

## Creación de usuarios

Para crear un usuario en MySQL, se utiliza el comando **CREATE USER**. A continuación, se pueden establecer diferentes restricciones y privilegios para ese usuario utilizando el comando **GRANT**.

1. Crear un usuario con acceso completo a todas las bases de datos:

1. 

```
CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'contraseña';
```
2. 

```
GRANT ALL PRIVILEGES ON *.* TO 'usuario'@'localhost' WITH GRANT OPTION;
```

2. Crear un usuario con acceso a una base de datos específica:

1. 

```
CREATE USER 'usuario2'@'localhost' IDENTIFIED BY 'contraseña';
```
2. 

```
GRANT ALL PRIVILEGES ON municipios.* TO 'usuario2'@'localhost' WITH GRANT OPTION;
```

3. Crear un usuario con acceso limitado a una tabla específica:

1. 

```
CREATE USER 'usuario3'@'localhost' IDENTIFIED BY 'contraseña';
```
2. 

```
GRANT SELECT, INSERT, UPDATE ON municipios.pueblos TO 'usuario3'@'localhost';
```

4. Modificar los permisos de un usuario en una base de datos existente:

1. 

```
GRANT SELECT, INSERT ON database_name.* TO 'user_name'@'localhost';
```

En este ejemplo, la sentencia "GRANT" otorga los permisos SELECT e INSERT al usuario "user\_name" en la base de datos "database\_name", reemplazando cualquier permiso anterior que el usuario pueda haber tenido en la misma base de datos.

5. Quitar los privilegios de un usuario:

1. 

```
REVOKE ALL PRIVILEGES ON *.* FROM 'usuario'@'localhost';
```

6. Revocar un permiso específico de un usuario en una base de datos:

```
REVOKE SELECT ON database_name.* FROM 'user_name'@'localhost';
```

En este ejemplo, la sentencia "REVOKE" revoca el permiso SELECT del usuario "user\_name" en la base de datos "database\_name".

7. Según la BBDD vas a necesitar el comando **FLUSH PRIVILEGES** para asegurar que los cambios se apliquen

## Constraint

El comando "constraint" se utiliza en MySQL para agregar restricciones o reglas adicionales a una tabla o columna.

Agregar o modificar una restricción de clave primaria:

```
ALTER TABLE malaga
ADD CONSTRAINT pk_nombre PRIMARY KEY (nombre_muni);
```

Agregar o modificar una restricción de clave foránea:

```
ALTER TABLE tabla1
ADD CONSTRAINT fk_nombre
FOREIGN KEY (columna) REFERENCES tabla2(columna);
```

Agregar o modificar una restricción de unicidad:

```
ALTER TABLE tabla
ADD CONSTRAINT uq_nombre UNIQUE (columna);
```

Agregar o modificar una restricción de verificación:

```
ALTER TABLE tabla
ADD CONSTRAINT chk_nombre CHECK (columna > 0);
```

## Modificar varios campos

Podemos modificar varios campos de una tabla en MySQL en una sola sentencia, se puede utilizar la sentencia "ALTER TABLE" seguida de la cláusula "MODIFY" para especificar cada campo que se desea modificar y su nuevo tipo de datos.

Por ejemplo, si queremos modificar los campos "nombre\_muni" y "altitud" de la tabla "malaga" y cambiar su tipo de datos a VARCHAR(50) y INT(11), respectivamente, podemos utilizar la siguiente sentencia:

```
ALTER TABLE malaga
MODIFY nombre_muni VARCHAR(50),
MODIFY altitud INT(11);
```

En este ejemplo, la sentencia "ALTER TABLE" modifica los campos "nombre\_muni" y "altitud" de la tabla "malaga" y utiliza la cláusula "MODIFY" para especificar su nuevo tipo de datos.

Es importante tener en cuenta que, al modificar varios campos en una tabla en MySQL en una sola sentencia, es necesario **separar cada modificación con una coma (,)**. Además, se debe

tener cuidado al modificar el tipo de datos de un campo, ya que **puede provocar la pérdida de datos** si no se realiza correctamente.

