

Subconsultas.

8.1 Introducción.....	1
8.2. Tipos de subconsultas.....	2
8.2.1. Subconsultas en la cláusula WHERE.....	3
8.2.2. Subconsultas en la cláusula HAVING.....	3
8.2.3. Subconsultas en la cláusula FROM.....	4
8.2.4. Subconsultas en la cláusula SELECT.....	5
8.3. Operadores usados en subconsultas.....	6
8.3.1. Operadores básicos de comparación.....	6
8.3.2. Subconsultas con ALL y ANY.....	6
8.3.3. Subconsultas con IN y NOT IN.....	7
8.3.4. Subconsultas con EXISTS y NOT EXISTS.....	8
8.4. Errores comunes.....	9
3.1 Número incorrecto de columnas en la subconsulta.....	9
3.2 Número incorrecto de filas en la subconsulta.....	9

8.1 Introducción.

Una subconsulta es una consulta anidada dentro de otra consulta.

Debe tener en cuenta que no existe una única solución para resolver una consulta en SQL. En esta unidad vamos a estudiar cómo podemos resolverla utilizando subconsultas.

Para realizar los ejemplos utilizaremos las siguientes bases de datos, cada una con su correspondiente estructura:

- **tiendadb.**

```
mysql> DESC producto;
```

Field	Type	Null	Key	Default	Extra
codigo	int(10) unsigned	NO	PRI	NULL	auto_increment
nombre	varchar(100)	NO		NULL	
precio	double	NO		NULL	
codigo_fabricante	int(10) unsigned	NO	MUL	NULL	

4 rows in set (0.01 sec)

```
mysql> DESC fabricante;
```

Field	Type	Null	Key	Default	Extra
codigo	int(10) unsigned	NO	PRI	NULL	auto_increment
nombre	varchar(100)	NO		NULL	

2 rows in set (0.00 sec)

```
Database changed
mysql> select * from producto;
```

codigo	nombre	precio	codigo_fabricante
1	Disco duro SATA3 1TB	91.99	5
2	Memoria RAM DDR4 8GB	125	6
3	Disco SSD 1 TB	155.99	4
4	GeForce GTX 1050Ti	190	7
5	GeForce GTX 1080 Xtreme	760	6
6	Monitor 24 LED Full HD	207	1
7	Monitor 27 LED Full HD	250.99	1
8	Portátil Yoga 520	564	2
9	Portátil Ideapad 320	449	2
10	Impresora HP Deskjet 3720	64.99	3
11	Impresora HP Laserjet Pro M26nw	185	3
12	Portátil i7	680	4
13	Disco SSD 480 GB	87	8
14	Disco SSD 1TB	175	16
15	Disco SSD 256 GB	50	16

15 rows in set (0.00 sec)

```
mysql> select * from fabricante;
```

codigo	nombre
1	Asus
2	Lenovo
3	Hewlett-Packard
4	Samsung
5	Seagate
6	Crucial
7	Gigabyte
8	Apple
9	Xiaomi
11	IBM
12	Acer
16	Dell
30	Huawei

13 rows in set (0.00 sec)

- **empleadodb:**

```
mysql> desc empleado;
```

Field	Type	Null	Key	Default	Extra
codigo	int(11)	NO	PRI	NUL	auto_increment
nif	varchar(9)	NO		NUL	
nombre	varchar(30)	NO		NUL	
apellido1	varchar(30)	NO		NUL	
apellido2	varchar(30)	YES		NUL	
codigo_departamento	int(11)	YES	MUL	NUL	
codigo_jefe	int(11)	YES		NUL	

7 rows in set (0.01 sec)

```
mysql> desc departamento;
```

Field	Type	Null	Key	Default	Extra
codigo	int(11)	NO	PRI	NUL	auto_increment
nombre	varchar(30)	NO		NUL	
presupuesto	int(11)	YES		0	
gastos	int(11)	YES		1500	

