MySQL correlated subquery

In the following query, we select products whose buy prices are greater than the average buy price of all products in each product line.

SELECT

    productname,

    buyprice

FROM

    products p1

WHERE

    buyprice > (SELECT

            AVG(buyprice)

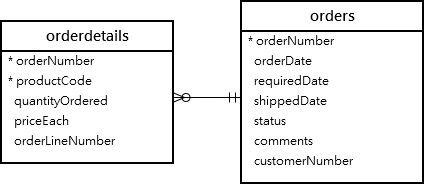
        FROM

            products

        WHERE

            productline = p1.productline)

2/



The following query finds sales orders whose total values are greater than 60K.

SELECT

    orderNumber,

    SUM(priceEach \* quantityOrdered) total

FROM

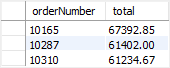
    orderdetails

        INNER JOIN

    orders USING (orderNumber)

GROUP BY orderNumber

HAVING SUM(priceEach \* quantityOrdered) > 60000;



It returns 3 rows, meaning that there are 3 sales orders whose total values are greater than 60K.

You can use the query above as a correlated subquery to find customers who placed at least one sales order with the total value greater than 60K by using the EXISTS operator:

SELECT

    customerNumber,

    customerName

FROM

    customers

WHERE

    EXISTS( SELECT

            orderNumber, SUM(priceEach \* quantityOrdered)

        FROM

            orderdetails

                INNER JOIN

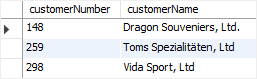
            orders USING (orderNumber)

        WHERE

            customerNumber = customers.customerNumber

        GROUP BY orderNumber

        HAVING SUM(priceEach \* quantityOrdered) > 60000);



B/

#### 13.2.11.6 Subqueries with EXISTS or NOT EXISTS

If a subquery returns any rows at all, EXISTS ***subquery*** is TRUE, and NOT EXISTS ***subquery*** is FALSE. For example:

SELECT column1 FROM t1 WHERE EXISTS (SELECT \* FROM t2);

Traditionally, an EXISTS subquery starts with SELECT \*, but it could begin with SELECT 5 or SELECT column1 or anything at all. MySQL ignores the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) list in such a subquery, so it makes no difference.

For the preceding example, if t2 contains any rows, even rows with nothing but NULL values, the EXISTS condition is TRUE. This is actually an unlikely example because a [NOT] EXISTS subquery almost always contains correlations. Here are some more realistic examples:

* What kind of store is present in one or more cities?
* SELECT DISTINCT store\_type FROM stores
* WHERE EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

* What kind of store is present in no cities?
* SELECT DISTINCT store\_type FROM stores
* WHERE NOT EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

* What kind of store is present in all cities?
* SELECT DISTINCT store\_type FROM stores s1
* WHERE NOT EXISTS (
* SELECT \* FROM cities WHERE NOT EXISTS (
* SELECT \* FROM cities\_stores
* WHERE cities\_stores.city = cities.city

AND cities\_stores.store\_type = stores.store\_type));

The last example is a double-nested NOT EXISTS query. That is, it has a NOT EXISTS clause within a NOT EXISTS clause. Formally, it answers the question “does a city exist with a store that is not in Stores”? But it is easier to say that a nested NOT EXISTS answers the question “is ***x*** TRUE for all ***y***?”

In MySQL 8.0.19 and later, you can also use NOT EXISTS or NOT EXISTS with [TABLE](https://dev.mysql.com/doc/refman/8.0/en/table.html) in the subquery, like this:

SELECT column1 FROM t1 WHERE EXISTS (TABLE t2);

The results are the same as when using SELECT \* with no WHERE clause in the subquery.