## SUBCONSULTAS

Podemos usarlas dentro del

**SELECT**

**FROM**

**WHERE**

Aquí hay algunos ejemplos de subconsultas en SQL:

**Obtener el nombre y la cantidad de pedidos de los clientes que hicieron más de 5 pedidos:**

```
SELECT customer_name, COUNT(order_id) as total_orders
FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING
COUNT(*) > 5)
GROUP BY customer_id;
```

En este ejemplo, la subconsulta dentro de la cláusula WHERE cuenta la cantidad de pedidos de cada cliente y selecciona solo aquellos que tienen más de 5 pedidos. La consulta principal luego muestra el nombre del cliente y el total de pedidos realizados.

**Obtener el nombre de los productos y su precio promedio, donde el precio es mayor que el promedio de todos los productos:**

```
SELECT product_name, price
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

En este ejemplo, la subconsulta dentro de la cláusula WHERE calcula el precio promedio de todos los productos y luego selecciona solo aquellos productos cuyo precio es mayor que el promedio. La consulta principal muestra el nombre del producto y su precio.

**Obtener el nombre de los empleados que trabajan en el mismo departamento que el empleado con el número de empleado 100:**

```
SELECT employee_name
```

```
FROM employees
```

```
WHERE department_id = (SELECT department_id FROM employees WHERE employee_id = 100);
```

En este ejemplo, la subconsulta dentro de la cláusula WHERE selecciona el departamento del empleado con el número de empleado 100 y la consulta principal muestra el nombre de todos los empleados que trabajan en ese departamento.

#### **Obtener el número total de órdenes y el número total de órdenes en los últimos 30 días:**

```
SELECT COUNT(order_id) AS total_orders,
```

```
    (SELECT COUNT(order_id) FROM orders WHERE order_date >= DATE_SUB(NOW(),  
INTERVAL 30 DAY)) AS orders_last_30_days
```

```
FROM orders;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el número total de órdenes en los últimos 30 días. La consulta principal devuelve el número total de órdenes junto con el número total de órdenes en los últimos 30 días.

#### **Obtener el precio promedio de los productos de cada categoría:**

```
SELECT category_name,
```

```
    (SELECT AVG(price) FROM products WHERE products.category_id = categories.category_id)  
AS avg_price
```

```
FROM categories;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el precio promedio de los productos de cada categoría. La consulta principal devuelve el nombre de la categoría junto con el precio promedio de los productos en esa categoría.

#### **Obtener el número total de órdenes y el número total de órdenes realizadas por clientes con más de 10 órdenes:**

```
SELECT COUNT(order_id) AS total_orders,
```

```
    (SELECT COUNT(order_id) FROM orders WHERE customer_id IN (SELECT customer_id  
FROM orders GROUP BY customer_id HAVING COUNT(*) > 10)) AS orders_by_top_customers
```

```
FROM orders;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el número total de órdenes realizadas por clientes con más de 10 órdenes. La consulta principal devuelve el número total de órdenes junto con el número total de órdenes realizadas por los clientes más importantes.

Para crear un usuario en MySQL, se utiliza el comando CREATE USER. A continuación, se pueden establecer diferentes restricciones y privilegios para ese usuario utilizando el comando GRANT.

Aquí hay algunos ejemplos de cómo crear un usuario y establecer restricciones en MySQL:

Crear un usuario con acceso completo a todas las bases de datos:

sql

Copy code

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';  
GRANT ALL PRIVILEGES ON *.* TO 'usuario'@'localhost' WITH GRANT OPTION;
```

Crear un usuario con acceso a una base de datos específica:

sql

Copy code

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';  
GRANT ALL PRIVILEGES ON basedatos.* TO 'usuario'@'localhost' WITH GRANT OPTION;
```

Crear un usuario con acceso limitado a una tabla específica:

sql

Copy code

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

```
GRANT SELECT, INSERT, UPDATE ON basedatos.tabla TO 'usuario'@'localhost';
```

Crear un usuario sin acceso a la base de datos:

sql

Copy code

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

```
REVOKE ALL PRIVILEGES ON *.* FROM 'usuario'@'localhost';
```

Es importante recordar que, una vez que se hayan establecido las restricciones, debe ejecutarse el comando FLUSH PRIVILEGES para asegurarse de que los cambios se actualicen y se apliquen.

