

EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: SUBCONSULTAS Y CONSULTAS AVANZADAS.
- EXERCISE 2: REALIZANDO CONSULTAS EN BASE A UN MODELO ENTIDAD - RELACIÓN.

EXERCISE 1: SUBCONSULTAS Y CONSULTAS AVANZADAS.

Una subconsulta es una instrucción **SELECT**, **INSERT**, **UPDATE** o **DELETE**, anidada dentro de otra instrucción o subconsulta.

Dada la siguiente sentencia **SELECT**: **SELECT columnas FROM expresión**; *expresión* puede ser reemplazada por una subconsulta, de la siguiente forma:

```
1 SELECT columnas FROM (SELECT columnas FROM Tabla)
```

Ejemplo:

```
1 SELECT CURREN_DATE AS Fecha
```

Se podría escribir con una subconsulta, así:

```
1 SELECT Fecha FROM (SELECT CURREN_DATE AS Fecha ) AS FechaActual;
```

Ambas consultas producen el mismo resultado.

Las subconsultas ayudan a solucionar algunas consultas complejas, las cuales no se pueden resolver de forma directa o a través de una relación directa, sino que primero deben pasar por otra antes de ser relacionadas.

En este caso, se relaciona la tabla **T1** con la subconsulta **SC1**, por medio de **SC1.D1 = T1.C3**.

```
1 SELECT T1.C1, T1.C2, SC1.D2
2 FROM T1 ON (
```

```
3 SELECT D1, SUM(D2) AS D2
4 FROM T2
5 WHERE D3 = 5
6 GROUP BY D1
7 ) SC1 ON SC1.D1 = T1.C3
```

Además, podemos tener subconsultas en SQL, donde se permitan expresiones con algunas restricciones.

```
1 SELECT E.RUT,
2 E.Nombre,
3 (SELECT Tipo FROM Tipos WHERE Tipo = E.Tipo) TipoEmpleado
4 FROM Empleados AS E
5 WHERE EXISTS(SELECT * FROM Contrato AS C WHERE C.RUT = E.RUT)
6 AND Rol IN (SELECT Rol FROM Roles WHERE Clase = 'Administrador')
```

En ésta, se utilizan subconsultas para determinar la columna **TipoEmpleado**. Si ella retorna más de un registro, se producirá un error, pues en un caso solo puede evaluarse una expresión, y solo con un resultado posible.

Además, se emplean subconsultas para determinar si existe algún registro de una subconsulta, y en caso de existir, la fila actual es agregada al resultado. Por último, se comprueba si el campo rol pertenece a alguno de la clase 'Administrador'.

