



Tecnicatura Universitaria  
en Programación

## BASES DE DATOS I

Unidad Temática N°2:  
Subconsultas

Teórico  
1° Año – 2° Cuatrimestre



## Índice

Subconsultas.....	2
Subconsultas en la cláusula WHERE.....	2
Test de comparación subconsulta.....	2
Test de pertenencia al conjunto para subconsultas. ....	3
Test de existencia .....	4
Test cuantificados .....	5
Subconsultas en la cláusula HAVING .....	6
Otras Subconsultas .....	7
Subconsultas en el UPDATE y DELETE .....	9
BIBLIOGRAFÍA .....	10

## Subconsultas

Una *subconsulta* es una consulta que puede aparecer dentro de la cláusula **WHERE** o **HAVING** de otra sentencia SQL. La subconsulta debe ir encerrada entre paréntesis, y tiene el formato de una sentencia **SELECT**, pero existen diferencias entre una subconsulta y una sentencia **SELECT** real.

Una subconsulta:

- Debe producir una **única** columna de datos como resultados. Esto significa que una subconsulta siempre tiene un único elemento de selección en su cláusula **SELECT**.
- No puede ser **UNION** de varias sentencias **SELECT** diferentes.
- La cláusula **ORDER BY** no puede ser especificada en una subconsulta.
- Los nombres de columna que aparecen en una subconsulta pueden referirse a columnas de tablas en la consulta principal, esto se denomina **referencias externas**.

### Subconsultas en la cláusula WHERE

Las subconsultas suelen ser utilizadas en la cláusula **WHERE** de una sentencia SQL, en este caso funciona como parte del proceso de selección de filas.

Las condiciones de búsqueda en subconsultas son las que se describen:

### Test de comparación subconsulta

Es una forma modificada del test de comparación simple, utiliza los operadores de comparación: **=**, **<**, **>**, **<=**, **>=**. Compara el valor de una expresión con el valor único producido por una subconsulta, y devuelve un resultado **TRUE** si la comparación es cierta.

Por ejemplo, si se necesita el listado de artículos cuyos precios son menores al promedio, se podría utilizar una subconsulta que calcule el precio promedio de los artículos a la venta; luego en la consulta principal listar los artículos cuyo precio unitario sea menor al resultado de dicha subconsulta:

```
Select cod_articulo, descripcion, pre_unitario
from articulos
where pre_unitario < (Select AVG(pre_unitario) from articulos)
```

Results

Messages

	cod_articulo	descripcion	pre_unitario
1	1	Lápiz Evolution HB2 * 4 u	15.50
2	2	Papel p/forrar Fantas?a * 10 u	22.30
3	3	Papel p/forrar Ara?a * 10 u	25.50
4	4	Lápices Color cortos * 12 u	20.50
5	5	Fibras cortas * 6	25.30
6	6	Lápices Color largos * 12 u	27.90
7	7	Separadores tamaño rivadavia * 6 u	15.70
8	8	Carpeta fibra negra - Tamaño rivadavia	12.90
9	9	Carpeta Fantasía - Tamaño Rivadavia	15.90
10	10	Adhesivo sintético 30 gr	6.00

Tabla 1: Elaboración propia

La subconsulta especificada en este test debe producir una *única fila* de resultados, es decir un único valor que en este caso sería el promedio de los precios.

### Test de pertenencia al conjunto para subconsultas.

Es una forma modificada del test de pertenencia a conjunto simple cuyo operador es **IN**; compara un único valor de datos con una columna de valores producida por una subconsulta y devuelve un resultado **TRUE** si el valor coincide con al menos uno de los valores de la columna.

Listar los datos de los clientes que compraron este año:

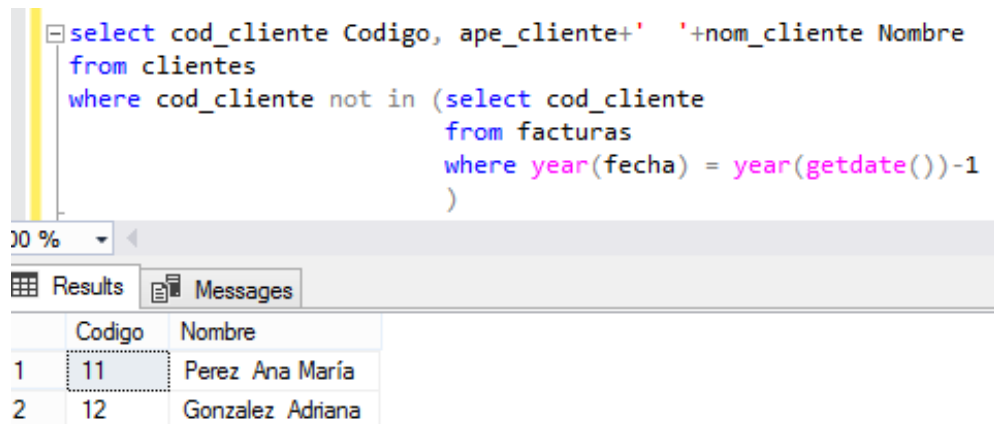
```
SELECT *
FROM clientes
WHERE cod_cliente IN (SELECT cod_cliente
FROM facturas
WHERE YEAR(fecha) = YEAR(GETDATE()))
```