José Antonio Pérez Alias

explicame las restricciones en mysql y pon varios ejemplos

Las restricciones en MySQL son utilizadas para limitar el acceso y los permisos de los usuarios en una base de datos. Las restricciones pueden ser aplicadas a diferentes niveles, incluyendo a nivel de usuario, a nivel de base de datos y a nivel de tabla.

Aquí hay algunos ejemplos de restricciones que se pueden establecer en MySQL:

Restricciones de privilegios: los privilegios controlan los tipos de operaciones que los usuarios pueden realizar en la base de datos. Por ejemplo, se puede restringir a un usuario para que sólo pueda realizar operaciones de lectura (SELECT) en una tabla específica, mientras que otro usuario puede tener permisos para realizar operaciones de escritura (INSERT, UPDATE, DELETE) en la misma tabla.

sql

Copy code

```
GRANT SELECT ON tabla1 TO 'usuario1'@'localhost';
```

```
GRANT INSERT, UPDATE, DELETE ON tabla1 TO 'usuario2'@'localhost';
```

Restricciones de acceso: el acceso a la base de datos se puede restringir por dirección IP o por nombre de host. Por ejemplo, se puede permitir el acceso a la base de datos sólo desde una dirección IP específica.

sql

Copy code

```
CREATE USER 'usuario'@'192.168.0.100' IDENTIFIED BY 'contraseña';  
GRANT ALL PRIVILEGES ON basedatos.* TO 'usuario'@'192.168.0.100';
```

Restricciones de tamaño y tipo de datos: se pueden establecer restricciones para asegurar que los datos ingresados en una tabla cumplan con ciertos requisitos. Por ejemplo, se puede establecer una restricción en la longitud máxima de un campo de texto.

sql

Copy code

```
CREATE TABLE tabla1 (  
    id INT NOT NULL,  
    nombre VARCHAR(50) NOT NULL,  
    edad INT CHECK (edad > 0 AND edad < 120),  
    PRIMARY KEY (id)  
);
```

Restricciones de integridad referencial: las restricciones de integridad referencial aseguran que las relaciones entre tablas se mantengan correctamente. Por ejemplo, se puede establecer una restricción para que los registros en una tabla de pedidos sólo puedan hacer referencia a registros válidos en una tabla de clientes.

sql

Copy code

```
CREATE TABLE clientes (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE pedidos (  
    id INT PRIMARY KEY,  
    cliente_id INT NOT NULL,  
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)  
);
```

Estas son sólo algunas de las restricciones que se pueden establecer en MySQL. Es importante conocer las restricciones adecuadas para cada situación para asegurarse de que los datos se mantengan seguros y se cumplan las regulaciones y normativas necesarias.

José Antonio Pérez Alias

¿puedes explicar el uso del contrains en mysql y poner varios ejemplos?

Las constraints (restricciones) son utilizadas en MySQL para establecer reglas que se deben cumplir para los datos almacenados en una tabla. Estas restricciones se aplican a nivel de columna o de tabla completa, y ayudan a garantizar la integridad de los datos y la coherencia de la base de datos.

