EJERCICIO 1A. PASOS PARA RESOLVER LA CREACIÓN DEL MODELO VISTAS

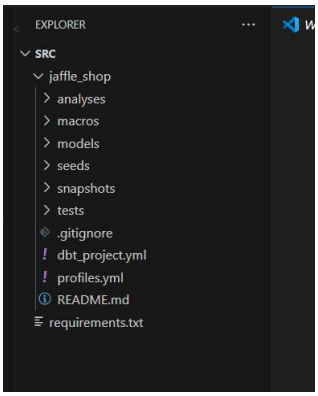
1.- Desktop: Para contenedores y arrancar el contenedor dbt_container

2.- Ejecutar una línea de comandos y lanzar las siguientes sentencias:

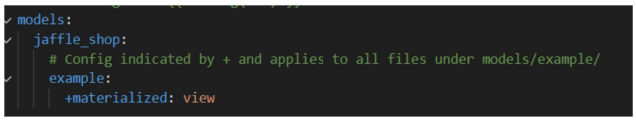
docker exec -ti dbt_container /bin/bash
cd app/src
ls jaffle_shop

```
C:\Windows\System32>docker exec -ti dbt_container /bin/bash
root@94e4d886efa2:/# cd app/src
root@94e4d886efa2:/app/src# ls
jaffle_shop  requirements.txt
root@94e4d886efa2:/app/src# cd jaffle_shop/
root@94e4d886efa2:/app/src/jaffle_shop# ls
README.md  dbt_packages     jaffle_shop.duckdb  macros  profiles.yml  snapshots  tests
analyses   dbt_project.yml  logs                models  seeds         target
root@94e4d886efa2:/app/src/jaffle_shop#
```

3.- Abrir **Visual Studio Code** y abrir la carpeta src del proyecto **PRO-000-dbt/src**

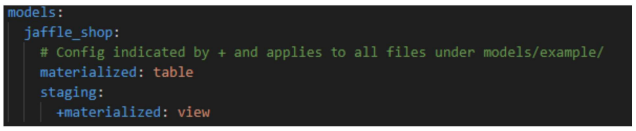


4.- Fichero ***dbt_project.yml*** en este fichero se encuentra toda la configuración del proyecto dbt, donde se especifica por ejemplo el nombre del proyect, la configuración de los distintos tipos de ficheros, etc.
En este caso, queremos modelar vistas y tablas para ello en la parte de models vamos a realizar la siguiente modificación:



```
models:
  jaffle_shop:
    # Config indicated by + and applies to all files under models/example/
    example:
      +materialized: view
```

Imagen 1



```
models:
  jaffle_shop:
    # Config indicated by + and applies to all files under models/example/
    materialized: table
    staging:
      +materialized: view
```

Imagen 2

En la imagen 2 lo que estamos haciendo es decir que todo lo que esté dentro de la carpetea de Models se va a materializar en Tabla y todo lo que esté dentro de la carpeta staging se va a materializar en vista.

5.- Creamos la carpeta staging dentro de la carpeta models.

6.- Crea el fichero SQL dentro de la carpeta staging llamado **stg_customers.sql**, donde el único cambio realizado del origen es renombrar la columna id por customer_id.
Quedando así el fichero:

src > jaffle_shop > models > staging > stg_customers.sql

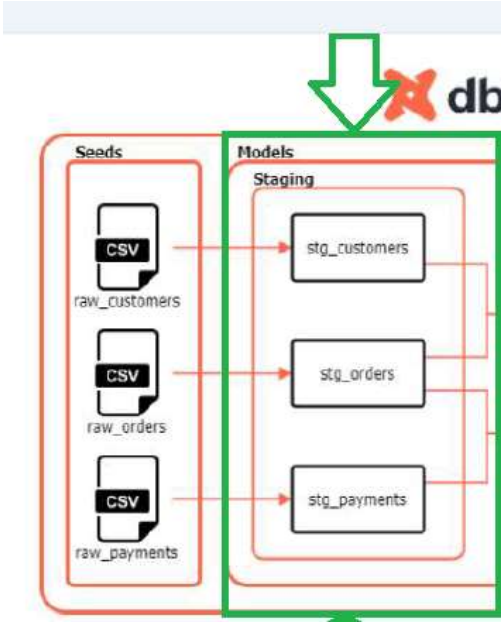
```sql
with source as (
    select * from {{ ref('raw_customers')}}
),

renamed as (
    select
        id as customer_id,
        first_name,
        last_name
    from source
)

select * from renamed