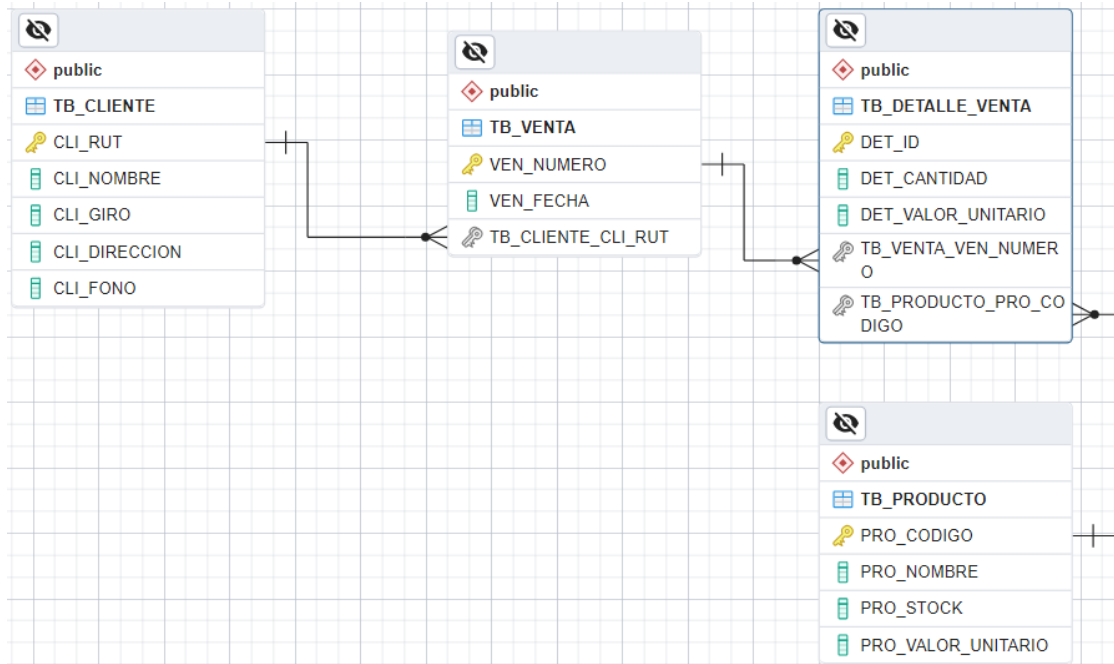
CONSULTAS COMPLEJAS

Algunas consultas presentan todo un desafío, desde el punto de vista del rendimiento y complejidad para entender el código SQL que se genera.

Para poder resolverlas, éstas deben ir implementándose por partes, hasta llegar a la solución de lo que se requiere.

Ejemplo:

Dado el siguiente modelo de datos:



Se solicita la siguiente consulta: informe de compras realizadas por los clientes en un determinado año.

Periodo	Nombre Cliente	Cantidad Compras	Total Periodo	Clasificación

Detallando la tabla:

- Periodo: el año del informe.
- Nombre Cliente: nombre del cliente.
- Cantidad Compras: cantidad de compras realizadas en el periodo.
- Total Periodo: monto total bruto de compras realizadas en el periodo.

- Clasificación: clasificación del cliente según total comprado en el periodo y la siguiente tabla:

Desde	Hasta	Clasificación
0	150.000	Ocasional
150.001	500.000	Gold
500.001	más	Premiun

Determinar las tablas del modelo que intervienen, y verificar que se cuenta con toda la información para diseñar la consulta.

- Periodo: VEN_FECHA.
- Nombre Cliente: CLI_NOMBRE.
- Cantidad Compras: se deben contar la cantidad de ventas del periodo.
- Total Periodo: DET_VALOR_UNITARIO, DET_CANTIDAD.
- Clasificación: DET_VALOR_UNITARIO, DET_CANTIDAD.

Tablas para responder este informe:

- TB_VENTA.
- TB_DETALLE_VENTA.
- TB_CLIENTE.

Se debe partir diseñando esta consulta de adentro hacia afuera, es decir, iniciando por las tablas con más detalle, hacia las más generales o maestras.

ANALICEMOS LAS COLUMNAS

- Periodo: se obtiene de forma directa.
- Nombre Cliente: se obtiene de forma directa.
- Cantidad Compras: sumar la cantidad de ventas.
- Total Periodo: sumar DET_VALOR_UNITARIO * DET_CANTIDAD agrupados por cliente, y año determinado.
- Clasificación: el Total Periodo debe ser clasificado mediante una sentencia CASE.

Debemos partir por la tabla de mayor detalle, hacia la de menor:

```
1 SELECT *
2 FROM TB_DETALLE_VENTA
```

Ahora, se agrupan para formar el total por venta:

```
1 SELECT DET_VEN_NUMERO,
2 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) SubTotal
3 FROM TB_DETALLE_VENTA
4 GROUP BY DET_VEN_NUMERO
```

Relacionamos la consulta anterior con la tabla TB_VENTA. Ahora, la relación entre ambas es de 1:1. Además, aprovechamos de agregar el filtro por periodo:

```
1 SELECT *
2 FROM TB_VENTA
3 INNER JOIN (
4 SELECT DET_VEN_NUMERO,
5 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) AS TotalVenta
6 FROM TB_DETALLE_VENTA
7 GROUP BY DET_VEN_NUMERO
8 ) DETALLE ON DET_VEN_NUMERO = VEN_NUMERO
9 WHERE TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) = 2021
```