A continuación se presentan algunos ejemplos de cómo se pueden utilizar las restricciones en MySQL:

**PRIMARY KEY:** se utiliza para identificar de forma única cada fila de una tabla. Una tabla puede tener solamente una primary key y no puede tener valores nulos.

sql

Copy code

```
CREATE TABLE clientes (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(50),  
    email VARCHAR(100) UNIQUE  
);
```

**FOREIGN KEY:** se utiliza para establecer relaciones entre tablas. La foreign key apunta a la primary key de otra tabla. Esta restricción asegura que los valores en una columna existen en la primary key de otra tabla.

sql

Copy code

```
CREATE TABLE pedidos (  
    id INT PRIMARY KEY,  
    cliente_id INT,  
    fecha_pedido DATE,  
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)  
);
```



UNIQUE: se utiliza para asegurar que los valores en una columna sean únicos. Una tabla puede tener múltiples columnas con valores únicos.

sql

Copy code

```
CREATE TABLE productos (  
  id INT PRIMARY KEY,  
  nombre VARCHAR(50),  
  sku VARCHAR(20) UNIQUE,  
  precio DECIMAL(10,2)  
);
```

CHECK: se utiliza para establecer una condición que debe cumplirse antes de que se puedan agregar valores a una columna.

sql

Copy code

```
CREATE TABLE empleados (  
  id INT PRIMARY KEY,  
  nombre VARCHAR(50),  
  edad INT CHECK (edad > 18),  
  puesto VARCHAR(50)  
);
```

En general, el uso de restricciones es muy importante para mantener la integridad de los datos en una base de datos y asegurarse de que se cumplan las normas y reglamentaciones necesarias.

José Antonio Pérez Alias

¿cuando usar el comando constraint en mysql?

El comando "constraint" se utiliza en MySQL para agregar restricciones o reglas adicionales a una tabla o columna.

Puede ser utilizado para agregar o modificar restricciones de clave primaria, clave foránea, restricciones de unicidad, restricciones de verificación y otros tipos de restricciones.

Algunos casos en los que se puede utilizar el comando "constraint" en MySQL son:

Agregar o modificar una restricción de clave primaria:

sql

Copy code

```
ALTER TABLE tabla
```

```
ADD CONSTRAINT pk_nombre PRIMARY KEY (columna);
```

Agregar o modificar una restricción de clave foránea:

sql

Copy code

```
ALTER TABLE tabla1
```

```
ADD CONSTRAINT fk_nombre
```

```
FOREIGN KEY (columna) REFERENCES tabla2(columna);
```

Agregar o modificar una restricción de unicidad:

sql

Copy code

```
ALTER TABLE tabla
```

```
ADD CONSTRAINT uq_nombre UNIQUE (columna);
```

Agregar o modificar una restricción de verificación:

sql

Copy code

```
ALTER TABLE tabla
```

```
ADD CONSTRAINT chk_nombre CHECK (columna > 0);
```

En general, se utiliza el comando "constraint" en MySQL para agregar o modificar restricciones que aseguren la integridad de los datos en una tabla y garanticen que se cumplan las normas y reglamentaciones necesarias.

José Antonio Pérez Alias

¿cuando usar el comando cascade on en mysql?

El comando "ON CASCADE" en MySQL se utiliza en conjunto con las restricciones de clave foránea (FOREIGN KEY) para especificar qué acción se debe tomar en una tabla secundaria cuando se actualiza o elimina una fila en la tabla primaria.

La sintaxis básica del comando "ON CASCADE" es la siguiente:

graphql

Copy code

ON DELETE | ON UPDATE CASCADE

Donde "ON DELETE" especifica la acción a tomar en la tabla secundaria cuando se elimina una fila de la tabla primaria, y "ON UPDATE" especifica la acción a tomar cuando se actualiza una fila de la tabla primaria.

En general, se utiliza el comando "ON CASCADE" en MySQL cuando se desea actualizar o eliminar una fila en la tabla primaria y todas las filas relacionadas en la tabla secundaria también deben actualizarse o eliminarse.

Por ejemplo, si tenemos una tabla "orders" y otra tabla "order\_details" con una restricción de clave foránea de "orders" a "order\_details", y queremos eliminar una fila de "orders" y también todas las filas relacionadas de "order\_details", podemos utilizar el comando "ON DELETE CASCADE" de la siguiente manera:

sql

Copy code

ALTER TABLE order\_details

ADD CONSTRAINT fk\_order\_id

FOREIGN KEY (order\_id) REFERENCES orders(id)

ON DELETE CASCADE;

En este ejemplo, cuando se elimina una fila de la tabla "orders", todas las filas relacionadas en la tabla "order\_details" también se eliminarán automáticamente debido a la cláusula "ON DELETE CASCADE".

