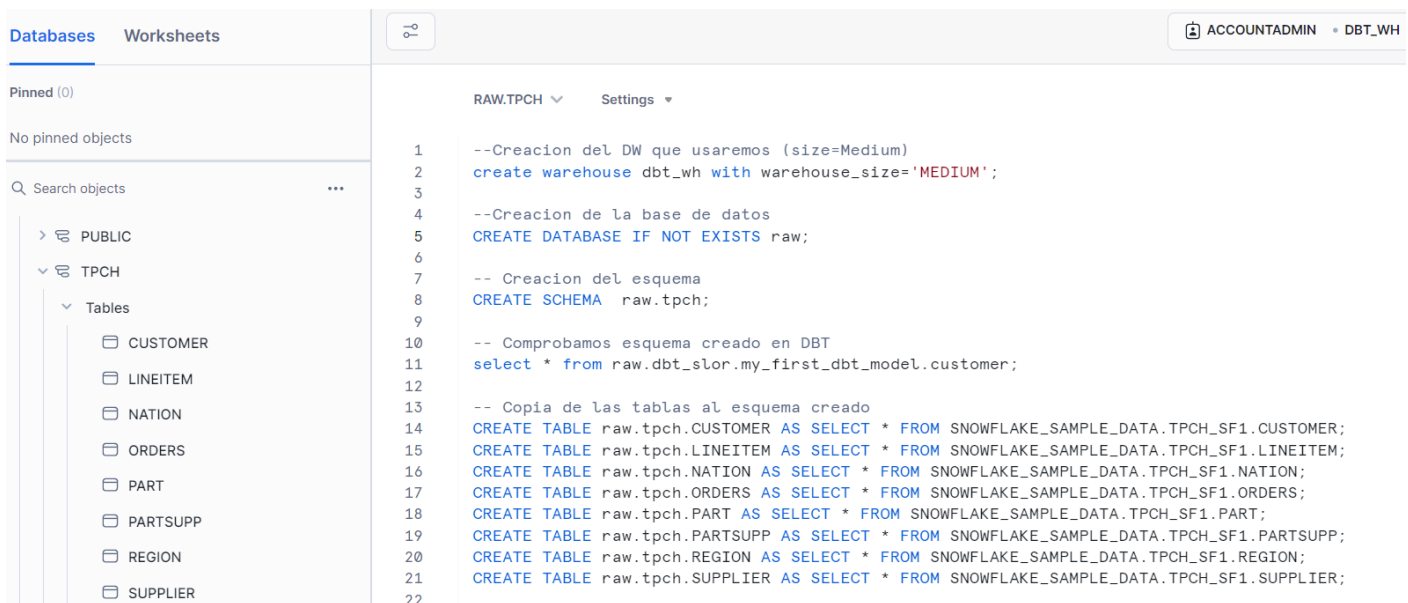


# Entrega Final Proyecto

El primer paso a realizar fue la conexión entre las distintas plataformas para la gestión del proyecto. Por un lado, Snowflake, es un almacén de datos en la nube que permite el almacenamiento y análisis de grandes volúmenes de datos. DBT es una herramienta de transformación de datos basada en SQL y que tomará gran importancia durante el proyecto y GitHub es una plataforma de desarrollo colaborativo que ofrece control de versiones para proyectos de software.

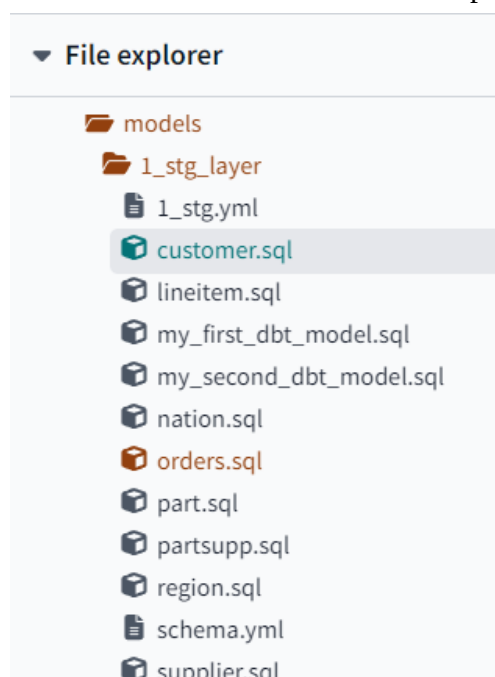
## SNOWFLAKE

A continuación, se realiza un análisis superficial para saber qué información contienen las tablas y que significa cada campo. En esta primera fase, copio las tablas a otro esquema en snowflake ya que el original no es editable.