Donde la consulta principal listaría todos los datos de los clientes cuyo cod\_cliente esté dentro del conjunto de cod\_cliente que surjan de la subconsulta. La subconsulta generará un listado de cod\_cliente desde la tabla facturas cuyo año de la fecha sea igual al año de la fecha actual. La subconsulta no muestra nada solo es utilizada para verificar la condición de búsqueda del **WHERE**.

Otro ejemplo podría ser el listado de clientes que no vinieron el año pasado

En este caso la subconsulta listaría todos los códigos de clientes que SI compraron el año pasado. En el **WHERE** de la consulta principal se va a analizar

fila por fila si el código de cliente que se quiere listar no pertenece al conjunto de valores devueltos por la subconsulta.



```

select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes
where cod_cliente not in (select cod_cliente
                          from facturas
                          where year(fecha) = year(getdate())-1
                          )

```

	Codigo	Nombre
1	11	Perez Ana María
2	12	Gonzalez Adriana

Tabla 2: Elaboración propia

### Test de existencia

Comprueba si una subconsulta produce alguna fila de resultados; el operador utilizado en este test es el EXISTS; no hay test de comparación simple que se asemeje al test de existencia, solamente se utiliza con subconsultas.

El ejemplo anterior, listado de los datos de los clientes que compraron este año, se podría resolver con este test:

```

SELECT  cod_cliente  Código,  ape_cliente  + ' ' + nom_cliente
Nombre
FROM    clientes c
WHERE   EXISTS (SELECT cod_cliente
                FROM facturas f
                WHERE c.cod_cliente = f.cod_cliente
                    AND YEAR(fecha) = YEAR(GETDATE())
                )

```

En este caso la subconsulta devolvería filas de resultado si el cliente que se quiere listar en la consulta principal (recuerde que se realiza el análisis fila a fila) posee algún registro en la tabla facturas correspondiente a la fecha con año igual al año de la fecha actual.

Observe el *c.cod\_cliente* de la subconsulta es una **referencia externa** dado que se trata de un campo utilizado en la subconsulta y que pertenece a la tabla de la consulta principal.

Para listar los datos de los clientes que no compraron el año pasado utilizando este test sería:

```

SELECT cod_cliente Codigo, ape_cliente + ' ' + nom_cliente Nombre
FROM clientes c
WHERE NOT EXISTS (SELECT cod_cliente
                  FROM facturas f
                  WHERE c.cod_cliente = f.cod_cliente
                  AND YEAR(fecha) = YEAR(GETDATE())-1
                  )

```

En este otro ejemplo, la subconsulta devolvería filas de resultado si el cliente que se quiere listar en la consulta principal posee algún registro en la tabla facturas correspondiente a la fecha con año igual al año anterior a la fecha actual. Si no existen filas de resultados el test **NOT EXISTS** devuelve verdadero y el registro se muestra en la tabla final.