```

7.- Crea el fichero stg_orders.sql con los siguientes cambios:

- orders:
  - id renombra order_id
  - user_id renombra customer_id

8.- Crea el fichero stg_paymets.sql con los siguientes cambios:

- payment
  - id renombra payment_id
  - amount /100 renombra amount



9.- Crea un fichero dentro de la carpeta staging llamado **schema.yml donde se especifica el esquema del modelo.**
Ejemplo de **stg_customers** y el resto lo debéis de completar vosotros **stg_orders** y **stg_payments**

src > jaffle_shop > models > staging > ! schema.yml

```yaml
version: 2

models:
  - name: stg_customers
    colums:
      - name: customer_id
        tests:
          - unique
          - not_null
      - name: first_name
      - name: last_name

```

Nota: Ponemos la primera línea version: 2 que es la versión que tenemos del proyecto dbt_project.yml y la etiqueta models:.

10.- A continuación lanzamos los siguientes comandos para crear las vistas y sus datos DuckDB database usando duckcli.

Comandos:
- Cargar los ficheros csv ejecutando el comando:
    dbt seed
- Compilar un sólo modelo dentro dbt build:
    dbt build -f -s <nombre del modelo>
- Compilar todos los modelos:
    dbt build
- Ejecutar duckcli y realizar la consulta select from table:
    cd app/src/jaffle_shop/
    duckcli jaffle_shop.duckdb
    *select * from <nombre de la tabla>*
- Salir de duckcli
    exit

Ejemplo:

customer_id	first_name	last_name
1	Michael	P.
2	Shawn	M.
3	Kathleen	P.
4	Jimmy	C.
5	Katherine	R.
6	Sarah	R.
7	Martin	M.
8	Frank	R.
9	Jennifer	F.
10	Henry	W.
11	Fred	S.
12	Amy	D.
13	Kathleen	M.
14	Steve	F.
15	Teresa	H.
16	Amanda	H.
17	Kimberly	R.

EJERCICIO 1B. PASOS PARA RESOLVER LA CREACIÓN DEL MODELO TABLAS

1.- En la carpeta models creamos el fichero **customers_orders.sql** cuya tabla contendrá joins de las vistas de staging customers y orders.

customers_orders
customer_id
customer_full_name
first_order
most_recent_order
number_of_orders

Siendo así:

src > jaffle_shop > models > customers_orders.sql

```sql
with source_customers as (
    select * from {{ ref('stg_customers')}}
),

source_orders as (
    select * from {{ ref('stg_orders')}}
),

final as (
    select c.customer_id,
           c.first_name || ' ' || c.last_name as customer_full_name,
           (select min(order_date) from source_orders where customer_id = c.customer_id) as first_order,
           (select max(order_date) from source_orders where customer_id = c.customer_id) as most_recent_order,
           (select count(order_id) from source_orders where customer_id = c.customer_id) as number_of_orders
    from source_customers c, source_orders o
    where c.customer_id = o.customer_id
)

select * from final
```

2.- Realizad vosotros el nuevo fichero sql de customers_payments.sql que cuya tabla contendrá joins de las vistas de staging customers, orders y payments.

customers_payments
customer_id
number_of_orders
total_amount

3.- Crea el fichero *schema.yml* dentro de la carpeta de *models* quedando así la primera parte customers_orders:

```yaml
version: 2

models:
  - name: customers_orders
    colums:
      - name: customer_id
        tests:
          - unique
          - not_null
      - name: customer_full_name
      - name: first_order
      - name: most_recent_order
      - name: number_of_orders
```

Nota: Ponemos la primera línea version: 2 que es la versión que tenemos del proyecto dbt_project.yml y la etiqueta models:.

4.- Añadid vosotros en el fichero *schema.yml* la información de la tabla customers_payments.

5.- A continuación lanzamos los siguientes comandos para crear las tablas y sus datos DuckDB database usando duckcli.

Comandos:

- Compilar un sólo modelo dentro dbt build:
    dbt build -f -s <nombre del modelo>
- Compilar todos los modelos:
    dbt build
- Ejecutar duckcli y realizar la consulta select from table:
    cd app/src/jaffle_shop/
    duckcli jaffle_shop.duckdb
    *select * from <nombre de la tabla>*
- Salir de duckcli
    exit