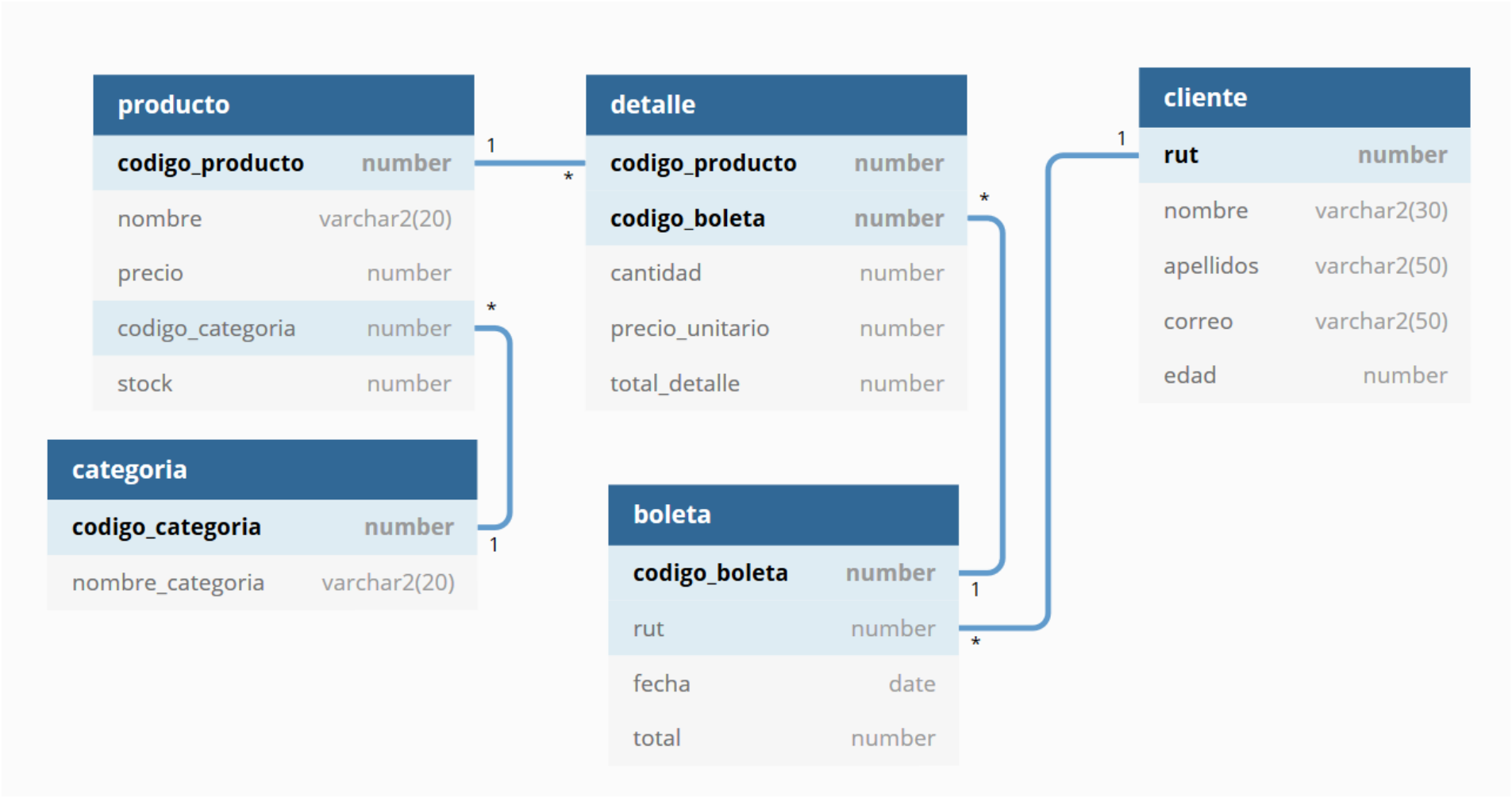
UNIVERSIDAD CATOLICA DEL MAULE

SubConsultas

Programa de hoy

- Uso de la instrucción HAVING en consultas de agrupación.
- Subconsultas
- Dudas de la tarea

Modelo de trabajo de hoy



Consultas de Agrupación

Este tipo de consultas son bastante especiales, ya que solo trabajan con un conjunto de funciones.