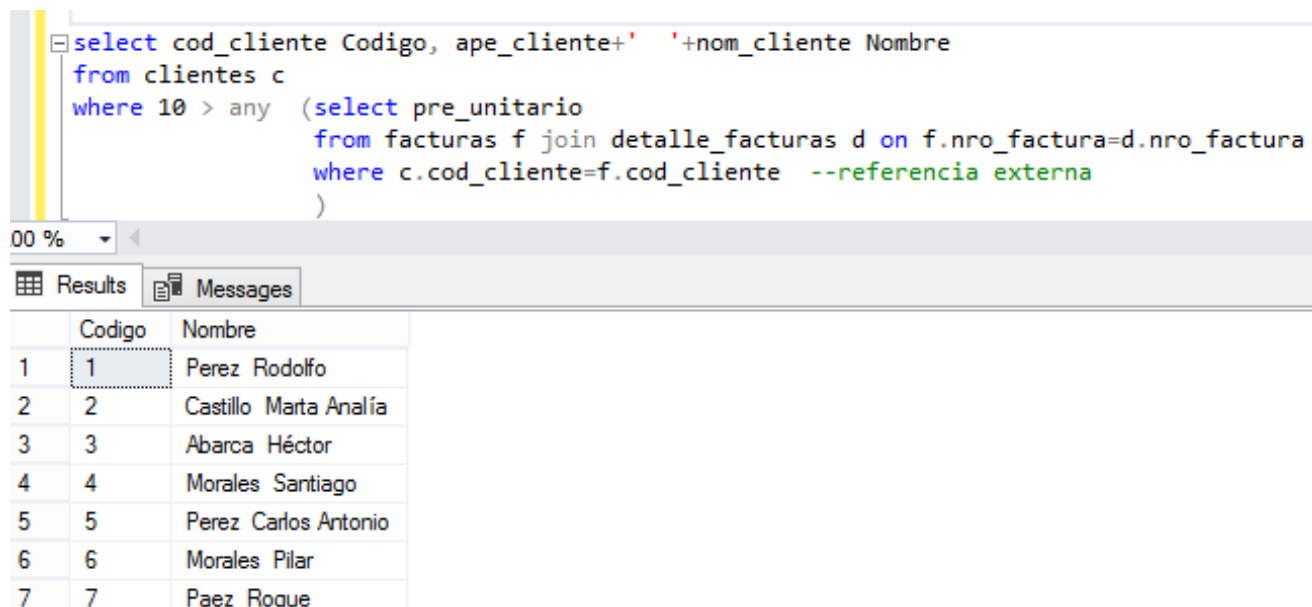
### Test cuantificados

Estos test extienden el test de pertenencia al conjunto **IN**, a otros operadores de comparación como **=**, **<**, **<=**, **>**, **>=** y **<>**.

#### El test ANY:

Se utiliza con uno de estos seis operadores de comparación. Para efectuar el test, SQL utiliza el operador de comparación especificado para comparar el valor del test con *cada* valor de datos en la columna, uno cada vez. Si **ALGUNAS** de las comparaciones individuales producen un resultado **TRUE**, el test **ANY** devuelve un resultado **TRUE**.

Por ejemplo, si se quiere listar los clientes que alguna vez compraron un producto menor a \$ 10.-



```

select cod_cliente Codigo, ape_cliente+' ' +nom_cliente Nombre
from clientes c
where 10 > any (select pre_unitario
                from facturas f join detalle_facturas d on f.nro_factura=d.nro_factura
                where c.cod_cliente=f.cod_cliente --referencia externa
                )

```

	Codigo	Nombre
1	1	Perez Rodolfo
2	2	Castillo Marta Analía
3	3	Abarca Héctor
4	4	Morales Santiago
5	5	Perez Carlos Antonio
6	6	Morales Pilar
7	7	Paez Roque

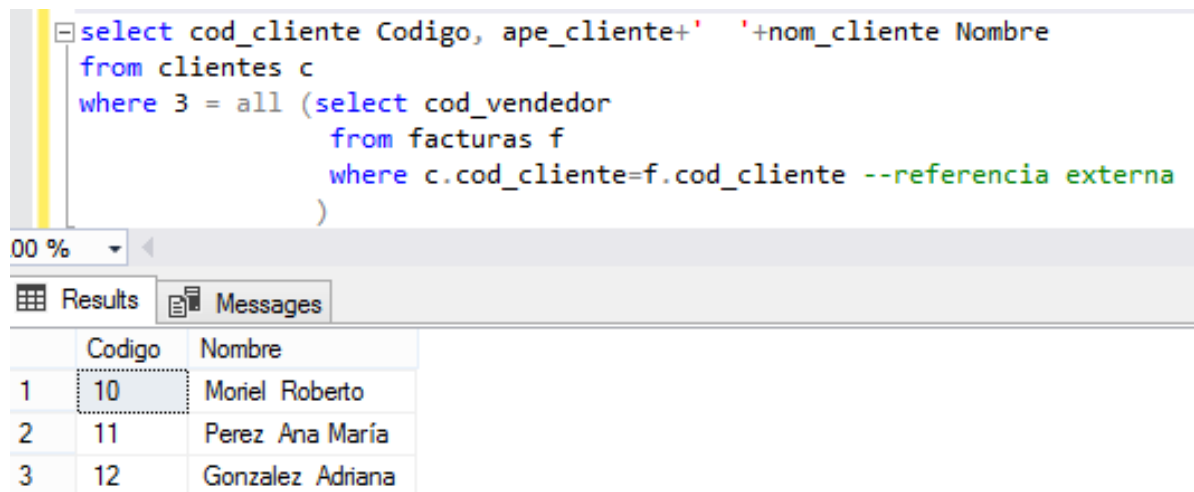
Tabla 2: Elaboración propia

La subconsulta de este ejemplo lista todos los precios unitarios de los artículos vendidos al cliente que la consulta principal quiere listar; si alguno de éstos precios es menor a 10 entonces el registro de ese cliente forma parte de las filas de resultado.