Luego, se deben agregar todas las columnas del reporte solicitado, dejando aquellas que no tenemos, o que son de compleja solución, expresadas como constantes; con esto, podemos saber que nos está faltando e ir probando de forma rápida nuestra consulta.

```
1 SELECT TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) AS Periodo,
2 '' AS NombreCliente,
3 0 AS CantidadCompras,
4 0 AS Total,
5 '' AS Clasificacion
```

```
6 FROM TB_VENTA
7 INNER JOIN (
8 SELECT DET_VEN_NUMERO,
9 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) AS TotalVenta
10 FROM TB_DETALLE_VENTA
11 GROUP BY DET_VEN_NUMERO
12 ) DETALLE ON DET_VEN_NUMERO = VEN_NUMERO
13 WHERE TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) = 2021
```

Se debe agregar la agrupación para las ventas:

```
1 SELECT TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) AS Periodo,
2 '' AS NombreCliente,
3 COUNT(*) AS CantidadCompras,
4 SUM(TotalVenta) AS Total,
5 '' AS Clasificacion
6 FROM TB_VENTA
7 INNER JOIN (
8 SELECT DET_VEN_NUMERO,
9 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) AS TotalVenta
10 FROM TB_DETALLE_VENTA
11 GROUP BY DET_VEN_NUMERO
12 ) DETALLE ON DET_VEN_NUMERO = VEN_NUMERO
13 WHERE TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) = 2021
14 GROUP BY TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')), CLI_NOMBRE
```

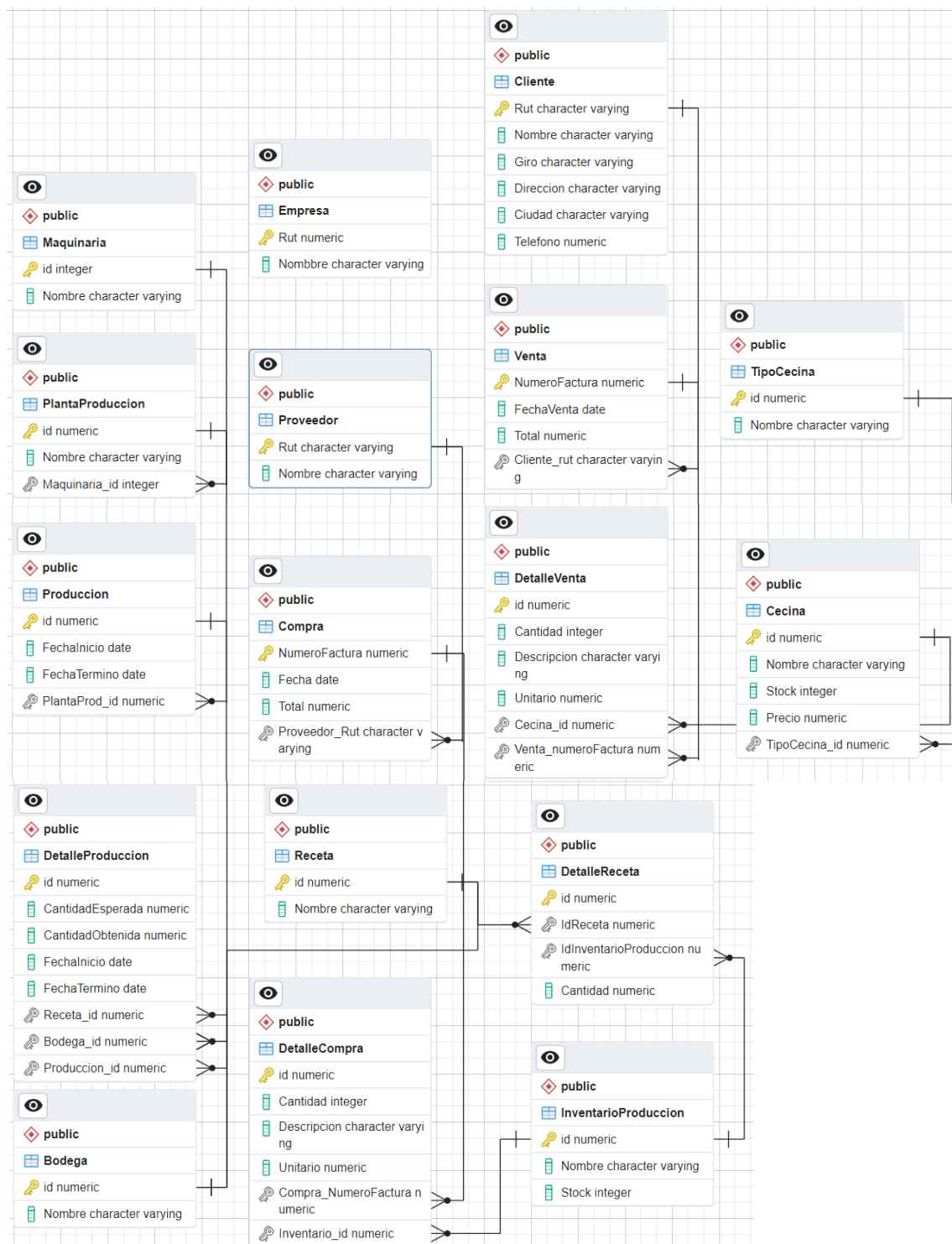
Diseñar las consultas más complejas:

```
1 SELECT TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) AS Periodo,
2 '' AS NombreCliente,
3 COUNT(*) AS CantidadCompras,
4 SUM(TotalVenta) AS Total,
5 CASE WHEN SUM(TotalVenta) <= 150000 THEN 'Ocasional'
6 WHEN SUM(TotalVenta) <= 500000 THEN 'Gold'
```

```
7 ELSE 'Premium' END AS Clasificacion
8 FROM TB_VENTA
9 INNER JOIN (
10 SELECT DET_VEN_NUMERO,
11 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) AS TotalVenta
12 FROM TB_DETALLE_VENTA
13 GROUP BY DET_VEN_NUMERO
14 ) DETALLE ON DET_VEN_NUMERO = VEN_NUMERO
15 WHERE TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) = 2021
16 GROUP BY TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')), CLI_NOMBRE
```