La agrupación permite realizar resúmenes sobre la información contenida en la base de datos.

Los resúmenes están enfocados a cantidades, sumas, totales, mínimos y máximos, por decir algo básico de su utilidad.

Consultas de Agrupación

Para realizar estos resúmenes las consultas de agrupación cuentan con las siguientes funciones:

Funciones de agrupación	
AVG	Calcula el promedio sobre una columna
COUNT	Realiza el conteo sobre una columna
MAX	Calcula el máximo valor sobre una columna
MIN	Calcula el mínimo valor sobre una columna
STDDEV	Calcula la desviación estándar sobre una columna
SUM	Realiza la suma sobre una columna
VARIANCE	Calcula la varianza sobre una columna

Recordatorio

Ejemplo de consulta de agrupación:

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$1.200.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

RUT	TOTAL
85762340	$874.000
183447872	$767.990
204821232	$1.150.990
145375682	$1.622.980
185756203	$1.229.480

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

Uso de la instrucción GROUP BY

RUT	TOTAL
145375682	$1.622.980
185756203	$1.229.480
204821232	$1.150.990
85762340	$874.000
183447872	$767.990

```
SELECT RUT, SUM(TOTAL)
FROM BOLETA
GROUP BY RUT
```

La información de resúmenes debería siempre estar ordenada. SQL ofrece una instrucción para realizar ordenamiento llamada ORDER BY

La instrucción ORDER BY puede ordenar en forma ascendente (ASC) o descendente (DESC) una o múltiples columnas.

Para el ejemplo, ordenaremos la consulta realizada de acuerdo al TOTAL, en forma descendente. Así mostrará primero el mayor TOTAL y luego irá descendiendo.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Puede existir la posibilidad de que al realizar una consulta de agrupamiento, se necesite establecer una condición sobre la agrupación.

Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Puede existir la posibilidad de que al realizar una consulta de agrupamiento, se necesite establecer una condición sobre la agrupación.

RUT	TOTAL
145375682	$1.622.980
185756203	$1.229.480
204821232	$1.150.990
85762340	$874.000
183447872	$767.990

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

RUT	TOTAL
145375682	$1.622.980
185756203	$1.229.480
204821232	$1.150.990
85762340	$874.000
183447872	$767.990

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

RUT	TOTAL
145375682	$1.622.980
185756203	$1.229.480
204821232	$1.150.990
85762340	$874.000
183447872	$767.990

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

Lo más lógico podría ser restringir eso agregando la instrucción WHERE.

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

Lo más lógico podría ser restringir eso agregando la instrucción WHERE.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Consulta Original

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

Lo más lógico podría ser restringir eso agregando la instrucción WHERE.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Consulta Original

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Instrucción HAVING

Por ejemplo, mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

Lo más lógico podría ser restringir eso agregando la instrucción WHERE.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

Consulta Original

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE TOTAL > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE TOTAL > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Estas alternativas pueden ser bastante lógicas, pero en realidad, no son correctas.

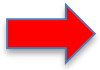
Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE TOTAL > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Estas alternativas pueden ser bastante lógicas, pero en realidad, no son correctas.

Instrucción HAVING



```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE TOTAL > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Estas alternativas pueden ser bastante lógicas, pero en realidad, no son correctas.

La primera consulta arrojará un error al momento de compilar.

Instrucción HAVING

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE SUM(TOTAL) > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
WHERE TOTAL > 1000000
GROUP BY RUT
ORDER BY TOTAL DESC
```

Si bien para algunos esto sería lo lógico, en realidad no es lo correcto.

La primera consulta arrojará un error al momento de compilar.

Y la segunda, arrojará un valor incorrecto a lo que esperamos.

¿Qué hacer?

Las consultas anteriores no funcionan porque están intentando restringir o filtrar sobre datos “agrupados”.

¿Datos agrupados?

