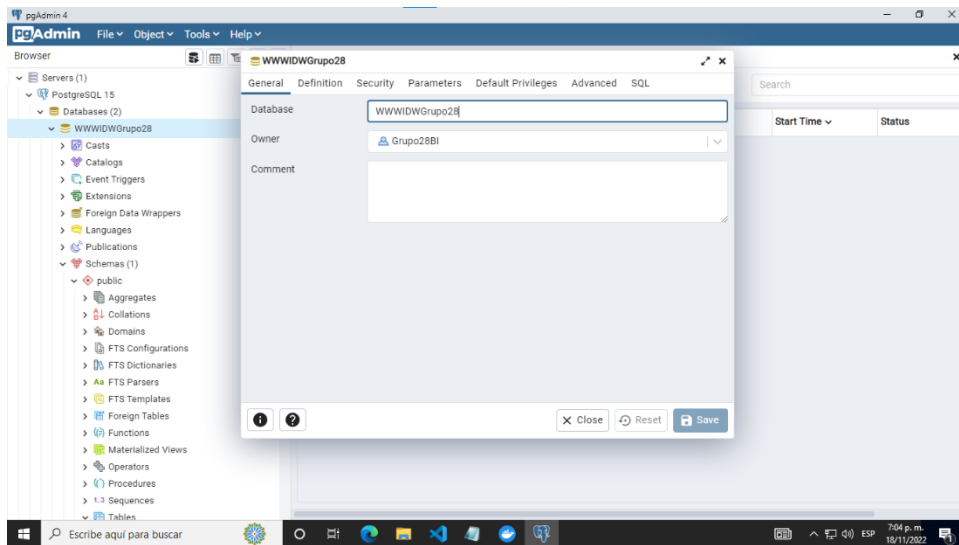
Errores más difíciles para solucionar

La forma en la que explicaron la conexión con la base de datos no funcionó al principio, así que tuvimos que buscar la forma de que funcionara.

Pasos

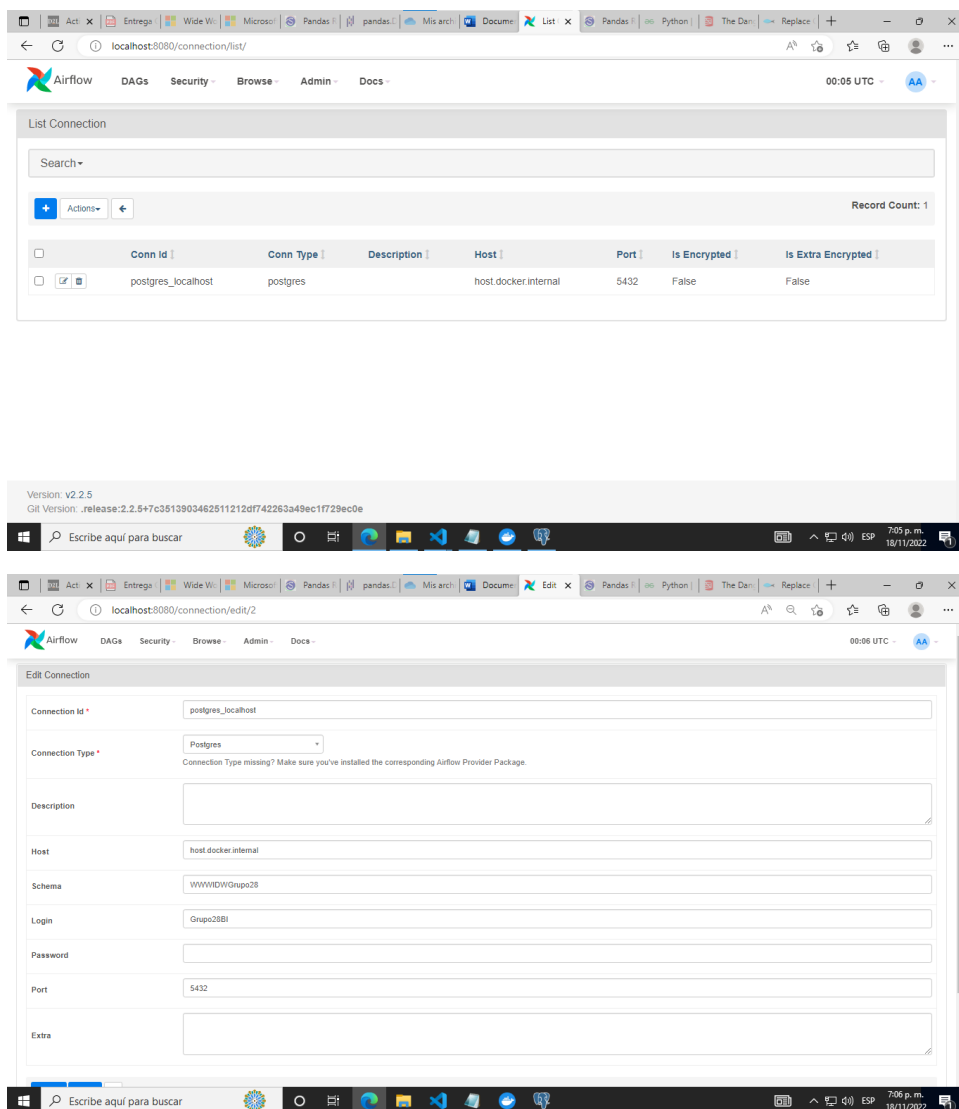
Creación de bases de datos:





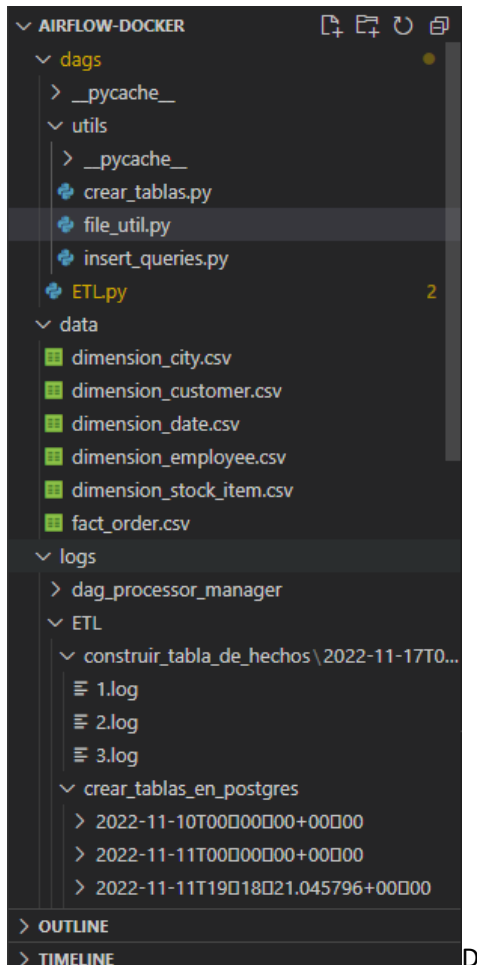
Esta es la base de datos que creamos de acuerdo con lo sugerido en el enunciado. Se creó una base de datos llamada WWWIDWGrupo28, en donde el dueño es el usuario Grupo28BI

Creación de una conexión postgres en Airflow:

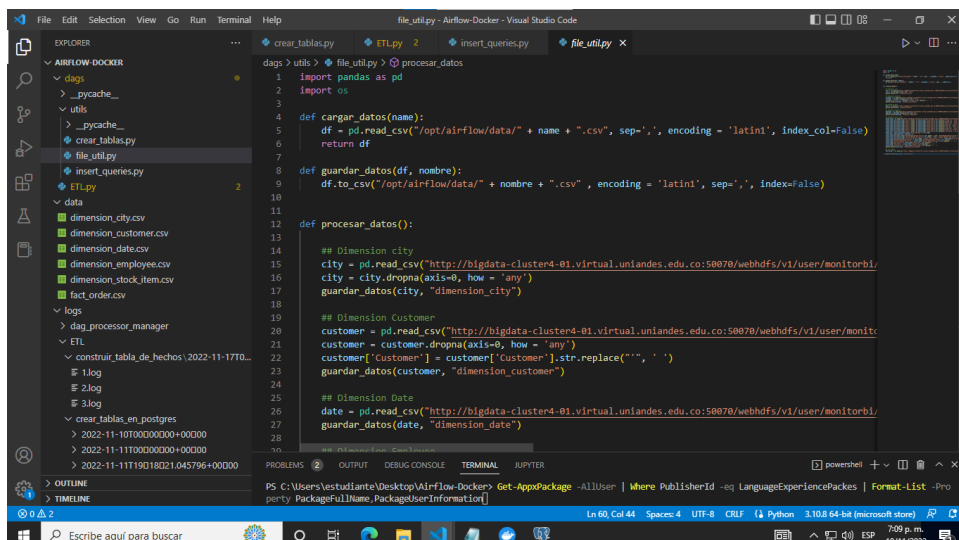


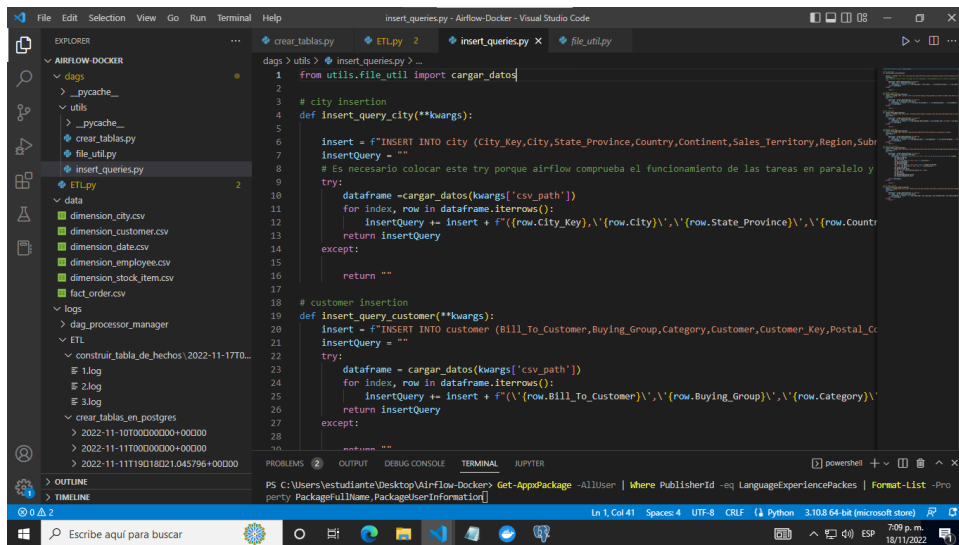
Se creó una conexión en Airflow a la base de datos con los datos que aparecen en pantalla y la contraseña de la base de datos.

Crear archivos de utilidad para los DAGS

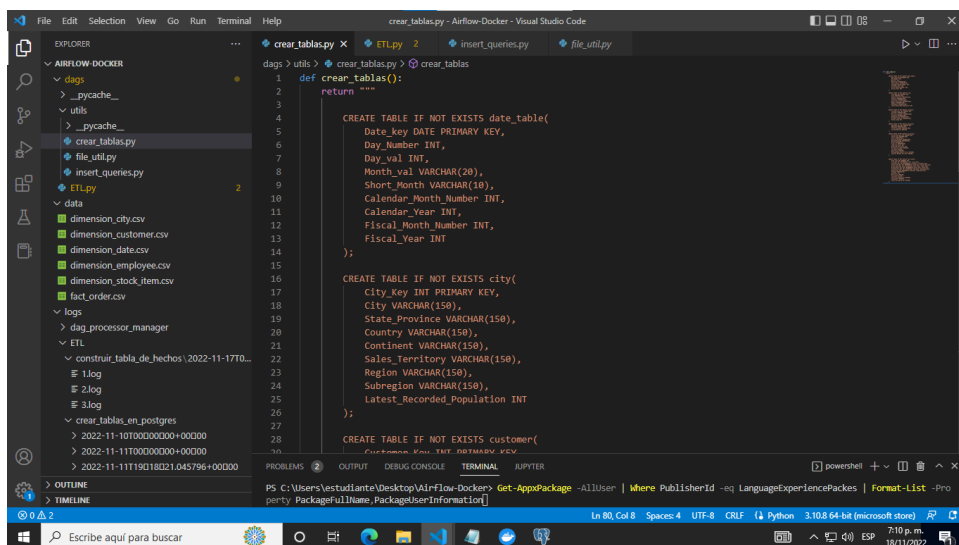


Se creó un folder llamado Utils en donde se crearon los archivos de utilidad para el DAG, estos archivos se llaman crear_tablas, file_util e insert_queries



