

## 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 SYSDATE AS Fecha FROM dual;
```

Se podría escribir con una subconsulta, así:

```
1 SELECT Fecha FROM (SELECT SYSDATE AS Fecha FROM dual);
```

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'.

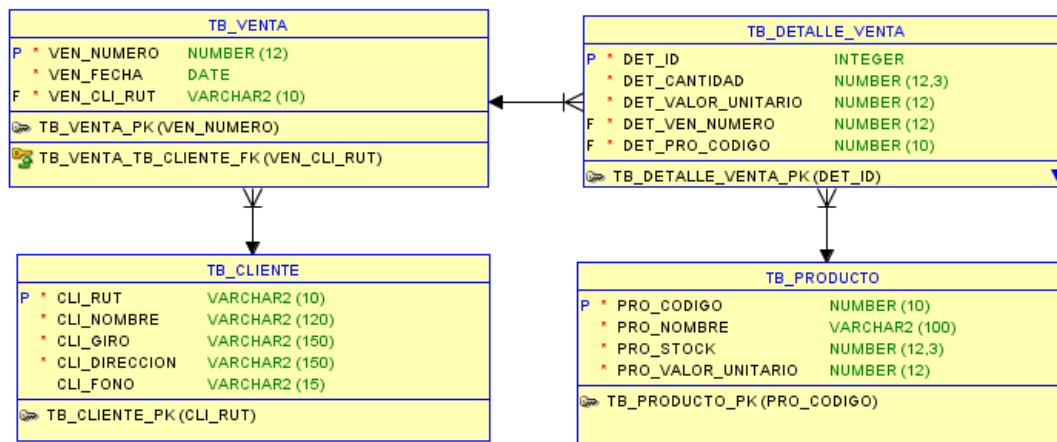
## CONSULTAS COMPLEJAS

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'
```

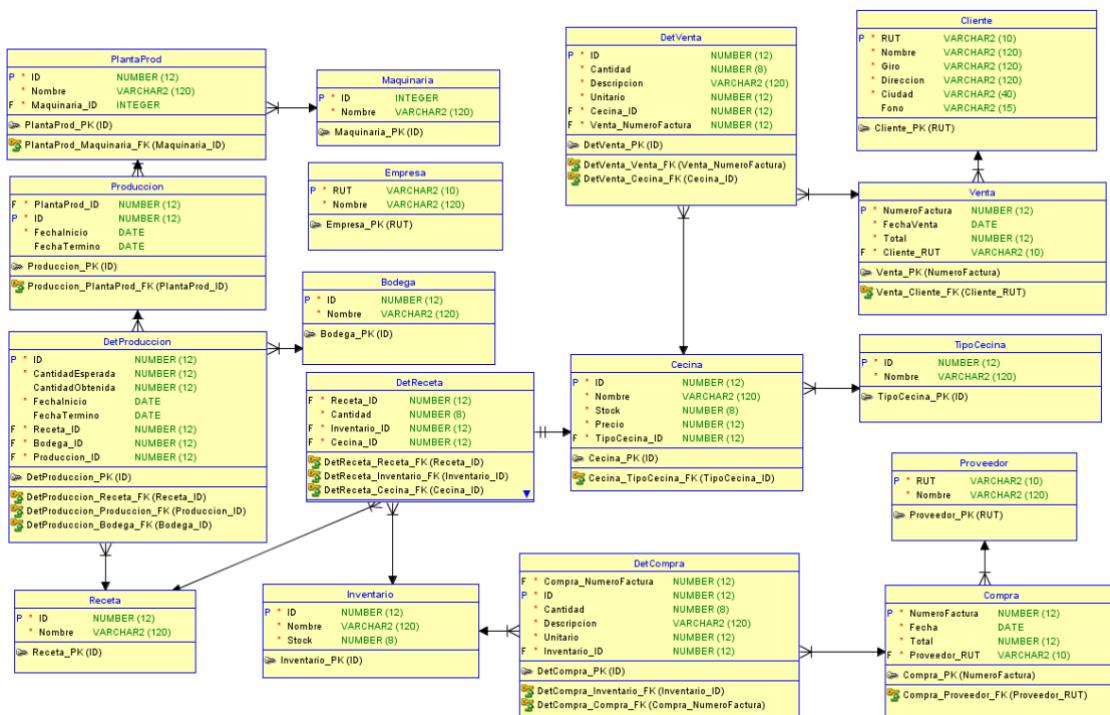
```

6 WHEN SUM(TotalVenta)<=500000 THEN 'Gold'
7 ELSE 'Premiun' 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

```

## EXERCISE 2: REALIZANDO CONSULTAS EN BASE A UN MODELO ENTIDAD RELACIÓN.

Observa el siguiente Modelo Entidad - Relación de la fábrica de cecinas, antes de comenzar con nuestra ejemplificación.