4 rows in set (0.00 sec)

```
mysql> select * from empleado;
```

codigo	nif	nombre	apellido1	apellido2	codigo_departamento	codigo_jefe
1	20111222x	Julio	Romero	de Torres	1	3
2	20111333y	Antonio	Reyna	Manescau	6	4
3	20111444z	Juana	de Arcos	de Arcos	1	NUL
4	20111111a	Ada	Bayron	NUL	6	NUL
5	20222111a	Jimena	Burgos	Madrid	2	NUL
6	20222555b	Remedios	Varo	Sánchez	3	8
7	20222666c	Frida	Kalo	NUL	3	8
8	20222666c	Antonio	Candela	Heredia	3	NUL
9	20222777r	Maria	Triana	Fuentes	NUL	NUL
10	20222888h	Fernano	Fernan	Gómez	NUL	NUL
11	33000111is	Pepe	Ruiz	Santana	2	5
12	33000222d	David	Ruiz	Santana	3	3
13	33000333e	David	Ruiz	Santana	1	3
14	33000444r	Pepe	Ruiz	Ruiz	1	3
15	33888444w	Luis	Varí	NUL	6	4

15 rows in set (0.00 sec)

```
mysql> select * from departamento;
```

codigo	nombre	presupuesto	gastos
1	Sistemas	2000	3500
2	Desarrollo	2000	1500
3	Recursos Humanos	4000	3500
4	Ventas	4000	3500
5	Contabilidad	2000	1500
6	I+D	4000	7732

6 rows in set (0.00 sec)

- **ventasdb.**

```
mysql> desc cliente;
```

Field	Type	Null	Key
id	int(10) unsigned	NO	PRI
nombre	varchar(100)	NO	
apellido1	varchar(100)	NO	
apellido2	varchar(100)	YES	
ciudad	varchar(100)	YES	
categoría	int(10) unsigned	YES	

6 rows in set (0.01 sec)

```
mysql> desc pedido;
```

Field	Type	Null	Key
id	int(10) unsigned	NO	PRI
total	double	NO	
fecha	date	YES	
id_cliente	int(10) unsigned	NO	MUL
id_comercial	int(10) unsigned	NO	MUL

5 rows in set (0.00 sec)

```
mysql> desc comercial;
```

Field	Type	Null	Key
id	int(10) unsigned	NO	PRI
nombre	varchar(100)	NO	
apellido1	varchar(100)	NO	
apellido2	varchar(100)	YES	
comision	float	YES	

5 rows in set (0.00 sec)

```
mysql> select * from cliente;
```

id	nombre	apellido1	apellido2	ciudad
1	Aarón	Rivero	Gómez	Almería
2	Adela	Salas	Granada	Sevilla
3	Adolfo	Rubio	Flores	Sevilla
4	Adrián	Suárez	NUL	Jaén
5	Marcos	Loyola	Méndez	Almería
6	María	Santana	Moreno	Cádiz
7	Pilar	Ruiz	NUL	Sevilla
8	Pepe	Ruiz	Santana	Huelva
9	Guillermo	López	Gómez	Granada
10	Daniel	Santana	Loyola	Sevilla

10 rows in set (0.00 sec)

```
mysql> select * from pedido;
```

id	total	fecha	id_cliente	id_comercial
1	150.5	2017-10-05	5	2
2	270.65	2016-09-10	1	1
3	65.26	2017-10-05	2	1
4	110.5	2016-08-17	8	3
5	948.5	2017-09-10	5	2
6	2400.6	2016-07-27	7	1
7	5760	2015-09-10	2	1
8	1983.43	2017-10-10	4	6
9	2480.4	2016-10-10	8	3
10	250.45	2015-06-27	8	2
11	75.29	2016-08-17	3	2
12	3045.6	2017-04-25	2	1
13	545.75	2019-01-25	6	1
14	145.82	2017-02-02	6	1
15	370.85	2019-03-11	1	5
16	2389.23	2019-03-11	1	5

16 rows in set (0.00 sec)

```
mysql> select * from comercial;
```

id	nombre	apellido1	apellido2
1	Daniel	Sáez	Vega
2	Juan	Gómez	López
3	Diego	Flores	Salas
4	Marta	Herrera	Gil
5	Antonio	Carretero	Ortega
6	Manuel	Dominguez	Hernández
7	Antonio	Vega	Hernández
8	Alfredo	Ruiz	Flores

8 rows in set (0.00 sec)

8.2. Tipos de subconsultas

El estándar SQL define tres tipos de subconsultas:

- **Subconsultas de tabla.** Son aquellas que devuelve una o varias columnas. En cero o varias filas.
- **Subconsultas de fila.** Son aquellas que devuelven más de una columna pero una única fila.
- **Subconsultas escalares.** Son aquellas que devuelven una columna y una fila. En definitiva un valor, que puede ser numérico, texto, fecha,..