## DBT

-Creo una carpeta en DBT, 'stg\_layer', para proceder a la ingestión de los datos y a realizar pequeños cambios como la eliminación de la columna 'comment' que no proporciona información de utilidad o el uso de otros nombres para ciertas columnas que son ambiguas.



-Ejemplo:

models > 1\_stg\_layer > customer.sql

```
1 With customer AS (  
2   SELECT  
3     c_custkey,  
4     c_name as customer_name,  
5     c_address as customer_address,  
6     c_nationkey,  
7     c_acctbal,C_MKTSEGMENT  
8   FROM tpch.customer  
9 )  
10  
11 Select * from customer
```

-A continuación, para la segunda capa del proyecto en la que aplicaremos algunas transformaciones más notorias,decido crear la carpeta 'mid\_layer'. En la que creo distintas métricas que pueden ser de utilidad para posteriores análisis. Algunas de ellas son:

- precio descontado por artículo en una línea de pedido
- precio base por artículo en una línea de pedido
- Cantidad total del descuento (valor producto-menos el descuento-menos el impuesto) por artículo

Podrían servir para responder a preguntas como:

- Como varía el precio descontado por articulo en una línea de pedido según el tipo de cliente o segmento de mercado.
- Porcentaje del precio final de venta que se atribuye al descuento y al impuesto por artículo en una línea de pedido.
- Influencia del descuento e impuesto en la compra de ciertos artículos

mid\_orders\_line.sql

models > 2\_mid\_layer > mid\_orders\_line.sql

```
8 with mid_orders_line AS (  
9   SELECT  
10    li.L_ORDERKEY,  
11    o.O_ORDERKEY,  
12    o.O_CUSTKEY,  
13    o.O_ORDERDATE,  
14    li.L_INESTATUS AS order_status_code,  
15    li.L_PARTKEY,  
16    li.L_SUPPKEY,  
17    li.L_RETURNFLAG,  
18    li.L_INENUMBER,  
19    li.L_SHIPDATE,  
20    li.L_COMMITDATE,  
21    li.L_RECEIPTDATE,  
22    li.L_SHIPMODE,  
23    li.L_EXTENDEDPRI / NULLIF(li.L_QUANTITY, 0) AS item_price, --precio base por articulo en una línea de pedido  
24    li.L_DISCOUNT,  
25    (li.L_EXTENDEDPRI * (1 - li.L_DISCOUNT)) AS item_discounted_price, -- precio descontado por articulo en una línea de pedido  
26    li.L_EXTENDEDPRI, -- precio total articulo sin descuentos  
27    li.L_TAX, --tasa impuestos articulo  
28    ((li.L_EXTENDEDPRI + (-1 * li.L_EXTENDEDPRI * li.L_DISCOUNT)) * li.L_TAX) AS item_tax_amount -- cantidad total del impuesto calcula  
29  FROM  
30    tpch.lineitem li  
31  JOIN  
32    tpch.orders o ON o.O_ORDERKEY = li.L_ORDERKEY  
33  ORDER BY  
34    o.O_ORDERDATE
```

Defer to production

-Para la última capa, la cual llamo '*analytics\_layer*', defino la tabla de hechos a partir de la '*mid\_layer*' seleccionando distintas operaciones, como la media o el sumatorio de distintas métricas anteriores.

```
fact_orders_line.sql  X

models > 3_analytics_layer > fact_orders_line.sql

1  WITH orders_line_agg AS (
2      SELECT
3          ol.O_ORDERKEY,
4          ol.item_discounted_price as discounted_price_item,
5          AVG(ol.item_discounted_price) AS avg_discounted_price,
6          SUM(ol.item_discounted_price) AS item_discounted_amount,
7          SUM(ol.item_tax_amount) AS item_tax_amount
8      FROM
9          mid_orders_line ol
10     GROUP BY
11         ol.O_ORDERKEY, discounted_price_item
12 )
13 SELECT
14     o.O_ORDERKEY,
15     o.O_ORDERDATE,
16     o.O_CUSTKEY,
17     o.O_ORDERSTATUS
```