En general, se debe tener cuidado al utilizar el comando "ON CASCADE" en MySQL ya que puede provocar la eliminación involuntaria de datos si no se realiza correctamente.

José Antonio Pérez Alias

Pon un ejemplo de como modificar varios campos de una tabla en mysql en una sola sentencia

Para modificar varios campos de una tabla en MySQL en una sola sentencia, se puede utilizar la sentencia "ALTER TABLE" seguida de la cláusula "MODIFY" para especificar cada campo que se desea modificar y su nuevo tipo de datos.

Por ejemplo, si queremos modificar los campos "name" y "age" de la tabla "users" y cambiar su tipo de datos a VARCHAR(50) y INT(11), respectivamente, podemos utilizar la siguiente sentencia:

scss

Copy code

```
ALTER TABLE users  
MODIFY name VARCHAR(50),  
MODIFY age INT(11);
```

En este ejemplo, la sentencia "ALTER TABLE" modifica los campos "name" y "age" de la tabla "users" y utiliza la cláusula "MODIFY" para especificar su nuevo tipo de datos.

Es importante tener en cuenta que al modificar varios campos en una tabla en MySQL en una sola sentencia, es necesario separar cada modificación con una coma (,). Además, se debe tener cuidado al modificar el tipo de datos de un campo, ya que puede provocar la pérdida de datos si no se realiza correctamente.

José Antonio Pérez Alias

ejemplos de como eliminar o modificar los permisos de usuarios en una base de datos con mysql

Para eliminar o modificar los permisos de un usuario en una base de datos MySQL, se puede utilizar la sentencia "REVOKE" para revocar los permisos existentes o la sentencia "GRANT" para otorgar nuevos permisos.

Aquí hay algunos ejemplos:

Revocar todos los permisos de un usuario en una base de datos:

sql

Copy code

```
REVOKE ALL PRIVILEGES ON database_name.* FROM 'user_name'@'localhost';
```

En este ejemplo, la sentencia "REVOKE" revoca todos los permisos del usuario "user\_name" en la base de datos "database\_name".

Revocar un permiso específico de un usuario en una base de datos:

sql

Copy code

```
REVOKE SELECT ON database_name.* FROM 'user_name'@'localhost';
```

En este ejemplo, la sentencia "REVOKE" revoca el permiso SELECT del usuario "user\_name" en la base de datos "database\_name".

Otorgar un nuevo permiso a un usuario en una base de datos:

sql

Copy code

```
GRANT INSERT ON database_name.* TO 'user_name'@'localhost';
```

En este ejemplo, la sentencia "GRANT" otorga el permiso INSERT al usuario "user\_name" en la base de datos "database\_name".

Modificar los permisos de un usuario en una base de datos existente:

sql

Copy code

```
GRANT SELECT, INSERT ON database_name.* TO 'user_name'@'localhost';
```

En este ejemplo, la sentencia "GRANT" otorga los permisos SELECT e INSERT al usuario "user\_name" en la base de datos "database\_name", reemplazando cualquier permiso anterior que el usuario pueda haber tenido en la misma base de datos.

Es importante tener en cuenta que al eliminar o modificar los permisos de un usuario en una base de datos MySQL, se debe tener cuidado para asegurarse de que se estén revocando o otorgando los permisos correctos y que no se estén dando permisos innecesarios o peligrosos.

José Antonio Pérez Alias

como hacer subconsultas en sql, teoria y varios ejemplos ilustrativos

Las subconsultas en SQL son consultas anidadas dentro de otras consultas. Se utilizan para obtener información de una tabla basándose en los resultados de otra consulta.

