

Sub-query calculated fields

A sub-query can be used to generate a calculated field that returns values from a table to an outer **SELECT** statement.

Given the “customers” and “orders” tables from the previous example, a sub-query could use the **COUNT()** aggregate function to calculate how many orders have been placed by each account number in the “orders” table. The outer **SELECT** statement can then retrieve each customer name to display with this calculated field. The **SELECT** query and sub-query could look like this:

```
SELECT name , customers.acc_num ,
       ( SELECT COUNT( * ) FROM orders
         WHERE orders.acc_num = customers.acc_num )
AS number_of_orders FROM customers
ORDER BY customers.acc_num ;
```

The script below produces the same result without a sub-query. Both methods generate a “number_of_orders” calculated field alongside customer names, sorted by their account number:



subquery-calc.sql

```
# Use the "my_database" database.
USE my_database ;

# Create a table called "customers".
CREATE TABLE IF NOT EXISTS customers
( acc_num INT PRIMARY KEY , name CHAR( 20 ) NOT NULL ) ;

# Insert 3 records into the "customers" table.
INSERT INTO customers ( acc_num , name )
VALUES ( 123 , "T.Smith" ) ;
INSERT INTO customers ( acc_num , name )
VALUES ( 124 , "P.Jones" ) ;
INSERT INTO customers ( acc_num , name )
VALUES ( 125 , "H.Nicks" ) ;

# Create a table called "orders".
CREATE TABLE IF NOT EXISTS orders
( ord_num INT PRIMARY KEY , acc_num INT NOT NULL ) ;

# Insert 5 records into the "orders" table.
INSERT INTO orders ( ord_num , acc_num ) VALUES ( 1 , 123 ) ;
INSERT INTO orders ( ord_num , acc_num ) VALUES ( 2 , 124 ) ;
```

...cont'd

```
INSERT INTO orders ( ord_num , acc_num ) VALUES ( 3 , 125 );  
INSERT INTO orders ( ord_num , acc_num ) VALUES ( 4 , 125 );  
INSERT INTO orders ( ord_num , acc_num ) VALUES ( 5 , 123 );
```

Display all data in "customers" and "orders" tables.

```
SELECT * FROM customers ; SELECT * FROM orders ;
```

Get the number of orders per customer.

```
SELECT name , customers.acc_num , COUNT( * ) AS number_of_  
orders
```

```
FROM customers , orders
```

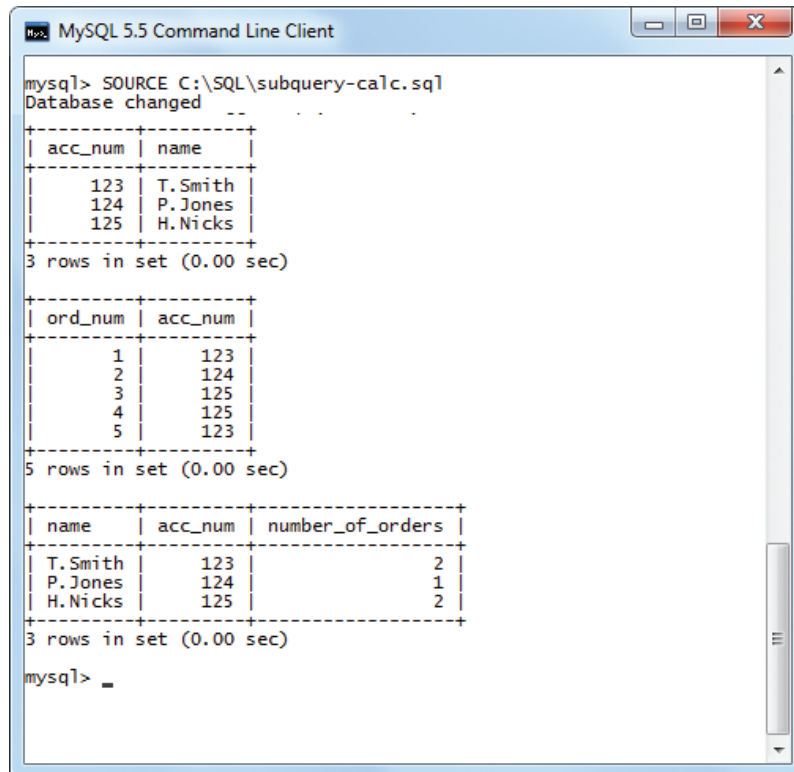
```
WHERE customers.acc_num = orders.acc_num
```

```
GROUP BY name ORDER BY customers.acc_num ;
```

Delete these sample tables.

```
DROP TABLE IF EXISTS customers ;
```

```
DROP TABLE IF EXISTS orders ;
```



The screenshot shows a MySQL 5.5 Command Line Client window. The user has executed the command `SOURCE C:\SQL\subquery-calc.sql`, which has changed the database. The output consists of three tables:

```
mysql> SOURCE C:\SQL\subquery-calc.sql  
Database changed
```

acc_num	name
123	T.Smith
124	P.Jones
125	H.Nicks

3 rows in set (0.00 sec)

ord_num	acc_num
1	123
2	124
3	125
4	125
5	123

5 rows in set (0.00 sec)

name	acc_num	number_of_orders
T.Smith	123	2
P.Jones	124	1
H.Nicks	125	2

3 rows in set (0.00 sec)

```
mysql> _  
mysql>
```