

LAB02: Subquery (Solution)

Submission:

- Submit a lab file named “int205_lab02_xxxxxxxxxxx.docx/.pdf” into the LEB2 system. xxxxxxxxxxxx = your student id

Due Date & Time:

- Lecturer will inform the LAB02 due date and time in lab class.
-

Guideline for writing sub-query

- The subquery (inner query) executes once before the main query (outer query).
- The result of the subquery is used by the main query.
- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- Use single-row operators ($>$, $<$, $=$, \leq , \geq , \neq) with single-row subqueries (return ONLY one value), and use multiple-row operators (IN, $>\text{ANY}$, $>\text{ALL}$, <ANY , <ALL) with multiple-row subqueries (return one or more values).
- A common problem with subqueries occurs when no rows are returned by the subquery. So, the main query also returns no rows.
- The ORDER BY clause in the subquery is NOT needed unless you are performing Top-N analysis.

Type 1 – Nested Subquery

- Database evaluates the whole query in two steps:
 - First, execute the subquery (inner query).
 - Second, use the result of the subquery in the parent statement (outer query).

```
select customername
from customers
where creditlimit > ( select creditlimit
                        from customers
                        where customername = 'Land of Toys Inc.')
```

Type 2 - Correlated Subquery

- Database evaluated once for each row processed by the parent statement.
 - This operation is used when a subquery refers to a column from a table in an outer query.
 - The unqualified columns in the subquery are resolved by looking in the tables named in the inner query and then in the tables named in the outer query.

```

select lastname
from employees e
where exists (select *
               from offices
              where officecode = e.officecode)

```

The Syntax of SELECT statement:

Documentation: <https://dev.mysql.com/doc/refman/8.0/en/select.html>

Note: The MySQL error code 1064 is a syntax error. This means the reason there's a problem is because MySQL doesn't understand what you're asking it to do.

Switch to SQL Editor

- You should specify the classicmodels database before writing SQL statements using the following command:

```
USE db_name;
```

The USE statement tells MySQL to use the named database as the default (current) database for subsequent statements. This statement requires some privilege for the database or some object within it.

The example of the result screen: Please insert your student id in the first line of the query section

MySQL Workbench interface showing the results of a query:

```

1 -- 64199999999
2 • use classicmodels;
3 • 1. List customer name of all customers whose name starts with the letter 'S'.
4 • select customername
5   from customers
6   where customername like 'S%';

```

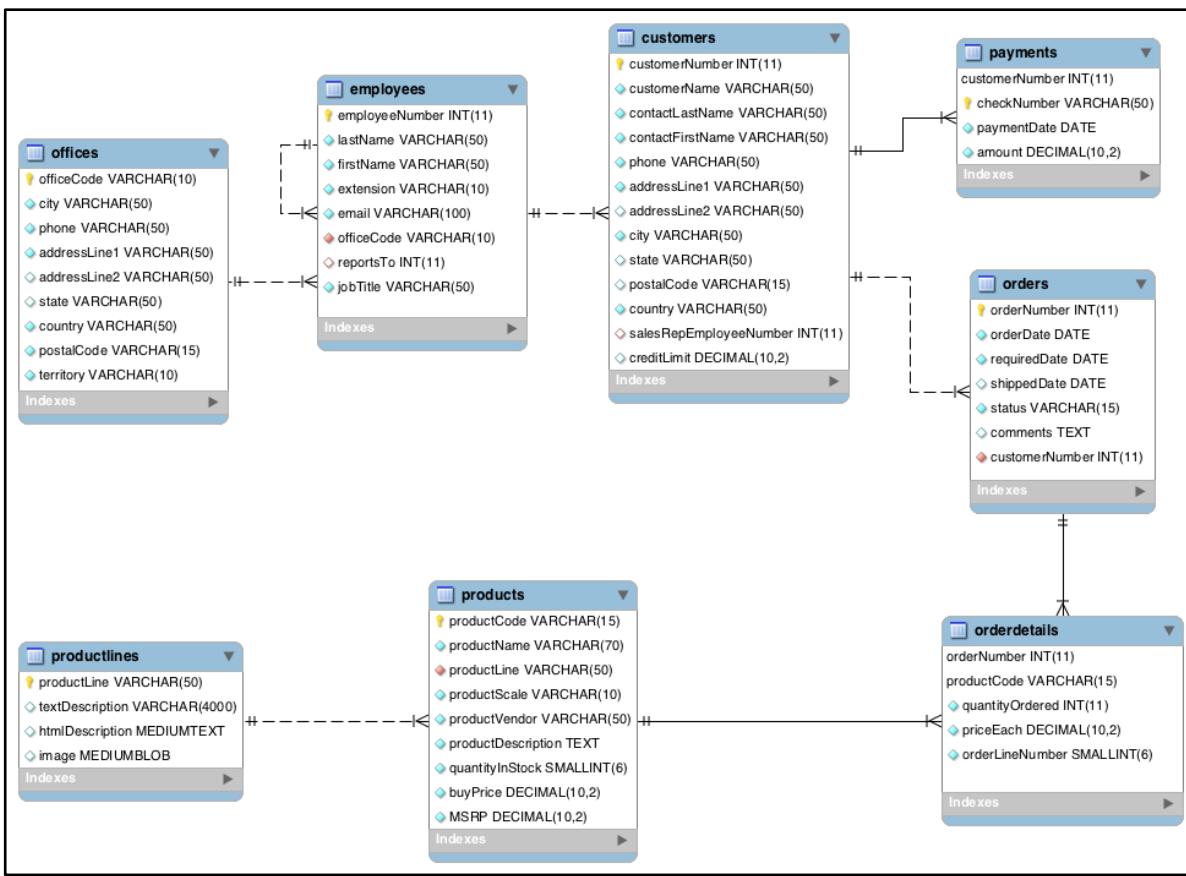
Result Grid:

customername
Signal Gift Stores
Saveley & Henriot, Co.
Souveniers And Things Co.
Schuylar Imports
Stylish Desk Decor, Co.
Suominen Souvenirs
SAR Distributors, Co
Salzburg Collectables
Stuttgart Collectable Exchange
Scandinavian Gift Ideas
Super Scale Inc.

Action Output:

Action	Time	Response
45	14:23:24	select customername from customers where customername like 'S%' LIMIT 0, 1000 12 row(s) returned

The ER diagram for the classicmodels.



Note: The MSRP is “Manufacturer's suggested retail price” (ราคาขายปลีกแนะนำของผู้ผลิต).

Task 1: Using the “classicmodels” schema and write SQL statements to answer the following questions.

3.1 List the customer name of all customers who live in the same country of customer named “Mini Classics”. Sort the results in ascending order by customer name.

-- Capture the SQL statement + Result Screen and place here

-- A1

```
select customername
from customers
where country = (select country
                  from customers
                  where customername = 'Mini Classics')
order by customername;
```

--A2

```
select customername
from customers
where country IN (select country
                  from customers
                  where customername = 'Mini Classics')
order by 1;
```

-- 36 rows returned.

```

1 •  use classicmodels;
2 -- 63199999999
3 -- 1.
4 •  select customername
5   from customers
6   where country IN (select country
7     from customers
8       where customername = 'Mini Classics')
9   order by 1;

```

Result Grid

customername
American Souvenirs Inc
Auto-Moto Classics Inc.
Boards & Toys Co.
Cambridge Collectables Co.
Classic Gift Ideas, Inc
Classic Legends Inc.
Collectable Mini Desians Co.
customers 1

Action Output

Time	Action	Response
52 21:32:09	select customername from customers where country IN (select country from customers...	36 row(s) returned

3.2 List the customer name of all customers who live in the same country of customer named “Mini Classics” and their customer names start with “Mini”.

--A1

```

select customername
from customers
where country IN (select country
                  from customers
                  where customername = 'Mini Classics')
and customername like 'Mini%';

```

-- A2

```

select customername
from customers c
where country IN (select country
                  from customers
                  where customername = 'Mini Classics'
                  and c.customername like 'Mini%');

```

-- 4 rows returned

```

10 -- 63199999999
11 --
12 • select customername
13   from customers
14   where country IN (Select country
15     from customers
16     where customername = 'Mini Classics')
17   and customername like 'Mini%';
18
19

```

Result Grid

customername
Mini Gifts Distributors Ltd.
Mini Wheels Co.
Mini Classics
Mini Creations Ltd.

Action Output

Time	Action	Response
52 21:32:00	select customername from customers where country IN (Select country	from... 0 row(s) returned
53 21:33:55	select customername from customers where country IN (Select country	from... 4 row(s) returned

3.3 List the product name and quantity in stock of the product that has the maximum quantities in stock.

```

select productname, quantityInStock
from products
where quantityinstock = (select max(quantityinstock) from products);
-- 1 row returned.