#### El test ALL:

Se utiliza también con uno de los seis operadores de comparación. Para efectuar el test, SQL utiliza el operador de comparación especificado para comparar el valor de test con todos y *cada* uno de los valores de datos de la columna. Si **TODAS** las comparaciones individuales producen un resultado **TRUE**, el test **ALL** devuelve un resultado **TRUE**.

Ejemplo: se quiere listar los clientes que siempre fueron atendidos por el vendedor 3:



```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes c
where 3 = all (select cod_vendedor
               from facturas f
               where c.cod_cliente=f.cod_cliente --referencia externa
            )
```

00 %

Results Messages

	Codigo	Nombre
1	10	Moriel Roberto
2	11	Perez Ana María
3	12	Gonzalez Adriana

Tabla 3: Elaboración propia

En este caso, la subconsulta emite un listado de códigos de vendedores cuyas ventas hayan sido realizadas al cliente que se quiere listar en la consulta principal (referencia externa).

El **WHERE** verificará si todos los códigos de vendedores listados en la subconsulta son iguales a 3.

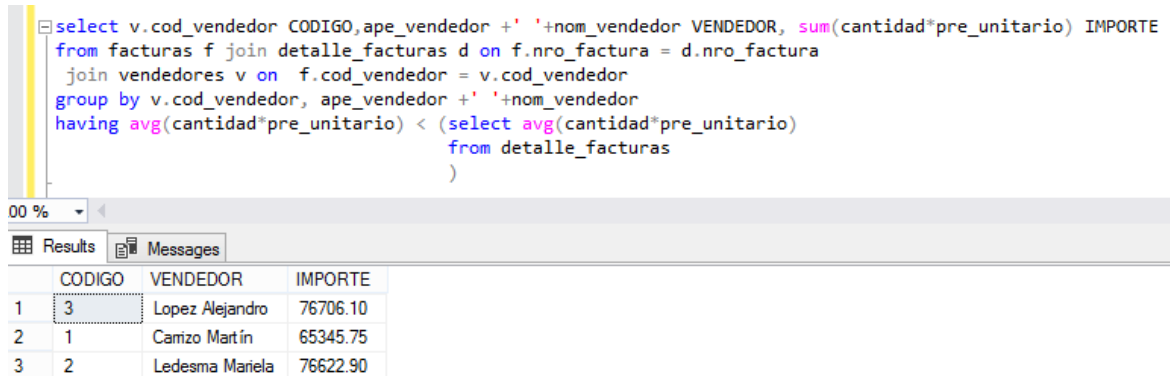
#### Subconsultas en la cláusula HAVING

También pueden utilizarse subconsultas en la cláusula **HAVING** y funciona como parte de la selección de un grupo de filas efectuada por esta cláusula.

Si se quisiera listar los vendedores cuyas ventas totales sea superior al promedio general de ventas.



Se debería listar, en la consulta principal, los vendedores con sus ventas totales agrupado por vendedor (código y nombre) y como condición en el **HAVING** es que su venta total sea superior al resultado de una subconsulta; esta subconsulta deberá calcular el promedio general de ventas.



```

select v.cod_vendedor CODIGO,ape_vendedor +' '+nom_vendedor VENDEDOR, sum(cantidad*pre_unitario) IMPORTE
from facturas f join detalle_facturas d on f.nro_factura = d.nro_factura
join vendedores v on f.cod_vendedor = v.cod_vendedor
group by v.cod_vendedor, ape_vendedor +' '+nom_vendedor
having avg(cantidad*pre_unitario) < (select avg(cantidad*pre_unitario)
from detalle_facturas
)

```

	CODIGO	VENDEDOR	IMPORTE
1	3	Lopez Alejandro	76706.10
2	1	Camizo Martín	65345.75
3	2	Ledesma Mariela	76622.90

Tabla 4: Elaboración propia

Hay que tener en cuenta que las referencias externas en las subconsultas en el HAVING deben ser parte de los campos de agrupación del GROUP BY

### Otras Subconsultas

En apartados anteriores se explicó el tema de subconsultas en el **WHERE** y en el **HAVING**, pero también pueden:

- Haber subconsultas dentro de subconsultas, se admiten hasta 32 niveles de anidación.
- Ocupar el lugar de una expresión, siempre que devuelvan un solo valor o una lista de valores.
- Encontrarse como columnas en la lista de selección
- Retornar un conjunto de registros de varios campos en lugar de una tabla (en el FROM) o para obtener el mismo resultado que una combinación (JOIN).