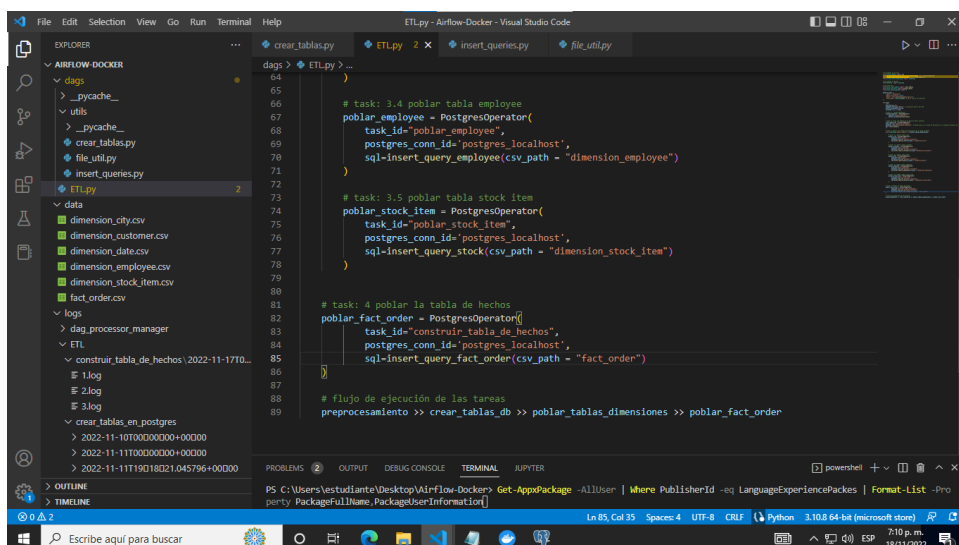


```
1 from utils.file_util import cargar_datos
2
3 # city insertion
4 def insert_query_city(**kwargs):
5
6     insert = f"INSERT INTO city (City_Key, City, State_Province, Country, Continent, Sales_Territory, Region, Subregion, Latest_Recorded_Population) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
7     insertQuery = ""
8     # Es necesario colocar este try porque airflow comprueba el funcionamiento de las tareas en paralelo y
9     try:
10         dataframe = cargar_datos(kwargs['csv_path'])
11         for index, row in dataframe.iterrows():
12             insertQuery += insert + f"({row.City_Key}, '{row.City}', '{row.State_Province}', '{row.Country}', '{row.Continent}', '{row.Sales_Territory}', '{row.Region}', '{row.Subregion}', '{row.Latest_Recorded_Population}'), "
13         return insertQuery
14     except:
15         return ""
16
17 # customer insertion
18 def insert_query_customer(**kwargs):
19     insert = f"INSERT INTO customer (Bill_To_Customer, Buying_Group, Category, Customer, Customer_Key, Postal_Code, Region, Subregion, Latest_Recorded_Population) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
20     insertQuery = ""
21     # Es necesario colocar este try porque airflow comprueba el funcionamiento de las tareas en paralelo y
22     try:
23         dataframe = cargar_datos(kwargs['csv_path'])
24         for index, row in dataframe.iterrows():
25             insertQuery += insert + f"({row.Bill_To_Customer}, '{row.Buying_Group}', '{row.Category}', '{row.Customer}', '{row.Customer_Key}', '{row.Postal_Code}', '{row.Region}', '{row.Subregion}', '{row.Latest_Recorded_Population}'), "
26         return insertQuery
27     except:
28         return ""
```



```
1 def crear_tablas():
2     return ""
3
4 CREATE TABLE IF NOT EXISTS date_table(
5     Date_Key DATE PRIMARY KEY,
6     Day_Number INT,
7     Day_Val INT,
8     Month_Val VARCHAR(20),
9     Short_Month VARCHAR(10),
10    Calendar_Month_Number INT,
11    Calendar_Year INT,
12    Fiscal_Month_Number INT,
13    Fiscal_Year INT
14);
15
16 CREATE TABLE IF NOT EXISTS city(
17     City_Key INT PRIMARY KEY,
18     City VARCHAR(150),
19     State_Province VARCHAR(150),
20     Country VARCHAR(150),
21     Continent VARCHAR(150),
22     Sales_Territory VARCHAR(150),
23     Region VARCHAR(150),
24     Subregion VARCHAR(150),
25     Latest_Recorded_Population INT
26);
27
28 CREATE TABLE IF NOT EXISTS customer(
29     Customer_Key INT PRIMARY KEY,
30     Customer VARCHAR(150),
31     Buying_Group VARCHAR(150),
32     Category VARCHAR(150),
33     Bill_To_Customer VARCHAR(150),
34     Postal_Code VARCHAR(150),
35     Region VARCHAR(150),
36     Subregion VARCHAR(150),
37     Latest_Recorded_Population INT
38);
```