Y por último, agregar las tablas maestras para resolver las columnas faltantes:

```
1 SELECT TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) AS Periodo,
2 CLI_NOMBRE AS NombreCliente,
3 COUNT(*) AS CantidadCompras,
4 SUM(TotalVenta) AS Total,
5 CASE WHEN SUM(TotalVenta) <= 150000 THEN 'Ocasional'
6 WHEN SUM(TotalVenta) <= 500000 THEN 'Gold'
7 ELSE 'Premium' END AS Clasificacion
8 FROM TB_VENTA
9 INNER JOIN (
10 SELECT DET_VEN_NUMERO,
11 SUM(DET_VALOR_UNITARIO * DET_CANTIDAD) AS TotalVenta
12 FROM TB_DETALLE_VENTA
13 GROUP BY DET_VEN_NUMERO
14 ) DETALLE ON DET_VEN_NUMERO = VEN_NUMERO
15 INNER JOIN TB_CLIENTE ON CLI_RUT = VEN_CLI_RUT
16 WHERE TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')) = 2021
17 GROUP BY TO_NUMBER(TO_CHAR(VEN_FECHA, 'YYYY')), CLI_NOMBRE
```



Para comenzar a desarrollar las consultas, trabajaremos en pgAdmin. Comenzaremos listando los clientes sin ventas en los últimos 4 meses, a través de una subconsulta. Comenzaremos escribiendo:

```
1 SELECT * FROM CLIENTE
```

Query Query History

```
1 SELECT * FROM "Cliente"
2
3
```

Data Output Messages Notifications

	Rut [PK] character	Nombre character var	Giro character var	Direccion character varying	Ciudad character varying	Telefono numeric
1	123456-k	Cecil	Carpintero	San Diego 1314	Santiago	555666
2	456789-0	Salim	Militar	Sierra Leona 2489	Puerto Montt	555888
3	789123-8	Dafne	Panadero	Santa Rosa 0284	Arica	555000

Posteriormente, realizaremos otro **select**:

```
1 SELECT * FROM VENTA
```

Query Query History

```
1 SELECT * FROM "Venta"
2
```

Data Output Messages Notifications

	NumeroFactura [PK] numeric	FechaVenta date	Total numeric	Cliente_rut character vary
1	1	2020-01-15	10000	789123-8
2	2	2020-06-18	15000	456789-0
3	3	2020-07-05	20000	456789-0
4	4	2020-09-20	18000	123456-k
5	5	2020-12-29	8000	123456-k

Continuaremos creando un filtro, el cual aplicaremos sobre el atributo "FechaVenta", para mostrar las ventas de los últimos 4 meses.