La sintaxis básica de una subconsulta en SQL es:

sql

Copy code

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
WHERE column_name operator (SELECT column_name FROM table_name WHERE condition);
```

Donde "table\_name" es el nombre de la tabla, "column\_name" es el nombre de la columna, "operator" es el operador de comparación (como "=" o ">"), "condition" es la condición que debe cumplir la subconsulta y "column1", "column2" son las columnas que se desean mostrar en la consulta principal.

Aquí hay algunos ejemplos de subconsultas en SQL:

Obtener el nombre y la cantidad de pedidos de los clientes que hicieron más de 5 pedidos:

sql

Copy code

```
SELECT customer_name, COUNT(order_id) as total_orders
```

```
FROM customers
```

```
WHERE customer_id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING  
COUNT(*) > 5)
```

GROUP BY customer\_id;

En este ejemplo, la subconsulta dentro de la cláusula WHERE cuenta la cantidad de pedidos de cada cliente y selecciona solo aquellos que tienen más de 5 pedidos. La consulta principal luego muestra el nombre del cliente y el total de pedidos realizados.

Obtener el nombre de los productos y su precio promedio, donde el precio es mayor que el promedio de todos los productos:

sql

Copy code

```
SELECT product_name, price  
FROM products  
WHERE price > (SELECT AVG(price) FROM products);
```

En este ejemplo, la subconsulta dentro de la cláusula WHERE calcula el precio promedio de todos los productos y luego selecciona solo aquellos productos cuyo precio es mayor que el promedio. La consulta principal muestra el nombre del producto y su precio.

Obtener el nombre de los empleados que trabajan en el mismo departamento que el empleado con el número de empleado 100:

sql

Copy code

```
SELECT employee_name  
FROM employees  
WHERE department_id = (SELECT department_id FROM employees WHERE employee_id =  
100);
```

En este ejemplo, la subconsulta dentro de la cláusula WHERE selecciona el departamento del empleado con el número de empleado 100 y la consulta principal muestra el nombre de todos los empleados que trabajan en ese departamento.

Es importante tener en cuenta que las subconsultas en SQL pueden tener un impacto en el rendimiento de la consulta, por lo que es importante asegurarse de que se estén utilizando de manera eficiente y que se estén limitando a las consultas necesarias.

José Antonio Pérez Alias

ejemplos se usar consultas dentro de los campos del propio select en el lenguaje sql

Sí, en SQL es posible utilizar subconsultas dentro de los campos del SELECT para realizar operaciones más complejas en los datos.

Aquí hay algunos ejemplos de cómo utilizar subconsultas en los campos del SELECT en SQL:

Obtener el número total de órdenes y el número total de órdenes en los últimos 30 días:

sql

Copy code

```
SELECT COUNT(order_id) AS total_orders,  
       (SELECT COUNT(order_id) FROM orders WHERE order_date >= DATE_SUB(NOW(),  
INTERVAL 30 DAY)) AS orders_last_30_days  
FROM orders;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el número total de órdenes en los últimos 30 días. La consulta principal devuelve el número total de órdenes junto con el número total de órdenes en los últimos 30 días.

Obtener el precio promedio de los productos de cada categoría:

sql

Copy code

```
SELECT category_name,  
       (SELECT AVG(price) FROM products WHERE products.category_id = categories.category_id)  
AS avg_price  
FROM categories;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el precio promedio de los productos de cada categoría. La consulta principal devuelve el nombre de la categoría junto con el precio promedio de los productos en esa categoría.

Obtener el número total de órdenes y el número total de órdenes realizadas por clientes con más de 10 órdenes:

sql

Copy code

```
SELECT COUNT(order_id) AS total_orders,
```



```
(SELECT COUNT(order_id) FROM orders WHERE customer_id IN (SELECT customer_id
FROM orders GROUP BY customer_id HAVING COUNT(*) > 10)) AS orders_by_top_customers
FROM orders;
```

En este ejemplo, se utiliza una subconsulta dentro de los campos del SELECT para obtener el número total de órdenes realizadas por clientes con más de 10 órdenes. La consulta principal devuelve el número total de órdenes junto con el número total de órdenes realizadas por los clientes más importantes.

José Antonio Pérez Alias

utilizar consultas dentro del from, en el lenguaje sql, teoria y varios ejemplos

En SQL, es posible utilizar consultas dentro de la cláusula FROM para crear tablas virtuales y utilizarlas en una consulta principal. Estas consultas, también conocidas como subconsultas de tabla, pueden ser muy útiles para realizar operaciones más complejas en los datos.

Aquí hay algunos ejemplos de cómo utilizar consultas dentro de la cláusula FROM en SQL:

Obtener el número total de órdenes y el número total de órdenes realizadas por clientes con más de 10 órdenes utilizando una consulta de tabla:

sql

Copy code