Implementar el DAG



```
1 # task: 3.4 poblar tabla employee
2 poblar_employee = PostgresOperator(
3     task_id="poblar_employee",
4     postgres_conn_id="postgres_localhost",
5     sql_insert_query_employee(csv_path = "dimension_employee")
6 )
7
8 # task: 3.5 poblar tabla stock item
9 poblar_stock_item = PostgresOperator(
10    task_id="poblar_stock_item",
11    postgres_conn_id="postgres_localhost",
12    sql_insert_query_stock(csv_path = "dimension_stock_item")
13 )
14
15 # task: 4 poblar la tabla de hechos
16 poblar_fact_order = PostgresOperator(
17    task_id="construir_tabla_de_hechos",
18    postgres_conn_id="postgres_localhost",
19    sql_insert_query_fact_order(csv_path = "fact_order")
20 )
21
22 # flujo de ejecución de las tareas
23 preprocesamiento >> crear_tablas_db >> poblar_tablas_dimensiones >> poblar_fact_order
```

Se creó el archivo ETL y se pegó lo que solicitaba el enunciado.

Consultas a las bases de datos

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_order/WWIDWGrupo28/postgres@PostgreSQL 15

public.fact_order/WWIDWGrupo28/postgres@PostgreSQL 15

Query Query History

```
1 SELECT * FROM public.fact_order
2 ORDER BY order_key ASC
```

Data Output Messages Notifications

	order_key [PK] integer	city_key integer	customer_key integer	stock_item_key integer	order_date date	picked_date date	salesperson_key integer	picker_key integer	package character varying (50)	quantity integer	unit_price numeric	tax_rate numeric
1	1	91	179	533	2014-02-05	2013-03-17	135	122	S	805	237.77	61
2	2	83	2	631	2015-11-17	2013-07-19	2	133	S	76	721.71	59
3	3	47	390	174	2015-04-29	2015-11-03	128	75	S	585	2866.71	9
4	4	8	218	157	2015-04-04	2014-12-03	84	191	S	878	4671.07	39
5	5	94	167	235	2015-01-26	2015-01-11	91	197	S	583	1950.68	24
6	6	3	376	124	2014-02-10	2013-06-05	135	181	S	727	775.09	13
7	7	10	328	44	2014-02-07	2014-07-29	105	11	S	679	4081.01	37
8	8	54	325	629	2013-10-23	2014-07-01	56	56	M	80	1852.75	6
9	9	64	263	357	2014-01-29	2015-01-18	184	25	S	777	295.73	41
10	10	87	236	607	2015-08-10	2014-06-25	71	200	XL	881	2036.81	22
11	11	63	191	132	2016-08-12	2015-04-22	40	203	S	454	2308.12	3

Total rows: 1000 of 1000 Query complete 00:00:00.417 Ln 1, Col 1

Captura para la tabla fact_order

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_order public.stockitem/WWIDWGrupo28/postgres@PostgreSQL 15

public.stockitem/WWIDWGrupo28/postgres@PostgreSQL 15

Query Query History

```
1 SELECT * FROM public.stockitem
2 ORDER BY stock_item_key ASC
```

Data Output Messages Notifications

	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)	lead_time_days integer	quantity_per_c integer
1	1	Void fill 400 L bag (Whi...	White	Each	Each	nan	400L	14	
2	2	Void fill 300 L bag (Whi...	White	Each	Each	nan	300L	14	
3	3	Void fill 200 L bag (Whi...	White	Each	Each	nan	200L	14	
4	4	Void fill 100 L bag (Whi...	White	Each	Each	nan	100L	14	
5	5	Air cushion machine (B...	Blue	Each	Each	nan	nan	20	
6	6	Air cushion film 200m...	nan	Each	Each	nan	325m	14	
7	7	Air cushion film 200m...	nan	Each	Each	nan	325m	14	
8	8	Large replacement bla...	nan	Each	Each	nan	18mm	14	
9	9	Small 9mm replaceme...	nan	Each	Each	nan	9mm	14	
10	10	Packing knife with met...	Yellow	Each	Each	nan	18mm	14	
11	11	Packing knife with met...	Yellow	Each	Each	nan	9mm	14	