Según en la cláusula en la que aparezcan:

- Subconsultas en WHERE.
- Subconsultas en HAVING.
- Subconsultas en FROM.
- Subconsultas en SELECT.

8.2.1. Subconsultas en la cláusula WHERE

Se trata de consultas que requieren hacer otra consulta en la condición de WHERE. Por ejemplo, suponga que queremos conocer el nombre del producto que tiene el mayor precio. En este caso podríamos realizar una primera consulta para buscar cuál es el valor del precio máximo. Con ese resultado, (que sería de una subconsulta escalar, pues nos devuelve un valor) se consultaría el nombre del producto cuyo precio coincide con ese valor. La consulta sería la siguiente:

```
mysql>SELECT MAX(precio) FROM producto;
```

```
mysql>SELECT nombre FROM producto
      WHERE precio=(SELECT MAX(precio) FROM producto);
```

```
mysql> SELECT nombre FROM producto WHERE precio =
      (SELECT MAX(precio) FROM producto);
+-----+
| nombre                |
+-----+
| GeForce GTX 1080 Xtreme |
+-----+
1 row in set (0.00 sec)
```

En este caso sólo hay un nivel de anidamiento entre consultas pero pueden existir varios niveles de anidamiento.

8.2.2. Subconsultas en la cláusula HAVING

HAVING es una cláusula asociada a las consultas resumen o de agrupamiento. Luego la condición de HAVING comparará una función resumen con el resultado de la subconsulta.

Ejemplo: Devuelve un listado con todos los nombres de los fabricantes que proporcionan el mismo número de productos que el fabricante “Asus”.

La subconsulta ofrecerá un escalar: Número de productos de Asus. Al estar el nombre del fabricante en otra tabla es necesario hacer un join de producto con fabricante.

```
mysql> SELECT COUNT(*) FROM producto AS p join fabricante AS f
      ON p.codigo_fabricante=f.codigo WHERE f.nombre=”Asus”;
```

```
mysql> SELECT f.nombre, COUNT(p.codigo) AS cantidad
      FROM producto AS p JOIN fabricante AS f ON p.codigo_fabricante=f.codigo
      GROUP BY f.codigo HAVING cantidad=
      (SELECT COUNT(*) FROM producto AS p join fabricante AS f
      ON p.codigo_fabricante=f.codigo WHERE f.nombre=”Asus”);
```

En este caso COUNT(*) debe contar cuantos productos hay del fabricante Asus. Como se hace un JOIN de producto y fabricante se puede poner en lugar de “*” cualquier atributo de producto o fabricante que no sea NULL. Preferible “código” pues nos garantiza que no es NULL.

```
mysql> mysql> SELECT COUNT(*) FROM producto AS p JOIN fabricante AS f ON p.codigo_fabricante=f.codigo WHERE f.nombre="Asus";
+-----+
| COUNT(*) |
+-----+
|         2 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(f.codigo) FROM producto AS p JOIN fabricante AS f ON p.codigo_fabricante=f.codigo WHERE f.nombre="Asus";
+-----+
| COUNT(f.codigo) |
+-----+
|         2 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(p.codigo) FROM producto AS p JOIN fabricante AS f ON p.codigo_fabricante=f.codigo WHERE f.nombre="Asus";
+-----+
| COUNT(p.codigo) |
+-----+
|         2 |
+-----+
1 row in set (0.00 sec)
```

Esa consulta ofrece el número de productos de Asus. Para mostrar todos los fabricantes que ofrecen esa cantidad, la consulta anterior sería la subconsulta.

```
mysql> SELECT f.nombre, COUNT(p.codigo) as cantidad FROM producto as p JOIN fabricante AS f
ON p.codigo_fabricante=f.codigo
GROUP BY f.codigo HAVING cantidad=
(SELECT COUNT(p.codigo) FROM producto AS p JOIN fabricante AS f
ON p.codigo_fabricante=f.codigo WHERE f.nombre="Asus");
+-----+-----+
| nombre | cantidad |
+-----+-----+
| Asus   |         2 |
| Lenovo |         2 |
| Hewlett-Packard |         2 |
| Samsung |         2 |
| Crucial |         2 |
| Dell   |         2 |
+-----+-----+
6 rows in set (0.00 sec)
```

8.2.3. Subconsultas en la cláusula FROM

Detrás de FROM se especifica/n la/s tablas en las que se basará la consulta. Por lo tanto, en este caso el resultado de la subconsulta será una tabla desde la que se realizará la consulta principal.