```
SELECT COUNT(o.order_id) AS total_orders, COUNT(o2.order_id) AS orders_by_top_customers
FROM orders o
LEFT JOIN (SELECT customer_id, COUNT(*) as order_count FROM orders GROUP BY
customer_id HAVING COUNT(*) > 10) c
ON o.customer_id = c.customer_id
LEFT JOIN orders o2
ON o.customer_id = o2.customer_id
WHERE o.order_date >= DATE_SUB(NOW(), INTERVAL 30 DAY);
```

En este ejemplo, se utiliza una consulta de tabla para obtener el número total de órdenes realizadas por clientes con más de 10 órdenes. La consulta principal utiliza LEFT JOIN para unirse a la tabla virtual con la tabla de órdenes, y devuelve el número total de órdenes junto

con el número total de órdenes realizadas por los clientes más importantes en los últimos 30 días.

Obtener la cantidad de órdenes y el total de ventas para cada cliente utilizando una consulta de tabla:

vbnet

Copy code

```
SELECT c.customer_id, c.customer_name, COUNT(o.order_id) AS order_count,  
SUM(o.total_price) AS total_sales  
  
FROM customers c  
  
LEFT JOIN (SELECT order_id, customer_id, total_price FROM orders) o  
ON c.customer_id = o.customer_id  
  
GROUP BY c.customer_id, c.customer_name;
```

En este ejemplo, se utiliza una consulta de tabla para unir la tabla de clientes con la tabla de órdenes y obtener la cantidad de órdenes y el total de ventas para cada cliente. La consulta principal utiliza LEFT JOIN para unirse a la tabla virtual con la tabla de clientes y devuelve el número total de órdenes y el total de ventas por cada cliente.

Obtener la cantidad de productos vendidos y el total de ventas por categoría utilizando una consulta de tabla:

vbnet

Copy code

```
SELECT c.category_name, COUNT(p.product_id) AS products_sold, SUM(o.total_price) AS  
total_sales  
  
FROM categories c  
  
LEFT JOIN (SELECT product_id, category_id FROM products) p  
ON c.category_id = p.category_id  
  
LEFT JOIN (SELECT order_id, product_id, total_price FROM order_items) oi  
ON p.product_id = oi.product_id  
  
LEFT JOIN orders o  
ON oi.order_id = o.order_id  
  
GROUP BY c.category_name;
```

En este ejemplo, se utiliza una consulta de tabla para unir la tabla de categorías con la tabla de productos y la tabla de elementos de órdenes y la tabla de órdenes, y obtener la cantidad de productos vendidos y el total de ventas por categoría. La consulta principal utiliza LEFT JOIN

para unirse a la tabla virtual con la tabla de categorías y devuelve la cantidad de productos vendidos y el total de ventas por cada categoría.

**Obtener el número total de órdenes y el número total de órdenes realizadas por clientes con más de 10 órdenes utilizando una consulta de tabla:**

```
SELECT COUNT(o.order_id) AS total_orders, COUNT(o2.order_id) AS orders_by_top_customers
FROM orders o
LEFT JOIN (SELECT customer_id, COUNT(*) as order_count FROM orders GROUP BY
customer_id HAVING COUNT(*) > 10) c
ON o.customer_id = c.customer_id
LEFT JOIN orders o2
ON o.customer_id = o2.customer_id
WHERE o.order_date >= DATE_SUB(NOW(), INTERVAL 30 DAY);
```