Total rows: 671 of 671 Query complete 00:00:00.281 Ln 1, Col 1

Captura para la tabla stockitem

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_order public.stockitem public.city/WWIDWGrupo28/postgres@PostgreSQL 15

public.city/WWIDWGrupo28/postgres@PostgreSQL 15

Query Query History

```
1 SELECT * FROM public.city
2 ORDER BY city_key ASC
```

Data Output Messages Notifications

	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)	subregion character varying (150)	latest_r integer
1	1	Carrollton	New York	United States	North America	Mideast	Americas	Northern America	
2	2	Carrollton	Virginia	United States	North America	Southeast	Americas	Northern America	
3	3	Carrollton	Illinois	United States	North America	Great Lakes	Americas	Northern America	
4	4	Carrollton	Missouri	United States	North America	Plains	Americas	Northern America	
5	5	Carrollton	Ohio	United States	North America	Great Lakes	Americas	Northern America	
6	6	Carrollton	Kentucky	United States	North America	Southeast	Americas	Northern America	
7	7	Carrollton	Georgia	United States	North America	Southeast	Americas	Northern America	
8	8	Carrollton	Alabama	United States	North America	Southeast	Americas	Northern America	
9	9	Carrollton	Mississippi	United States	North America	Southeast	Americas	Northern America	
10	10	Carrollton	Texas	United States	North America	Southwest	Americas	Northern America	
11	11	Carrollton Manor	Maryland	United States	North America	Mideast	Americas	Northern America	

Total rows: 97 of 97 Query complete 00:00:00.309 Ln 1, Col 1

Captura para la tabla city

pgAdmin 4

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_ord... public.stockite... public.city/WW... public.cust...

Query Query History

```
1 SELECT * FROM public.customer
2 ORDER BY customer_key ASC
```

Data Output Messages Notifications

customer_key	customer	bill_to_customer	category	buying_group	primary_contact	postal_code
1	Tailspin Toys (Head Of...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Waldemar Fisar	90410
2	Tailspin Toys (Sylvant...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Lorena Cindric	90216
3	Tailspin Toys (Peepies ...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Bhaargav Rambhatta	90205
4	Tailspin Toys (Medicin...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Daniel Roman	90152
5	Tailspin Toys (Gaspert...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Johanna Hurling	90261
6	Tailspin Toys (Jessie...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Biswajeet Thakur	90298
7	Tailspin Toys (Frankiew...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Kalidas Nadar	90761
8	Tailspin Toys (Bow Mar...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Kanti Kotadia	90484
9	Tailspin Toys (Netcong...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Sointu Aalto	90129
10	Tailspin Toys (Wimbled...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Siddhartha Pankar	90061
11	Tailspin Toys (Devault...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Elnaz Javan	90185
12	Tailspin Toys (Biscay...	Tailspin Toys (Head Of...	Novelty Shop	Tailspin Toys	Heloisa Fernandes	90054

Total rows: 402 of 402 Query complete 00:00:00.315 Ln 1, Col 1

Captura para la tabla customer

pgAdmin 4

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_ord... public.stockite... public.city/WW... public.cust...

Query Query History

```
1 SELECT * FROM public.date_table
2 ORDER BY date_key ASC
```

Data Output Messages Notifications

date_key	day_number	month_val	short_month	calendar_month_number	calendar_year	fiscal_month_number	fiscal_year
1	2013-01-01	1	January	Jan	1	2013	3
2	2013-01-02	2	January	Jan	1	2013	3
3	2013-01-03	3	January	Jan	1	2013	3
4	2013-01-04	4	January	Jan	1	2013	3
5	2013-01-05	5	January	Jan	1	2013	3
6	2013-01-06	6	January	Jan	1	2013	3
7	2013-01-07	7	January	Jan	1	2013	3
8	2013-01-08	8	January	Jan	1	2013	3
9	2013-01-09	9	January	Jan	1	2013	3
10	2013-01-10	10	January	Jan	1	2013	3
11	2013-01-11	11	January	Jan	1	2013	3
12	2013-01-12	12	January	Jan	1	2013	3

Total rows: 1000 of 1461 Query complete 00:00:00.406 Ln 1, Col 1

Captura para la tabla date_table

pgAdmin 4

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.fact_ord... public.stockite... public.city/WW... public.cust...

