```
-- ------------------------------------------------
-- Proyecto 2 - Exploratory Data Analysis con SQL
-- Análisis de Datos con SQL
-- ------------------------------------------------


-- ------------------------------------------------
-- Importar el archivo "supermarket_sales.csv"
-- (https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales)
-- -- Generacion de Tabla "supermarket_sales"
-- ------------------------------------------------

CREATE TABLE IF NOT EXISTS public.supermarket_sales
(
Id character varying(50) NOT NULL,
Branch character varying(5) NOT NULL,
City character varying(50) NOT NULL,
Customer_Type character varying(20) NOT NULL,
Gender character varying(20) NOT NULL,
Product_Line character varying(50) NOT NULL,
Price Float NOT NULL,
Quantity INTEGER NOT NULL,
Tax Float NOT NULL,
Total Float NOT NULL,
purchase_date DATE,
purchase_time TIME NOT NULL,
Payment character varying(30) NOT NULL,
Cogs Float NOT NULL,
Gross_Margin_Percentage Float NOT NULL,
Gross_Income Float NOT NULL,
Rating Float NOT NULL,

CONSTRAINT supermarket_sales_Id PRIMARY KEY (Id)
);

select * from supermarket_sales;

ALTER TABLE supermarket_sales
ALTER COLUMN purchase_date TYPE VARCHAR,
ALTER COLUMN purchase_time TYPE VARCHAR;

COPY supermarket_sales(Id, Branch, City, Customer_Type,
                       Gender, Product_Line, Price, Quantity,
                       Tax, Total, purchase_date, purchase_time,
                       Payment, Cogs, Gross_Margin_Percentage, Gross_Income,
                       Rating)
FROM 'D:\WORK IN PROGRESS\Data Analytics course\parte 3 my sql\week 40\supermarket_sales
- Sheet1.csv' (FORMAT CSV, HEADER, DELIMITER ',');

select * from supermarket_sales;

ALTER TABLE supermarket_sales ALTER COLUMN purchase_date TYPE DATE
using to_date(purchase_date, 'MM/DD/YYYY');

ALTER TABLE supermarket_sales ALTER COLUMN purchase_time TYPE TIME
using TO_TIMESTAMP(purchase_time, 'HH24:MI:SS');

select * from supermarket_sales LIMIT 5;
```

```
-- Control de importacion
-- Contar el numero de registros y validar que sea el mismo numero del archivo original
select count(Id) from supermarket_sales;

----------------------------------------
--  Reporte total de ventas en $ de toda la base de datos,
-- por mes,
-- ordenados por mes
----------------------------------------

select
    date_part('month',purchase_date) as Mes,
    sum(total) as total_ventas
from supermarket_sales ss
group by 1
order by 1;

----------------------------------------
-- Reporte de la factura promedio de toda la base de datos,
-- por mes,
-- ordenados por mes
----------------------------------------

select
    date_part('month',purchase_date) as Mes,
    ROUND(avg( total ):: numeric ,2) as Promedio_ventas
from supermarket_sales ss
group by 1
order by 1;


----------------------------------------
--Reporte de número total de ventas (COUNT)
-- por mes,
-- ordenados por mes
----------------------------------------

select
    date_part('month',purchase_date) as Mes,
    Count( total ) as Cantidad_ventas
from supermarket_sales ss
group by 1
order by 1;


----------------------------------------
--Reporte de número de clientes (COUNT)
-- por mes,
-- ordenados por mes
----------------------------------------

-- Para este reporte utilizo la función de crosstable
-- pero para hacerlo tengo que cambiar los registros "member" de la columna customer_type
-- yaque la palabra "member" hace parte de las palabras usadas por postgresql.