```

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** The current tab is "SQL File 3*". The code entered is:


```

25 -- 63199999999
26 -- 3
27 • select productname, quantityInStock
28 from products
29 where quantityinstock = (select max(quantityinstock) from products);
30
31
32
33
      
```
- Result Grid:** The result of the query is displayed in a grid:

productname	quantityInStock
2002 Suzuki XREO	9997
- Action Output:** A log of actions taken:

Time	Action	Response
21:41:33	select productname, quantityInStock from products where quantityinstock = (select max(quantityinstock) from products);	1 row(s) returned

3.4 List the order number and the total amount of sales of all orders that their total amount of sales is more than the total amount of sales order number 10103. Name the total amount of sales column to “total_amount”.

```

select ordernumber, sum(quantityordered*priceeach) as total_amount
from orderdetails
group by ordernumber
having sum(quantityordered*priceeach) > (select sum(quantityordered*priceeach)
                                             from orderdetails
                                             where ordernumber = 10103);
-- 34 rows returned
      
```

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:


```

30 -- 63199999999
31 --
32 • select ordernumber, sum(quantityordered*priceeach) as total_amount
33   from orderdetails
34   group by ordernumber
35   having sum(quantityordered*priceeach) > (select sum(quantityordered*priceeach)
36     from orderdetails
37     where ordernumber = 10103);
38
39
      
```
- Result Grid:** Displays the results of the query, showing the ordernumber and total_amount for various orders. The results are as follows:

ordernumber	total_amount
10105	53959.21
10106	52151.81
10108	51001.22
10122	50824.66
10126	57131.92
10127	58841.35

Result 5
- Action Output:** Shows the history of actions taken, including the execution of the query.

3.5 List the customer name and the employee last name of all customers that their sales rep employee worked in the office located in London city. [Write two SQL statements using each type of subquery => Type I: Nested Subquery and Type II: Correlated Subquery subquery]

-- A1: Type I: Nested Subquery

```

select customername, lastname as empname
from customers c join employees e
on c.salesRepEmployeeNumber = e.employeeNumber
where officecode IN (select officecode
                     from offices
                     where city = 'London');
      
```

-- A2: Type II: Correlated Subquery

```

select customerName, lastName
from customers c join employees e
on c.salesRepEmployeeNumber = e.employeeNumber
where exists (select *
              from offices
              where officeCode = e.officeCode
              and city = "London");
      
```

-- A3

```

select c.customerName, e.lastname
from customers c,(select employeeNumber,lastName
                  from employees
                  where officeCode = (select officeCode
                                      from offices
                                      where city = 'London')) e
      
```

where c.salesRepEmployeeNumber = e.employeeNumber;

-- 17 rows returned.

-- 6319999999
 39 -- 5
 40 • select customername, lastname as empname
 41 from customers c join employees e
 42 on c.salesRepEmployeeNumber = e.employeeNumber
 43 where officecode IN (select officecode
 44 from offices
 45 where city = 'London');
 46
 47

customername	empname
Toys of Finland, Co.	Bott
AV Stores, Co.	Bott
UK Collectables, Ltd.	Bott
giftsbymail.co.uk	Bott
Oulu Toy Supplies, Inc.	Bott
Stylish Desk Decors, Co.	Bott

Result 6

Action Output	Time	Action	Response
56	21:44:10	select orderNumber, sum(quantityOrdered * unitPrice) as totalAmount from orderDetails group by...	34 row(s) returned
57	21:45:34	select customername, lastname as empname from customers c join employees e on c.salesRepE...	17 row(s) returned

Query Completed

3.6 List all cities that are both an office location and a customer location. Remove the duplicate answer. [Write two SQL statements using each type of subquery => Type I: Nested Subquery and Type II: Correlated Subquery subquery]

-- A1: Type II: Correlated Subquery

```
select DISTINCT city
from customers c
where exists (select *
              from offices o
              where c.city=o.city);
```

-- A2: Type I: Nested Subquery

```
select DISTINCT city
from customers
where city IN (select city from offices);
```

-- A3: Type II: Correlated Subquery

```
select DISTINCT city
from customers
where city = (select city
              from offices
              where offices.city = customers.city);
```

-- A4: Type II: Correlated Subquery

```
select DISTINCT city
from customers
where exists (select city
               from offices
              where offices.city = customers.city);
```

-- A5: Type II: Correlated Subquery

```
select distinct city
from offices o
where exists (select city
               from customers
              where city = o.city);
```

-- 5 rows returned.

city
San Francisco
NYC
Paris
London
Boston

Action Output

Time	Action	Response
21:52:31	select distinct city from customers c where exists (select * from offices o where c.c...	5 row(s) returned

3.7 List all cities where customers who do not live in the same city of their sales rep employee's office city. Remove the duplicate answer. [Write two SQL statements using each type of subquery]

=> Type I: Nested Subquery and Type II: Correlated Subquery subquery]

-- A1: Type II: Correlated Subquery

```
select distinct city
from customers c join employees e
on c.salesRepEmployeeNumber = e.employeeNumber
where not exists (select city
                  from offices
                 where officeCode = e.officeCode
                   and city = c.city);
```

-- A2: Type II: Correlated Subquery

```
select distinct city
from customers c join employees e
on c.salesRepEmployeeNumber = e.employeeNumber
where city not in (select city
                    from offices
                    where officeCode = e.officeCode
                    and city = c.city);
```

-- 73 rows returned.