Query Query History

```
1 SELECT * FROM public.employee
2 ORDER BY employee_key ASC
```

Data Output Messages Notifications

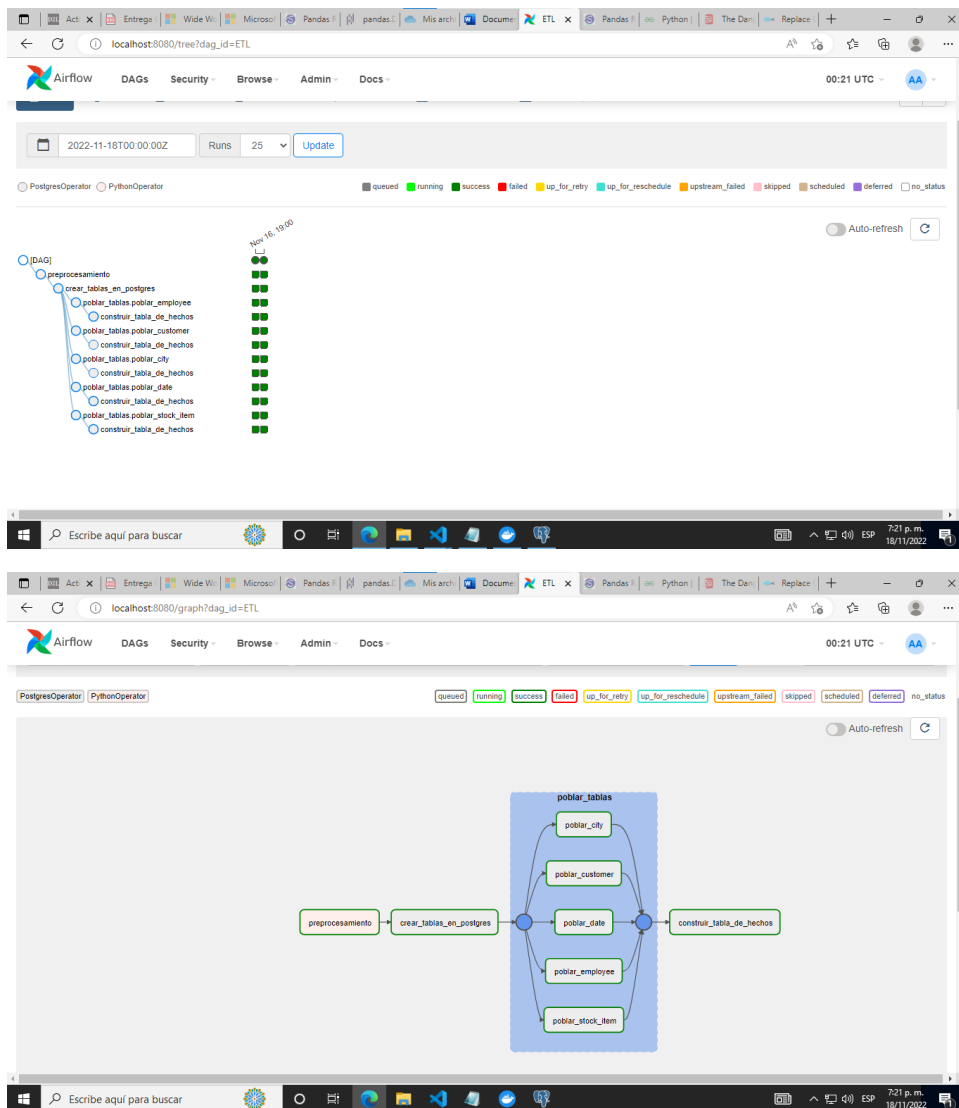
employee_key	employee	preferred_name	is_salesperson
1	Lily Code	Lily	true
2	Isabella Rupp	Isabella	false
3	Ethan Onslow	Ethan	false
4	Amy Treff	Amy	true
5	Jai Shand	Jai	false
6	Anthony Grosse	Anthony	true
7	Taj Shand	Taj	true
8	Hudson Hollinworth	Hudson	true
9	Jack Potter	Jack	true
10	Piper Koch	Piper	false
11	Hudson Onslow	Hudson	true
12	Sophia Hinton	Sophia	true

Total rows: 212 of 212 Query complete 00:00:00.315 Ln 1, Col 1

Successfully run. Total query runtime: 315 msec. 212 rows affected.

Captura para la tabla employee

Capturas del arbol y grafo en ejecución



Preguntas:

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

Task: Es la unidad básica de ejecución, estás se organizan en DAGs y se establecen dependencias ascendentes y descendentes para saber el orden en que deben llevarse a cabo.

Operator: Es una plantilla que utiliza una tarea, estos se pueden definir dentro del DAG.

Airflow tiene un conjunto muy amplio de operadores disponibles, con algunos integrados en el núcleo o proveedores preinstalados. Algunos operadores populares del núcleo incluyen:

- BashOperator- ejecuta un comando bash
- PythonOperator- llama a una función de Python arbitraria

- EmailOperator- envía un correo electrónico
- Use el @taskdecorador para ejecutar una función de Python arbitraria. No admite la representación de plantillas jinja pasadas como argumentos.