Para desarrollar estas acciones, trabajaremos en SQL Developer. Comenzaremos listando los clientes sin ventas en los últimos 4 meses, a través de una subconsulta. Comenzaremos escribiendo:

```
1 SELECT * FROM CLIENTE
```

Posteriormente, realizaremos otro **select**:

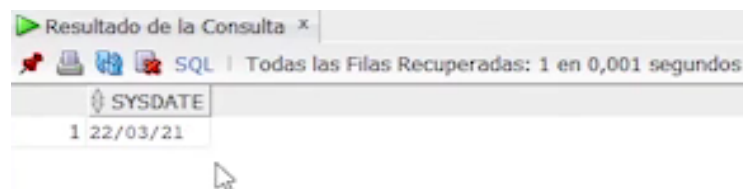
```
1 SELECT * FROM VENTA
```

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:

```
1 SELECT SYSDATE FROM DUAL
```

La sentencia escrita muestra la fecha de hoy, que fue obtenida de una tabla de prueba de expresiones de Oracle llamada "DUAL". Si ejecutamos esta instrucción, en consola obtendremos como resultado la fecha actual:



SYSDATE
1 22/03/21

Continuaremos utilizando la función **ADD\_MONTHS**, que suma N meses a una fecha establecida. De esta forma, la consulta quedará así:

```
1 SELECT ADD_MONTHS(SYSDATE,-4) FROM DUAL
```

Al indicar un signo menos, obtendremos los últimos 4 meses. Ejecutamos la consulta, y el resultado será:



ADD_MONTHS(SYSDATE,-4)
1 22/11/20



Una vez que tenemos la expresión de tiempo, escribiremos en la consulta de la tabla Venta para terminar la expresión, resultando:

```
1 SELECT * FROM VENTA
2 WHERE FECHAVENTA >= ADD_MONTHS (SYSDATE, -4)
```

Aún nos falta filtrar por los clientes sin ventas. La consulta completa queda de la siguiente forma:

```
1 SELECT * FROM CLIENTE
2 WHERE RUT NOT IN (
3 SELECT CLIENTE_RUT FROM VENTA
4 WHERE FECHAVENTA >= ADD_MONTHS (SYSDATE, -4)
5 )
6 )
```

Al ejecutarla, obtendremos el siguiente resultado:

RUT	NOMBRE	GIRO	DIRECCION	CIUDAD	FONO
1 8634715-9	NOMBRE 4	GIRO C	DIRECCION 3	CIUDAD 3	986465473
2 1684455-K	NOMBRE 2	GIRO B	DIRECCION 2	CIUDAD 2	916384002

Recibimos como respuesta todos los clientes ingresados anteriormente, pues no les hemos registrado ninguna venta previa.

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'
```

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

```
1 SELECT NUMEROFACTURA FACTURA,
2 FECHAVENTA FECHA,
3 TOTAL,
4 ' ' NOMBRECLIENTE,
5 ' ' RUTCLIENTE,
```

```
6 ' ' CLASIFICACION
7 FROM VENTA
8 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
```

La indicación ' ' es un alias en blanco.

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

```
1 SELECT NUMEROFACTURA FACTURA,
2 FECHAVENTA FECHA,
3 TOTAL,
4 NOMBRE NOMBRECLIENTE,
5 RUT RUTCLIENTE,
6 ' ' CLASIFICACION
7 FROM VENTA
8 INNER JOIN CLIENTE ON RUT = CLIENTE_RUT
9 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
```

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

```
1 SELECT CLIENTE_RUT, SUM(TOTAL) TOTALCLIENTE
2 FROM VENTA
3 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
4 GROUP BY CLIENTE_RUT
```

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

```
1 SELECT NUMEROFACTURA FACTURA,
2 FECHAVENTA FECHA,
3 TOTAL,
4 NOMBRE NOMBRECLIENTE,
5 RUT RUTCLIENTE,
6 ' ' CLASIFICACION
7 FROM VENTA
8 INNER JOIN CLIENTE ON RUT = CLIENTE_RUT
9 INNER JOIN (SELECT CLIENTE_RUT RUT2, SUM(TOTAL) TOTALCLIENTE
10 FROM VENTA
11 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
12 GROUP BY CLIENTE_RUT
13 ) VAC ON RUT2 = RUT
14 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
```

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

```

1 SELECT NUMEROFACTURA FACTURA,
2 FECHAVENTA FECHA,
3 TOTAL,
4 NOMBRE NOMBRECLIENTE,
5 RUT RUTCLIENTE,
6 CASE WHEN TOTALCLIENTE <= 500000 THEN 'CLASE C'
7 WHEN TOTALCLIENTE <= 5000000 THEN 'CLASE B'
8 ELSE 'CLASE A' END CLASIFICACION
9 FROM VENTA
10 INNER JOIN CLIENTE ON RUT = CLIENTE_RUT
11 INNER JOIN (SELECT CLIENTE_RUT RUT2,
12 SUM(TOTAL) TOTALCLIENTE
13 FROM VENTA
14 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'
15 GROUP BY CLIENTE_RUT
16 ) VAC ON RUT2 = RUT
17 WHERE TO_CHAR(FECHAVENTA, 'YYYY') = '2020'

```

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