Ejemplo: Devuelve una listado de todos los productos que tienen un precio mayor o igual al precio medio de cada fabricante. Empezaremos por obtener una consulta que nos devuelva una tabla con: código de fabricante y su precio medio, posteriormente haremos un join con productos y mostraremos los productos cuyo precio sea mayor que el de la media de su fabricante.


```
mysql> SELECT codigo_fabricante AS fab, AVG(precio) AS media FROM producto
GROUP BY codigo_fabricante;
```

fab	media
1	228.995
2	506.5
3	124.995
4	417.995
5	91.99
6	442.5
7	190
8	87
16	112.5

9 rows in set (0.00 sec)

```
mysql> SELECT * FROM producto AS p JOIN
(SELECT codigo_fabricante AS fab, AVG(precio) AS media FROM producto
GROUP BY codigo_fabricante) AS t
ON p.codigo_fabricante=t.fab WHERE p.precio>=t.media;
```

codigo	nombre	precio	codigo_fabricante	fab	media
7	Monitor 27 LED Full HD	250.99	1	1	228.995
8	Portátil Yoga 520	564	2	2	506.5
11	Impresora HP Laserjet Pro M26nw	185	3	3	124.995
12	Portátil i7	680	4	4	417.995
1	Disco duro SATA3 1TB	91.99	5	5	91.99
5	GeForce GTX 1080 Xtreme	760	6	6	442.5
4	GeForce GTX 1050Ti	190	7	7	190
13	Disco SSD 480 GB	87	8	8	87
14	Disco SSD 1TB	175	16	16	112.5

9 rows in set (0.01 sec)

8.2.4. Subconsultas en la cláusula SELECT

```
mysql> SELECT codigo, nombre, precio,
(select avg(t.precio) from producto as t
where p.codigo_fabricante=t.codigo_fabricante) as "Media fabricante",
codigo_fabricante as fab
from producto as p order by p.codigo_fabricante;
```

codigo	nombre	precio	Media fabricante	fab
6	Monitor 24 LED Full HD	207	228.995	1
7	Monitor 27 LED Full HD	250.99	228.995	1
8	Portátil Yoga 520	564	506.5	2
9	Portátil Ideapd 320	449	506.5	2
10	Impresora HP Deskjet 3720	64.99	124.995	3
11	Impresora HP Laserjet Pro M26nw	185	124.995	3
3	Disco SSD 1 TB	155.99	417.995	4
12	Portátil i7	680	417.995	4
1	Disco duro SATA3 1TB	91.99	91.99	5
2	Memoria RAM DDR4 8GB	125	442.5	6
5	GeForce GTX 1080 Xtreme	760	442.5	6
4	GeForce GTX 1050Ti	190	190	7
13	Disco SSD 480 GB	87	87	8
14	Disco SSD 1TB	175	112.5	16
15	Disco SSD 256 GB	50	112.5	16

15 rows in set (0.01 sec)

La subconsulta ha de devolver un escalar que se mostrará o con el que se realizará una expresión en los elementos del SELECT de la consulta principal.

Ejemplo: Realiza un consulta en la que se muestren los datos de todos los productos comparados con la media de cada fabricante. La media de cada fabricante es un valor que se ha de calcular para cada producto a mostrar. Obviamente, todos los productos del mismo fabricante tendrán la misma media.

8.3. Operadores usados en subconsultas

Los operadores que podemos usar en las subconsultas son los siguientes:

- Operadores básicos de comparación (>, >=, <, <=, !=, <>, =).
- Predicados ALL y ANY.
- Predicado IN y NOT IN.
- Predicado EXISTS y NOT EXISTS.