DAG: Un DAG recopila tareas organizadas con sus dependencias y relaciones correspondientes para decir como deben ejecutarse. El DAG no se preocupa por lo que hace cada tarea, su ocupación es simplemente el orden de ejecución y especificaciones de ejecución como cantidad y tiempos.

- ¿Por qué para la columna de día se utiliza el nombre “day_val” y no “day”?

Este nombre se usa porque al usar solo el nombre day es abierto a la interpretación de que la columna se puede referir al número de día en el mes, el número de día en la semana, el nombre del día de la semana o incluso a la fecha entera.

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

Esta información se basa en el diseño que se determina para el modelo multidimensional ya que las columnas que se crean son directamente los atributos de las dimensiones.

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

El flujo de ejecución le indica a la máquina virtual de Apache en que orden se deben ejecutar las tareas que están en el DAG.

- ¿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?

En caso de que en la bodega de datos ya se encuentren generadas las tablas que se esperan en el diseño entonces se debería remover la parte de la creación de datos y solo agregar los nuevos a la bodega. En caso de que la bodega contenga tablas diferentes a las deseadas, el proceso deberá traer los datos existentes, adaptarlos al nuevo modelo (agregando o eliminando las columnas que sean o no necesarias, como llaves subrogadas) y sumarle los datos nuevos para ser cargados a la bodega.

- ¿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?

Para esto se debe agregar a la tabla la columna de las nuevas llaves primarias que se asignen de modo incremental y renombrar en la creación de la tabla la columna con la llave transaccional afectando los archivos de crear tablas e insertar los datos con los nombres nuevos para las columnas. Además, la referencia de la tabla de hechos también se debe renombrar.

```
# employee insertion
def insert_query_employee(**kwargs):
    insert = f"INSERT INTO employee (Employee_Key_T,Employee,Preferred_Name,Is_Salesperson) VALUES "
    insertQuery = ""
    try:
        dataframe = cargar_datos(kwargs['csv_path'])
        for index, row in dataframe.iterrows():
            insertQuery += insert + f"({row.Employee_Key},{row.Employee},{row.Preferred_Name},{row.Is_Salesperson}) VALUES "
        return insertQuery
    except:
        return ""
```

```
CREATE TABLE IF NOT EXISTS employee(
    Employee_Key_DWH SERIAL PRIMARY KEY,
    Employee_Key_T INT,
    Employee VARCHAR(150),
    Preferred_Name VARCHAR(150),
    Is_Salesperson BOOLEAN
);
```

```
CREATE TABLE IF NOT EXISTS fact_order(
    Order_Key INT PRIMARY KEY,
    City_Key INT REFERENCES city (city_key),
    Customer_Key INT REFERENCES customer (customer_key),
    Stock_Item_Key INT REFERENCES stockitem (stock_item_key),
    Order_Date_Key DATE REFERENCES date_table (date_key),
    Picked_Date_Key DATE REFERENCES date_table (date_key),
    Salesperson_Key INT REFERENCES employee (employee_key_DWH),
    Picker_Key INT REFERENCES employee (employee_key_DWH),
    Package VARCHAR(50),
    Quantity INT,
    Unit_Price DECIMAL,
    Tax_Rate DECIMAL,
    Total_Excluding_Tax DECIMAL,
    Tax_Amount DECIMAL,
    Total_Including_Tax DECIMAL
);
```

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

Al cargar los datos a la bodega se debe realizar una transformación sobre la fecha para que esta sea almacenada en la llave primaria como el entero que se desea.

Referencias

<https://airflow.apache.org/>

<https://airflow.apache.org/docs/>

<https://docs.docker.com/desktop/windows/>

https://github.com/apache/airflow/tree/main/airflow/example_dags

<https://hadoop.apache.org/docs/stable/>

<https://www.postgresql.org/docs/>

<https://docs.microsoft.com/es-es/sql/samples/wide-world-importers-what-is?view=sql-server-ver15>

[3] Capítulo 20 (desde pág. 512 hasta pág. 520). - [3] KIMBALL, Ralph, ROSS, Margy. "The Data Warehouse Toolkit: the definitive guide to dimensional modeling". Third Edition. John Wiley & Sons, Inc, 2013.

Opc. [6] Capítulo 11 (págs. 464-472) - [6] KIMBALL, Ralph, ROSS, Margy, BECKER, Bob, MUNDY Joy, and THORNTHWAITE, Warren. "The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence Remastered Collection". Wiley, 2016