BOLETA			
CODIGO_BOLETA	RUT	FECHA	TOTAL
2	145375682	14-08-2019	$110.000
1	145375682	12-07-2019	$388.500
3	145375682	17-08-2019	$81.990
4	145375682	24-08-2019	$3.500
5	204821232	12-08-2019	$370.000
6	204821232	16-08-2019	$370.000
7	183447872	02-09-2019	$260.000
8	85762340	10-09-2019	$17.000
9	183447872	16-09-2019	$110.000
10	145375682	20-09-2019	$560.000
11	85762340	01-10-2019	$7.000
12	204821232	11-10-2019	$280.990
13	185756203	12-10-2019	$1.200.490
14	183447872	05-11-2019	$397.990
15	145375682	08-11-2019	$478.990
16	185756203	12-12-2019	$28.990
17	204821232	12-12-2019	$130.000
18	85762340	21-12-2019	$850.000

Datos en su estado puro

RUT	TOTAL
145375682	$1.622.980
185756203	$1.229.480
204821232	$1.150.990
85762340	$874.000
183447872	$767.990

Datos agrupados

¿Qué hacer?

Las consultas anteriores no funcionan porque están intentando restringir o filtrar sobre datos “agrupados”.

Para estos casos, es cuando se utiliza la instrucción “HAVING”, la cual permite restringir o filtrar sobre información agrupada.

Solución

Mostrar solamente los clientes que han comprado por más del 1.000.000 de pesos.

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
HAVING SUM(TOTAL) > 1000000
ORDER BY TOTAL DESC
```

Aclaraciones

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
ORDER BY TOTAL DESC
```

```
SELECT RUT, SUM(TOTAL) AS TOTAL
FROM BOLETA
GROUP BY RUT
HAVING SUM(TOTAL) > 1000000
ORDER BY TOTAL DESC
```

La instrucción WHERE se utiliza generalmente para filtrar información antes de utilizar la instrucción GROUP BY.

La instrucción HAVING se utiliza para filtrar información después de utilizar la instrucción GROUP BY.

Subconsultas

Las subconsultas es una consulta anidada en una instrucción SELECT, INSERT, UPDATE o DELETE, o bien en otra subconsulta.

Subconsultas

Las subconsultas es una consulta anidada en una instrucción SELECT, INSERT, UPDATE o DELETE, o bien en otra subconsulta.

Las subconsultas se pueden utilizar en cualquier parte en la que se permita una expresión.

Subconsultas

Las subconsultas es una consulta anidada en una instrucción SELECT, INSERT, UPDATE o DELETE, o bien en otra subconsulta.

Las subconsultas se pueden utilizar en cualquier parte en la que se permita una expresión.

La consulta SELECT de una subconsulta se presenta siempre entre paréntesis.

¿Cómo funciona?

Usualmente, una subconsulta se utiliza para dar apoyo a otra consulta cuando se desconoce algún valor.

¿Cómo funciona?

Usualmente, una subconsulta se utiliza para dar apoyo a otra consulta cuando se desconoce algún valor.

Qué quiere decir con “se desconoce algún valor”. Por ejemplo, obtener todos los productos que tengan menos stock que el promedio.

¿Cómo funciona?

Usualmente, una subconsulta se utiliza para dar apoyo a otra consulta cuando se desconoce algún valor.

Qué quiere decir con “se desconoce algún valor”. Por ejemplo, obtener todos los productos que tengan menos stock que el promedio.

Antes de obtener aquellos productos, debemos primero, saber el promedio del stock que existe.

Ejemplo 1:

Obtener los productos que tienen menos stock que el promedio.