Para realizar esto, utilizaremos algunas funciones. Escribiremos:

```
SELECT "Cliente_rut" FROM "Venta"
```

Ahora utilizaremos EXTRACT, que nos permite filtrar las ventas realizadas en el año 2020


```
WHERE EXTRACT(YEAR FROM "FechaVenta") = 2020
```

Continuamos con DATE_PART que nos permitirá extraer partes específicas de un valor de fecha o de tiempo, como primer argumento recibe la cadena que especifica la unidad de tiempo que se desea extraer en este caso "MONTH" (mes) y como segundo argumento recibe el valor de fecha o de tiempo del cual se extraerá la parte por último "IN" verifica si el valor extraído con "DATE_PART" se encuentra en la lista proporcionada, que contiene los meses 9, 10, 11 y 12.

```
AND (DATE_PART('MONTH', "FechaVenta")) IN ('9', '10', '11', '12')
```

El resultado será:

```
1
2 SELECT "Cliente_rut" FROM "Venta"
3 WHERE EXTRACT(YEAR FROM "FechaVenta") = 2020
4 AND (DATE_PART('MONTH', "FechaVenta")) IN ('9', '10', '11', '12')
5
```

Data Output		Messages	Notifications
			
	Cliente_rut character varying 		
1	123456-k		
2	123456-k		

En nuestro ejemplo se destacaron diferencias respecto al video debido al avance tecnológico. En el video, se utilizó `current_date` en PostgreSQL para acceder a la fecha actual de manera directa. En el ejemplo escrito, recurrimos a `DATE_PART` para acceder a los meses (una función que extrae partes de una fecha). La evolución tecnológica promueve cambios que simplifican y mejoran el código. `current_date` se emplea para obtener la fecha actual, mientras que `DATE_PART` permite extraer componentes específicos de la fecha. Ambas son herramientas valiosas en desarrollo.

Ahora nos falta filtrar por los clientes sin ventas para eso utilizaremos "NOT IN"

```
SELECT * FROM "Cliente"  
WHERE "Rut" NOT IN (  
  
)
```

Y dentro de "NOT IN" colocaremos la consulta donde obtuvimos los clientes que, si tuvieron ventas, quedando nuestra consulta de la siguiente manera:

```
SELECT * FROM "Cliente"  
WHERE "Rut" NOT IN (  
  SELECT "Cliente_rut" FROM "Venta"  
  WHERE EXTRACT(YEAR FROM "FechaVenta") = 2020  
  AND (DATE_PART('MONTH', "FechaVenta")) IN ('9', '10', '11', '12')  
)
```

Al ejecutarla, obtendremos el siguiente resultado:

	Rut [PK] character	Nombre character vary	Giro character vary	Direccion character varying	Ciudad character varyin	Telefono numeric
1	456789-0	Salim	Militar	Sierra Leona 2489	Puerto Montt	555888
2	789123-8	Dafne	Panadero	Santa Rosa 0284	Arica	555000

Seguiremos listando todas las ventas con los atributos: Factura, Fecha, Total, NombreCliente y RutCliente.

Comenzaremos escribiendo:

```
1 SELECT * FROM "Venta"
2 WHERE TO_CHAR("FechaVenta", 'YYYY') = '2020'
```

```
14 SELECT * FROM "Venta"
15 WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```

Data Output Messages Notifications

	NumeroFactura [PK] numeric	FechaVenta date	Total numeric	Cliente_rut character varying
1	1	2020-01-15	10000	789123-8
2	2	2020-06-18	15000	456789-0
3	3	2020-07-05	20000	456789-0
4	4	2020-09-20	18000	123456-k
5	5	2020-12-29	8000	123456-k

Para este caso, estamos filtrando por el año 2020. Reemplazaremos el Asterisco, por los atributos que deseamos obtener.