Como ejemplo de una subconsulta como parte de una expresión, se quiere listar el precio de los artículos y la diferencia de éste con el precio del artículo más caro:

```

SELECT descripcion,
       precio_unit,
       (SELECT MAX(pre_unitario) FROM articulos)- pre_unitario 'DIFERENCIA'
FROM articulos

```



Otro caso sería cuando se usa una subconsulta en una columna, por ejemplo: listar el precio actual de los artículos y el precio histórico vendido más barato:

```
SELECT descripcion,
       pre_unitario 'precio actual',
       (SELECT MIN(pre_unitario)
        FROM detalle_facturas d
        WHERE d.cod_articulo = a.cod_articulo) 'PRECIO HISTORICO'
FROM articulos a
```

Observe que en ambos ejemplos las subconsultas devuelven un solo registro en caso contrario daría error.

Por último, se quiere emitir un listado de las facturas del año en curso detallando número de factura, cliente, fecha y total de la misma

```
SELECT nro_factura, fecha, apellido + ' ' + nombre 'CLIENTE',
       (SELECT SUM(d.precio_unit*cantidad)
        FROM detalle_facturas d
        WHERE f.nro_factura = d.nro_factura) 'TOTAL'
FROM facturas f JOIN clientes c ON c.cod_cliente =
f.cod_cliente
WHERE YEAR(fecha) = YEAR(GETDATE())
```

Se pueden emplear subconsultas que retornen un conjunto de registros de varios campos en lugar de una tabla. Se la denomina tabla derivada y se coloca en la cláusula **"FROM"** para que la use un **"SELECT"** externo. La tabla derivada debe ir entre paréntesis y tener un alias para poder referenciarla.

Supongamos el ejemplo anterior, donde se quiere emitir un listado de las facturas del año en curso detallando número de factura, cliente, fecha y total de la misma se podría resolver de la siguiente manera:

```
SELECT f.nro_factura, fecha, apellido + ' ' + nombre 'Cliente',
       f2.total
FROM facturas f JOIN clientes c ON c.cod_cliente = f.cod_cliente
JOIN (SELECT nro_factura, SUM(precio_unit*cantidad) 'total'
      FROM detalle_facturas
      GROUP BY nro_factura) AS f2 ON f2.nro_factura =
f.nro_factura
WHERE YEAR(fecha)= YEAR(GETDATE())
```

## Subconsultas en el UPDATE y DELETE

En la cláusula **WHERE** de las sentencias **UPDATE** y **DELETE** pueden ir todas las condiciones de búsquedas que ya se han visto en el **WHERE** de la sentencia **SELECT**, incluso subconsultas.

Por ejemplo, si se quiere aumentar un 5% los precios de los artículos cuyos precios son inferiores al promedio general.

```
UPDATE articulos
  SET pre_unitario = pre_unitario*1.05
  WHERE pre_unitario < (SELECT AVG(pre_unitario)
                        FROM articulos)
```

Otro ejemplo: se quiere eliminar los clientes que no vinieron nunca.

```
DELETE clientes
  WHERE cod_client NOT IN (SELECT cod_cliente
                           FROM facturas)
```

## BIBLIOGRAFÍA

Gorman K., Hirt A., Noderer D., Rowland-Jones J., Sirpal A., Ryan D. & Woody B (2019) Introducing Microsoft SQL Server 2019. Reliability, scalability, and security both on premises and in the cloud. Packt Publishing Ltd. Birmingham UK

Microsoft (2021) SQL Server technical documentation. Disponible en: <https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15>

Opel, A. & Sheldon, R. (2010). Fundamentos de SQL. Madrid. Editorial Mc Graw Hill

Varga S., Cherry D., D'Antoni J. (2016). Introducing Microsoft SQL Server 2016 Mission-Critical Applications, Deeper Insights, Hyperscale Cloud. Washington. Microsoft Press



### Atribución-No Comercial-Sin Derivadas

Se permite descargar esta obra y compartirla, siempre y cuando no sea modificado y/o alterado su contenido, ni se comercialice. Referenciarlo de la siguiente manera: Universidad Tecnológica Nacional Facultad Regional Córdoba (S/D). Material para la Tecnicatura Universitaria en Programación, modalidad virtual, Córdoba, Argentina.