PRODUCTO				
CODIGO_PRODUCTO	NOMBRE	PRECIO	CODIGO_CATEGORIA	STOCK
1	NOTEBOOK LENOVO	$380.000	1	20
2	CELULAR MOTOROLA	$110.000	2	15
3	AUDIFONOS MACROTEL	$3.500	3	12
4	NOTEBOOK SAMSUNG	$500.000	1	3
5	MONITOR 17" AOC	$67.990	4	80
6	MONITOR 21" DELL	$81.990	4	20
7	NOTEBOOK HP	$280.990	1	1
8	CELULAR IPHONE 6S	$560.000	2	6
9	CELULAR LG	$450.000	2	7
10	MOUSE BLUETOOTH	$28.990	5	9
11	MONITOR 17" SAMSUNG	$150.000	4	7
12	MONITOR 17" LENOVO	$250.000	4	7
13	MONITOR 17" LG	$125.000	4	4
14	MONITOR 15" LENOVO	$200.000	4	3
15	MOTO G 2013	$130.000	2	45
16	MOTO G 2015	$180.000	2	32
17	MOTO X PLAY	$370.000	2	12
18	IPAD MINI 2	$200.000	6	4
19	IPAD AIR	$260.000	6	2
20	NOTEBOOK MSI	$850.000	1	4
21	MOUSE MICROSOFT	$8.500	5	5

Ejemplo 1:

Primero debemos obtener el promedio del stock de todos los productos.

Ejemplo 1:

Primero debemos obtener el promedio del stock de todos los productos.

Para esto, utilizamos la función AVG.

```
SELECT AVG(STOCK)
FROM PRODUCTO
```

Ejemplo 1:

Primero debemos obtener el promedio del stock de todos los productos.

Para esto, utilizamos la función AVG.

```
SELECT AVG(STOCK)
FROM PRODUCTO
```

Esta consulta retorna el valor: 14.1904, lo que corresponde al promedio del stock de todos los productos.

Ejemplo 1:

De la forma tradicional, podríamos utilizar el resultado de la consulta, “14.19” para obtener los productos que tengan un menor stock a ese valor.

Ejemplo 1:

De la forma tradicional, podríamos utilizar el resultado de la consulta, “14.19” para obtener los productos que tengan un menor stock a ese valor.

```
SELECT CODIGO_PRODUCTO, NOMBRE, STOCK
FROM PRODUCTO
WHERE STOCK < 14.19
```

Ejemplo 1:

De la forma tradicional, podríamos utilizar el resultado de la consulta, “14.19” para obtener los productos que tengan un menor stock a ese valor.

```
SELECT CODIGO_PRODUCTO, NOMBRE, STOCK
FROM PRODUCTO
WHERE STOCK < 14.19
```

Si realizamos la consulta de esta forma, siempre vamos a estar calculando primero el promedio y luego agregándolo a la consulta.

Ejemplo 1:

De la forma tradicional, podríamos utilizar el resultado de la consulta, “14.19” para obtener los productos que tengan un menor stock a ese valor.

```
SELECT CODIGO_PRODUCTO, NOMBRE, STOCK
FROM PRODUCTO
WHERE STOCK < 14.19
```

Si realizamos la consulta de esta forma, siempre vamos a estar calculando primero el promedio y luego agregándolo a la consulta.

Para esto utilizamos la subconsulta.

Ejemplo 1

La sintaxis final para realizar nuestra consulta sería la siguiente:

```
SELECT CODIGO_PRODUCTO, NOMBRE, STOCK
FROM PRODUCTO
WHERE STOCK < (SELECT AVG(STOCK)
               FROM PRODUCTO)
```

Resultado ejemplo 1

CODIGO_PRODUCTO	NOMBRE	STOCK
3	AUDIFONOS MACROTEL	12
4	NOTEBOOK SAMSUNG	3
7	NOTEBOOK HP	1
8	CELULAR IPHONE 6S	6
9	CELULAR LG	7
10	MOUSE BLUETOOTH	9
11	MONITOR 17" SAMSUNG	7
12	MONITOR 17" LENOVO	7
13	MONITOR 17" LG	4
14	MONITOR 15" LENOVO	3
17	MOTO X PLAY	12
18	IPAD MINI 2	4
19	IPAD AIR	2
20	NOTEBOOK MSI	4
21	MOUSE MICROSOFT	5

Sintaxis de la subconsulta

```
SELECT COLUMNAS
FROM TABLA1
WHERE COLUMNA OPERATOR (SELECT COLUMNA
                        FROM TABLAN
                        WHERE COLUMNA OPERATOR VALOR)
```

Ejemplo 2

Supongamos que usted compra el producto con código 21. Este corresponde de acuerdo a nuestros datos, al Mouse Microsoft.