```
SELECT "NumeroFactura" AS "Factura",
"FechaVenta" AS "Venta",
"Total",
"Nombre" AS "NombreCliente",
"Cliente_rut" AS "RutCliente"
FROM "Venta"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```

Luego, establecemos una relación entre la tabla Venta y Cliente, empleando **INNER JOIN**. Esto quedará de la siguiente forma:

```
SELECT "NumeroFactura" AS "Factura",
"FechaVenta" AS "Venta",
"Total",
"Nombre" AS "NombreCliente",
"Rut" AS "RutCliente"
FROM "Venta"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```

	Factura numeric	Venta date	Total numeric	NombreCliente character varying	RutCliente character varying
1	5	2020-12-29	8000	Cecil	123456-k
2	4	2020-09-20	18000	Cecil	123456-k
3	3	2020-07-05	20000	Salim	456789-0
4	2	2020-06-18	15000	Salim	456789-0
5	1	2020-01-15	10000	Dafne	789123-8

Agregaremos además una nueva columna que estará vacía y se llamará clasificación donde catalogaremos a los clientes según el monto de las ventas

```
SELECT "NumeroFactura" AS "Factura",
       "FechaVenta" AS "Venta",
       "Total",
       "Nombre" AS "NombreCLiente",
       "Rut" AS "RutCLiente",
       '' AS "Clasificacion"
FROM "Venta"
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```

Obteniendo:

	Factura numeric	Venta date	Total numeric	NombreCLiente character varying	RutCLiente character varying	Clasificacion text
1	5	2020-12-29	8000	Cecil	123456-k	
2	4	2020-09-20	18000	Cecil	123456-k	
3	3	2020-07-05	20000	Salim	456789-0	
4	2	2020-06-18	15000	Salim	456789-0	
5	1	2020-01-15	10000	Dafne	789123-8	

En una consulta aparte, obtendremos la información para realizar la clasificación de los clientes por año. Para ello escribiremos:

```
SELECT "Cliente_rut" AS "Rut2",
       SUM ("Total") "TotalCliente"
FROM "Venta"
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
GROUP BY "Cliente_rut"
```

	Rut2 character varying	TotalCliente numeric
1	123456-k	26000
2	456789-0	35000
3	789123-8	10000

Ahora que tenemos escrita la consulta, la incorporamos a la consulta principal mediante un **INNER JOIN**, quedando de la siguiente forma:




```
SELECT "NumeroFactura" AS "Factura",  
       "FechaVenta" AS "Venta",  
       "Total",  
       "Nombre" AS "NombreCliente",  
       "Rut" AS "RutCliente",  
       '' AS "Clasificacion", |  
FROM "Venta"  
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"  
INNER JOIN (  
    SELECT "Cliente_rut" "Rut2",  
           SUM ("Total") "TotalCliente"  
    FROM "Venta"  
    WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'  
    GROUP BY "Cliente_rut"  
)  
VAC ON "Rut2" = "Rut"  
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```

Nos resta clasificar el total cliente. Para desarrollar esta clasificación condicional, utilizamos: **CASE**, **WHEN**, **THEN** y **END**. Finalmente, quedará de la siguiente forma:

```
SELECT "NumeroFactura" AS "Factura",  
       "FechaVenta" AS "Fecha",  
       "Total",  
       "Nombre" AS "NombreCliente",  
       "Rut" AS "RutCliente",  
       CASE WHEN "Total" <= 10000 THEN 'Clase C'  
            WHEN "Total" <= 20000 THEN 'Clase B'  
            ELSE 'Clase A'  
            END "Clasificacion"  
FROM "Venta"  
INNER JOIN "Cliente" ON "Cliente_rut" = "Rut"  
INNER JOIN (  
    SELECT "Cliente_rut" "Rut2",  
           SUM ("Total") "TotalCliente"  
    FROM "Venta"  
    WHERE TO_CHAR("FechaVenta", 'YYYY') = '2020'  
    GROUP BY "Cliente_rut"  
)  
VAC ON "Rut2" = "Rut"  
WHERE TO_CHAR ("FechaVenta", 'YYYY') = '2020'
```



Como resultado, obtendremos cada una de las columnas que debemos consultar.

	Factura numeric 	Fecha date 	Total numeric 	NombreClien character var	RutCliente character var	Clasificacion text
1	4	2020-09-20	18000	Cecil	123456-k	Clase B
2	5	2020-12-29	8000	Cecil	123456-k	Clase C
3	2	2020-06-18	15000	Salim	456789-0	Clase B
4	3	2020-07-05	20000	Salim	456789-0	Clase B
5	1	2020-01-15	10000	Dafne	789123-8	Clase C