UPDATE supermarket_sales
SET Customer_Type = 'Normal_customer'
WHERE Customer_Type = 'Normal';
```

```sql
UPDATE supermarket_sales
SET Customer_Type = 'Member_customer'
WHERE Customer_Type = 'Member';


select *
from crosstab('select extract(month from purchase_date)::INTEGER as Mes,
            Customer_Type,
            count(*)::integer
            from supermarket_sales ss
            group by 1,2
            order by 1,2')
as final_result(Mes int,Normal_customer int, Member_customer int );


select *
from crosstab('select
            Customer_Type,
            extract(month from purchase_date) as Mes,
            count(*)::integer
            from supermarket_sales ss
            group by 2, rollup (1)
            order by 1,2')
as final_result(Customer_Type varchar, "1" INTEGER, "2" INTEGER, "3" INTEGER);


----------------------------------------
--Reporte ranking de los ítems más vendidos (TOP 10)
-- por mes
----------------------------------------

-- Enero

select
    product_line,
    count(product_line)as cantidad,
    RANK () OVER ( ORDER BY count(product_line) DESC) rank_by_ventas
from supermarket_sales ss
where date_part('month',purchase_date)= '1'
group by 1;


-- Febrero

select
    product_line,
    count(product_line)as cantidad,
    RANK () OVER ( ORDER BY count(product_line) DESC) rank_by_ventas
from supermarket_sales ss
where date_part('month',purchase_date)= '2'
group by 1;

-- Marzo

select
    product_line,
    count(product_line)as cantidad,
    RANK () OVER ( ORDER BY count(product_line) DESC) rank_by_ventas
```

```
from supermarket_sales ss
where date_part('month',purchase_date)= '3'
group by 1;

-- Si quiero ver todos los resultados en una sola tabla

select *
from crosstab('select extract(month from purchase_date)::INTEGER as Mes,
            product_line,
            count(*)::integer
            from supermarket_sales ss
            group by 1,2
            order by 1,2')
as final_result(Mes int,Home_and_lifestyle int, Electronic_accessories int,
Health_and_beauty int,
Food_and_beverages int, Fashion_accessories int, Sports_and_travel int);



SELECT *
FROM crosstab('SELECT
        product_line,
        extract(month FROM purchase_date) AS Mes,
        count(*)::integer AS count
     FROM supermarket_sales
     GROUP BY 1, 2
     ORDER BY 1, 2')
AS final_result(product_line VARCHAR, "1" INTEGER, "2" INTEGER, "3" INTEGER);

-------------------------------------------------------------------------
-- KPIs:
-- 1) venta total,
-- 2) ticket promedio total,
-- 3) ranking de ítems totals (TOP 20) por ventas totales
-- 4) ranking de ítems más vendidos por número de unidades vendidas
-------------------------------------------------------------------------

-- 1) venta total

SELECT
    round(sum(total)::numeric,2) as total_ventas
    from supermarket_sales ss;

-- 2) ticket promedio total,

SELECT
    round(avg(total)::numeric,2) as total_ventas
    from supermarket_sales ss;

-- 3) ranking de ítems totals (TOP 20) por ventas totales

select
    product_line,
    sum(total) as total_ventas,
    RANK () OVER ( ORDER BY sum(total) DESC) rank_by_ventas
from supermarket_sales ss
group by 1;

-- 4) ranking de ítems más vendidos por número de unidades vendidas
```

```sql
select
    product_line,
    count(product_line) as cantidades_vendidas,
    RANK () OVER ( ORDER BY count(product_line) DESC) rank_by_cantidades_vendidas
from supermarket_sales ss
group by 1;


-- adicional

SELECT *
FROM crosstab('SELECT
        product_line,
        gender,
        sum(total)::integer AS count
     FROM supermarket_sales
     GROUP BY 1, 2
     ORDER BY 1, 2')
AS final_result(product_line VARCHAR, Male INTEGER, Female INTEGER);


SELECT *
FROM crosstab('SELECT
        product_line,
        extract(month FROM purchase_date) AS Mes,
        count(*)::integer AS count
     FROM supermarket_sales
     GROUP BY 1, 2
     ORDER BY 1, 2')
AS final_result(product_line VARCHAR, "1" INTEGER, "2" INTEGER, "3" INTEGER);


select
    extract(hour FROM purchase_time) AS Hora,
    count(id)
    from supermarket_sales ss
    group by 1
    order by 1


select *
FROM crosstab('SELECT
            extract(hour FROM purchase_time)::INTEGER AS Hora,
            product_line,
            count(product_line):: integer as cantidad_ventas
            from supermarket_sales ss
            group by rollup(1),2
            order by 1,2')
as final_result(Hora int,Home_and_lifestyle int, Electronic_accessories int,
Health_and_beauty int,
Food_and_beverages int, Fashion_accessories int, Sports_and_travel int);
```