Ejemplo 2

Supongamos que usted compra el producto con código 21. Este corresponde de acuerdo a nuestros datos, al Mouse Microsoft.

Sería interesante saber qué otros productos compraron los clientes cuando compraron el Mouse Microsoft.

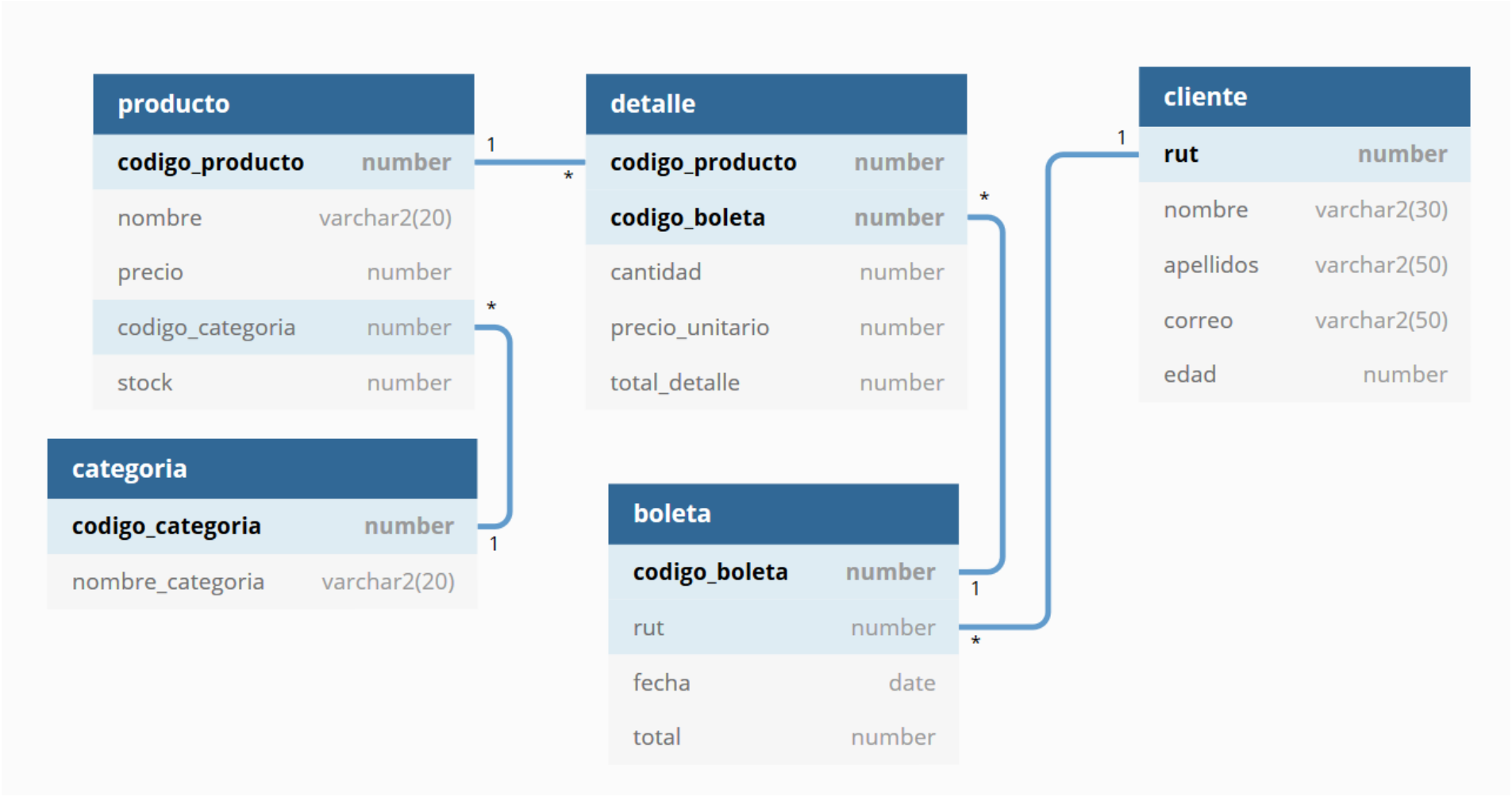
Ejemplo 2

Supongamos que usted compra el producto con código 21. Este corresponde de acuerdo a nuestros datos, al Mouse Microsoft.