8.3.1. Operadores básicos de comparación

Los operadores básicos de comparación (>, >=, <, <=, !=, <>, =) se pueden usar cuando queremos comparar una expresión con el valor que devuelve una subconsulta.

Los operadores básicos de comparación los vamos a utilizar para realizar comparaciones con **subconsultas que devuelven un único valor, es decir, una columna y una fila.**

Ejemplo: Devuelve todos los productos de la base de datos que tienen un precio mayor o igual al producto más caro del fabricante Asus.

```
mysql> SELECT * FROM producto WHERE precio >=
      (SELECT MAX(precio) FROM producto WHERE codigo_fabricante =
      (SELECT codigo FROM fabricante WHERE nombre = "Asus")
      );
```

codigo	nombre	precio	codigo_fabricante
5	GeForce GTX 1080 Xtreme	760	6
7	Monitor 27 LED Full HD	250.99	1
8	Portátil Yoga 520	564	2
9	Portátil Ideapad 320	449	2
12	Portátil i7	680	4

5 rows in set (0.00 sec)

La consulta anterior también se puede hacer utilizando un JOIN:

```
mysql> SELECT * FROM producto WHERE precio >=
      (SELECT MAX(p.precio) FROM producto AS p JOIN fabricante AS f
      ON p.codigo_fabricante = f.codigo WHERE f.nombre = "Asus"),
```

8.3.2. Subconsultas con ALL y ANY

ALL y ANY se utilizan con los operadores de comparación (>, >=, <, <=, !=, <>, =) y nos permiten comparar una expresión con el conjunto de valores que devuelve una subconsulta.

ALL y ANY los vamos a utilizar para realizar comparaciones con **subconsultas que pueden devolver varios valores, es decir, una columna y varias filas.**

Ejemplo: Podemos escribir la consulta que devuelve todos los productos de la base de datos que tienen un precio mayor o igual al producto más caro del fabricante Asus, haciendo uso de ALL. En primer lugar habría que preparar una consulta que devuelva los precios de los productos de Asus, esta sería la subconsulta. La consulta principal será la de los productos que tengan un precio mayor o igual a **todos** (ALL) los de la lista de la subconsulta.

El resultado sería el mismo si se utiliza la función MAX:

```
mysql> SELECT * FROM producto WHERE precio >= ALL
(SELECT p.precio FROM producto AS p
JOIN fabricante AS f ON p.codigo_fabricante=f.codigo
WHERE f.nombre = "Asus");
```

codigo	nombre	precio	codigo_fabricante
5	GeForce GTX 1080 Xtreme	760	6
7	Monitor 27 LED Full HD	250.99	1
8	Portátil Yoga 520	564	2
9	Portátil Ideapad 320	449	2
12	Portátil i7	680	4

5 rows in set (0.01 sec)

```
mysql> SELECT * FROM producto WHERE precio >=
(SELECT MAX(p.precio) FROM producto AS p
JOIN fabricante AS f ON p.codigo_fabricante=f.codigo
WHERE f.nombre = "Asus");
```

La palabra reservada **SOME** es un alias de **ANY**. Por lo tanto, las siguientes consultas devolverían el mismo resultado:

```
mysql> SELECT s1 FROM t1 WHERE s1 <> ANY (SELECT s1 FROM t2)
mysql> SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2)
```

8.3.3. Subconsultas con IN y NOT IN

IN y **NOT IN** nos permiten comprobar si un valor está o no incluido en un conjunto de valores, que puede ser el conjunto de valores que devuelve una subconsulta.

IN y **NOT IN** los vamos a utilizar para realizar comparaciones con **subconsultas que pueden devolver varios valores, es decir, una columna y varias filas**.