The screenshot shows the MySQL Workbench interface. The title bar says "dreamhome_er.mwb - MySQL Workbench". The left sidebar shows the schema "classicmodels" with the "customers" table selected. The main area contains the following SQL code:

```
53 -- 63199999999
54 --
55 • select distinct city
56   from customers c join employees e
57   on c.salesRepEmployeeNumber = e.employeeNumber
58   where not exists (select city
59     from offices
60     where officeCode = e.officeCode
61     and city = c.city);
62
```

The results grid below shows the output of the query:

city
San Rafael
Burlingame
San Jose
Brisbane
Las Vegas
Pasadena

At the bottom, the status bar shows "Result 10" and "Action Output". The log pane shows two entries:

- 60 22:00:48 select distinct city from customers c join employees e on c.salesRepEmployeeNumber = e.employeeNumber 73 row(s) returned
- 61 22:01:32 select distinct city from customers c join employees e on c.salesRepEmployeeNumber = e.employeeNumber 73 row(s) returned

3.8 List the customer name of all customers who have a credit limit greater than all average credit limits of customers in each city.

-- Type I: Nested Subquery

```
select customername
from customers
where creditlimit > ALL
      (select avg(creditlimit)
       from customers
       group by city);
```

-- 1 row returned.

```

71
72 -- 63199999999
73 --
74 • select customername
75   from customers
76   where creditlimit > ALL
77   (select avg(creditlimit)
78     from customers
79       group by city);
80

```

Result Grid

customername
Euro+ Shopping Channel
customers 17

Action Output

Time	Action	Response
22:10:27	select customername from customers where creditlimit > (select avg(creditlimit) ...	(select avg(creditlimit)... 1 row(s) returned

3.9 List the customer name of all customers who have a credit limit greater than their average credit limits of customers in their cities.

-- A1: Type II: Correlated Subquery

```

select customername
from customers c
where creditlimit >
    (select avg(creditlimit)
     from customers
     where city =c.city
     group by city);

```

-- A2:

```

select c.customerName
from customers c join (select city, avg(creditLimit) as creditLimit_avg
                      from customers
                      group by city) x
on c.city = x.city
where c.creditLimit > x.creditLimit_avg;

```

-- 21 rows returned

The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Home, 63199999999, MySQL Model*, EER Diagram.
- Schemas:** classicmodels selected.
- Tables:** customers selected.
- Columns:** customerNumber, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2.
- SQL Editor:** Contains the following query:


```

80 -- 63199999999
81 -- 9
82 • select customername
83   from customers c
84   where creditlimit >
85       (select avg(creditlimit)
86        from customers
87        where city =c.city
88        group by city);
89
      
```
- Result Grid:** Displays the results of the query, showing 18 rows of customer names.
- Action Output:** Shows two log entries:

Action	Time	Response
select customername from customers where creditlimit > 9	22:10:27	(select avg(creditlimit)... 21 row(s) returned)
select customername from customers c where creditlimit >	22:12:13	(select avg(creditlimit)... 21 row(s) returned)

3.10 List the products that were never ordered by any customers.

-- A1: Type II: Correlated Subquery

```

select productname
from products p
where NOT EXISTS (select *
                   from orderdetails
                   where productcode = p.productcode);
      
```

-- A2: Type I: Nested Subquery

```

select productname
from products p
where productcode NOT IN (select productcode
                           from orderdetails );
      
```

-- 1 rows returned.

The screenshot shows the MySQL Workbench interface with the following details:

- Top Bar:** dreamhome_er.mwb - MySQL Workbench
- Left Sidebar (SCHEMAS):** classicmodels
- Tables:** customers
- Columns:** customerNumber, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit
- SQL Editor:** Contains the following SQL code:

```
89 -- 63199999999
90 -- 10
91 • select productname
92   from products p
93   where NOT EXISTS (select *
94     from orderdetails
95     where productcode = p.productcode);
96
97
98
```
- Result Grid:** Shows the result of the query, displaying a single row: **1985 Toyota Supra**.
- Action Output:** Shows the execution history with two entries:

Time	Action	Response
22:12:03	select customername from customers c where creditlimit >	1 row(s) returned
22:16:31	select productname from products p where NOT EXISTS (select * from orderdetails...	1 row(s) returned