Sería interesante saber qué otros productos compraron los clientes cuando compraron el Mouse Microsoft.

Antes de realizar la consulta, analicemos nuestro modelo.

Modelo de datos



*Explicación de la consulta hecha en el laboratorio

Ejemplo 2

Primero, debemos obtener los códigos de boleta donde esté presente el código de producto 21.

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
2	2	1	110000	110000
6	3	1	81990	81990
3	4	1	3500	3500
17	5	1	370000	370000
17	6	1	370000	370000
19	7	1	260000	260000
21	8	2	8500	17000
2	9	1	110000	110000
8	10	1	560000	560000
3	11	2	3500	7000
7	12	1	280990	280990
19	13	1	260000	260000
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990
2	14	3	110000	330000
5	14	1	67990	67990
10	15	1	28990	28990
9	15	1	450000	450000
10	16	1	28990	28990
15	17	1	130000	130000
20	18	1	850000	850000

Ejemplo 2

Primero, debemos obtener los códigos de boleta donde esté presente el código de producto 21.

```
SELECT CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_PRODUCTO = 21
```

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
2	2	1	110000	110000
6	3	1	81990	81990
3	4	1	3500	3500
17	5	1	370000	370000
17	6	1	370000	370000
19	7	1	260000	260000
21	8	2	8500	17000
2	9	1	110000	110000
8	10	1	560000	560000
3	11	2	3500	7000
7	12	1	280990	280990
19	13	1	260000	260000
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990
2	14	3	110000	330000
5	14	1	67990	67990
10	15	1	28990	28990
9	15	1	450000	450000
10	16	1	28990	28990
15	17	1	130000	130000
20	18	1	850000	850000

Ejemplo 2

Primero, debemos obtener los códigos de boleta donde esté presente el código de producto 21.

```
SELECT CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_PRODUCTO = 21
```

Luego, obtener los productos que están en esas boletas.

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
21	8	2	8500	17000
19	13	1	260000	260000
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990

Ejemplo 2

Primero, debemos obtener los códigos de boleta donde esté presente el código de producto 21.

```
SELECT CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_PRODUCTO = 21
```

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
21	8	2	8500	17000
19	13	1	260000	260000
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990

Luego, obtener los productos que están en esas boletas.

```
SELECT CODIGO_PRODUCTO, CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_BOLETA IN (SELECT CODIGO_BOLETA
                        FROM DETALLE
                        WHERE CODIGO_PRODUCTO = 21)
```

Ejemplo 2

Primero, debemos obtener los códigos de boleta donde esté presente el código de producto 21.

```
SELECT CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_PRODUCTO = 21
```

DETALLE				
CODIGO_PRODUCTO	CODIGO_BOLETA	CANTIDAD	PRECIO_UNITARIO	TOTAL
1	1	1	380000	380000
21	1	1	8500	8500
21	8	2	8500	17000
19	13	1	260000	260000
20	13	1	850000	850000
21	13	1	8500	8500
6	13	1	81990	81990

Luego, obtener los productos que están en esas boletas.

```
SELECT CODIGO_PRODUCTO, CODIGO_BOLETA
FROM DETALLE
WHERE CODIGO_BOLETA IN (SELECT CODIGO_BOLETA
                        FROM DETALLE
                        WHERE CODIGO_PRODUCTO = 21)
AND CODIGO_PRODUCTO <> 21
```

Y por último, indicar que muestre solo los productos que son distintos al producto con código igual a 21.

Resultado ejemplo 2

CODIGO_PRODUCTO	CODIGO_BOLETA
1	1
19	13
20	13
6	13

Observaciones de la subconsulta

- Cuando se sabe que la subconsulta retorna un valor exacto se puede utilizar “=“.
- Cuando una subconsulta retorna muchos elementos debe utilizarse la instrucción “IN”. De no usarla, aparecerá un error.
- La subconsulta no necesariamente debe utilizarse en conjunto con la instrucción WHERE.
- Hay consultas que pueden hacerse con JOIN y subconsultas.