Además, creo varias dimensiones usando joins. Por ejemplo, la dimensión customers:

```
sergio-f90-patch-1

dim_customers.sql  dim_suppliers.sql

models > 3_analytics_layer > dim_customers.sql

1  {{ config(materialized = 'table') }}
2
3  WITH cust_reg_nat AS (
4      SELECT
5          c.c_custkey,
6          c.c_name,
7          c.c_address,
8          c.c_acctbal,
9          c.C_MKTSEGMENT,
10         n.n_nationkey,
11         n.N_NAME,
12         r.r_regionkey,
13         r.R_NAME
14     FROM
15         tpch.customer c
16     INNER JOIN tpch.nation n
17         ON c.c_nationkey=n.n_nationkey
18     INNER JOIN tpch.region r
19         ON n.N_REGIONKEY=r.r_regionkey
20 )
21
22 select
23     *
24 from
25     cust_reg_nat
26 order by
27     c_custkey
```

La dimensión suppliers, que agrupa varias tablas:

The screenshot shows the dbt CLI interface. On the left, the 'File explorer' pane displays the project structure, including 'Snowflake-DBT', 'analyses', 'dbt\_packages', 'macros', 'models', '1\_stg\_layer', '2\_mid\_layer', '3\_analytics\_layer', 'core.yml', 'dim\_customers.sql', 'dim\_suppliers.sql' (selected), 'fact\_orders\_line.sql', 'views', 'seeds', 'snapshots', 'target', and 'tests'. The main pane shows the SQL code for 'dim\_suppliers.sql' in the '3\_analytics\_layer' model. The code defines a 'WITH' clause for 'part\_psup\_sup' and a 'select' statement that joins 'part p', 'partsup ps', 'supplier s', 'nation n', and 'region r' based on their respective keys. The 'select' statement includes columns: 'ps.part\_supplier\_key', 'p.part\_key', 's.supplier\_key', 's.S\_NAME', 's.S\_ADDRESS', 's.S\_ACCTBAL', 'n.n\_nationkey', 'n.N\_NAME as supplier\_nation\_name', 'r.r\_regionkey as supplier\_region\_key', and 'r.R\_NAME as supplier\_region\_name'.

## Vistas:

Creación de varias vistas para un análisis en concreto:

-(EJEMPLO) Análisis detallado de las órdenes de compra(beneficio,margen,etc.) de productos para clientes del segmento de mercado "HOUSEHOLD" (familiar) entre 1992 y 1993

The screenshot shows the dbt CLI interface with the 'view\_orders\_household.sql' file open. The code defines a view that calculates various metrics for orders from the 'HOUSEHOLD' segment between 1992 and 1993. The 'SELECT' statement includes columns for order keys, dates, customer keys, market segments, discounted prices, item taxes, part names, supplier costs, and calculated metrics: 'total\_sales\_per\_part', 'total\_profit\_per\_order', and 'margin\_of\_profit'. The 'FROM' clause lists 'FACT\_ORDERS\_LINE fo'. The 'JOIN' clauses connect 'CUSTOMER c', 'PART p', and 'PARTSUPP ps' to 'fo'. The 'WHERE' clause filters for orders between '1992-01-01' and '1993-12-31' and for the 'HOUSEHOLD' market segment. The 'ORDER BY' clause orders by 'fo.o\_orderdate'.

# Git-Hub

sergio-f90 / Snowflake-DBT

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

Files

main

Go to file

macros

models

1\_stg\_layer

1\_stg.yml

customer.sql

lineitem.sql

my\_first\_dbt\_model.sql

my\_second\_dbt\_model.sql

nation.sql

orders.sql

part.sql

partsupp.sql

region.sql

schema.yml

supplier.sql

2\_mid\_layer

mid\_orders\_line.sql

mid\_orders\_line.yml

3\_analytics\_layer

core.yml

dim\_customers.sql

dim\_suppliers.sql

fact\_orders\_line.sql

views

view\_costsupplier\_automobile.sql

view\_orders\_summary.sql

view\_salesclient\_date\_incremental.sql

seeds

Snowflake-DBT / models /

sergio-f90 last\_updates

Name	Last commit message
..	
1_stg_layer	updates_keys
2_mid_layer	last_updates
3_analytics_layer	last_updates
views	last_updates