Ejemplo: Devuelve un listado de los clientes que no han realizado ningún pedido. La subconsulta ofrece una lista de clientes que sí han realizado pedidos.

```
mysql> SELECT * FROM cliente WHERE id NOT IN
(SELECT id_cliente FROM pedido);
```

id	nombre	apellido1	apellido2	ciudad	categoria
9	Guillermo	López	Gómez	Granada	225
10	Daniel	Santana	Loyola	Sevilla	125

2 rows in set (0.00 sec)

Cuando estamos trabajando con subconsultas, **IN** y **= ANY** realizan la misma función. Por lo tanto, las siguientes consultas devolverían el mismo resultado:

```
mysql> SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
mysql> SELECT s1 FROM t1 WHERE s1 IN (SELECT s1 FROM t2);
```

Ocurre lo mismo con **NOT IN** y **<> ALL**. Por lo tanto, las siguientes consultas devolverían el mismo resultado:

```
mysql> SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);
mysql> SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);
```

Importante:

Tenga en cuenta que cuando hay **algún valor NULL** en el resultado de la consulta interna, la consulta externa no devuelve ningún resultado.

Ejemplo: Devuelve un listado con el nombre de los departamentos que no tienen empleados asociados.

```
mysql> SELECT codigo_departamento FROM empleado;
+-----+
| codigo_departamento |
+-----+
| NULL                |
| NULL                |
| 1                    |
| 1                    |
| 1                    |
| 1                    |
| 2                    |
| 2                    |
| 2                    |
| 3                    |
| 3                    |
| 3                    |
| 6                    |
| 6                    |
+-----+
15 rows in set (0.00 sec)

mysql> SELECT nombre FROM departamento WHERE codigo NOT IN
      (SELECT codigo_departamento FROM empleado);
Empty set (0.00 sec)
```

Al tener la subconsulta dos resultados NULL cuando tiene que compara (NOT IN) el código de los departamentos con la lista de la subconsulta no sabe como valorar la comparación con el NULL por eso devuelve 0 resultados. Para ello habría que proteger la consulta interna de forma que nunca pueda valer NULL.

```
mysql> SELECT nombre FROM departamento WHERE codigo NOT IN
      (SELECT codigo_departamento FROM empleado
      WHERE codigo_departamento is NOT NULL);
+-----+
| nombre |
+-----+
| Ventas |
| Contabilidad |
+-----+
2 rows in set (0.00 sec)
```

8.3.4. Subconsultas con EXISTS y NOT EXISTS

Con esta cláusula EXISTS, basta con que lo que venga detrás tenga algún resultado para darlo por válido o no.

Ejemplo: Devuelve un listado de los clientes que no han realizado ningún pedido.

```
mysql> SELECT CONCAT_WS(" ", CONCAT_WS(" ", c.apellido1, c.apellido2), c.nombre)
      AS cliente FROM cliente AS c
      WHERE NOT EXISTS (SELECT * FROM pedido AS p WHERE p.id_cliente=c.id);
+-----+
| cliente |
+-----+
| López Gómez, Guillermo |
| Santana Loyola, Daniel |
+-----+
2 rows in set (0.00 sec)
```

Debemos tener en cuenta que dentro del paréntesis que va detrás del SELECT se debe hacer referencia a algún contenido de fuera de este de otra forma no tendría efecto, pues si ofrece alguna fila siempre la ofrecerá al no intervenir ningún elemento de fuera; y si no la ofrece pues nunca la ofrecerá.

8.4. Errores comunes

3.1 Número incorrecto de columnas en la subconsulta

Cuando una subconsulta ofrece más de una columna y donde se espera el resultado solo se espera una columna nos daría este tipo de error.

Ejemplo:

```
mysql>SELECT (SELECT column1, column2 FROM t2) FROM t1;
```

La subconsulta solo debería mostrar una única columna.

El error que aparecería:

```
ERROR 1241 (ER_OPERAND_COL) SQLSTATE = 21000 Message = "Operand should contain 1 column(s)"
```

3.2 Número incorrecto de filas en la subconsulta

Igualmente si el resultado de una subconsulta se espera que sea una única fila y nos ofrece más de una también daría error, pues no sabría cual de ellas utilizar. Supongamos que estamos haciendo una comparación con el valor que ofrezca una subconsulta, si esta ofrece más de un valor, ¿Con cual de ellos se compara?

Ejemplo:

```
mysql>SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

La consulta anterior sólo se podrá ejecutar cuando la consulta interna `SELECT column1 FROM t2` devuelva una única fila.

El error que aparecería:

```
ERROR 1242 (ER_SUBSELECT_NO_1_ROW) SQLSTATE = 21000 Message = "Subquery returns more than 1 row"
```