Soluciones para las consultas

- Consulta 1

	CLIENTE
1	CLIENTE RUT: 145375682 - ALEJANDRO SEPULVEDA
2	CLIENTE RUT: 85762340 - CAMILA FERNANDEZ

- Consulta 2 y 3

	RUT	NOMBRE
1	183447872	Javiera
2	204821232	Francisca
3	145375682	Alejandro
4	85762340	Camila

Soluciones para las consultas

- Consulta 4

	RUT	NOMBRE	TOTAL_DE_VENTAS
1	145375682	ALEJANDRO	1622980
2	185756203	FELIPE	1229480
3	204821232	FRANCISCA	1150990

Soluciones para las consultas

- Consulta 5

	RUT	NOMBRE_COMPLETO	FECHA	TOTAL_DE_VENTAS
1	145375682	ALEJANDRO SEPULVEDA	JULIO	388500
2	145375682	ALEJANDRO SEPULVEDA	AGOSTO	195490
3	204821232	FRANCISCA PEREZ	AGOSTO	740000
4	145375682	ALEJANDRO SEPULVEDA	SEPTIEMBRE	560000
5	85762340	CAMILA FERNANDEZ	SEPTIEMBRE	17000
6	183447872	JAVIERA FAUNDEZ	SEPTIEMBRE	370000
7	85762340	CAMILA FERNANDEZ	OCTUBRE	7000
8	185756203	FELIPE TAPIA	OCTUBRE	1200490
9	204821232	FRANCISCA PEREZ	OCTUBRE	280990
10	145375682	ALEJANDRO SEPULVEDA	NOVIEMBRE	478990
11	183447872	JAVIERA FAUNDEZ	NOVIEMBRE	397990
12	85762340	CAMILA FERNANDEZ	DICIEMBRE	850000
13	185756203	FELIPE TAPIA	DICIEMBRE	28990
14	204821232	FRANCISCA PEREZ	DICIEMBRE	130000

Soluciones para las consultas

- Consulta 6

	NOMBRE_MES	CANTIDAD_PRODUCTOS_VENDIDOS
1	JULIO	2
2	AGOSTO	5
3	SEPTIEMBRE	5
4	OCTUBRE	7
5	NOVIEMBRE	6
6	DICIEMBRE	3

- Consulta 7

	RUT	NOMBRE_COMPLETO	NOMBRE_MES	TOTAL_DE_VENTAS
1	145375682	ALEJANDRO SEPULVEDA	JULIO	388500
2	204821232	FRANCISCA PEREZ	AGOSTO	370000
3	204821232	FRANCISCA PEREZ	AGOSTO	370000
4	145375682	ALEJANDRO SEPULVEDA	SEPTIEMBRE	560000
5	185756203	FELIPE TAPIA	OCTUBRE	1200490
6	183447872	JAVIERA FAUNDEZ	NOVIEMBRE	397990
7	145375682	ALEJANDRO SEPULVEDA	NOVIEMBRE	478990
8	85762340	CAMILA FERNANDEZ	DICIEMBRE	850000

Soluciones para las consultas

- Consulta 8 y 9

	CODIGO_PRODUCTO	NOMBRE
1	3	AUDIFONOS MACROTEL
2	8	CELULAR IPHONE 6S
3	9	CELULAR LG
4	10	MOUSE BLUETOOTH

Soluciones para las consultas

- Consulta 10

	CODIGO_PRODUCTO	NOMBRE
1	1	NOTEBOOK LENOVO
2	2	CELULAR MOTOROLA
3	3	AUDIFONOS MACROTEL
4	6	MONITOR 21" DELL
5	8	CELULAR IPHONE 6S
6	9	CELULAR LG
7	10	MOUSE BLUETOOTH
8	21	MOUSE MICROSOFT

Soluciones para las consultas

- Consulta 11

	CODIGO_PRODUCTO	NOMBRE	CANTIDAD
1	1	NOTEBOOK LENOVO	1
2	2	CELULAR MOTOROLA	5
3	17	MOTO X PLAY	2
4	15	MOTO G 2013	1

UNIVERSIDAD CATOLICA DEL MAULE