LAB02: Subquery (Solution)

Submission:

- Submit a lab file named "int205_lab02_xxxxxxxxxxx.docx/.pdf" into the LEB2 system. xxxxxxxxxxx = 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 (>, <, =, <=, >=, <>) with single-row subqueries (return ONLY one value), and use multiple-row operators (IN, >ANY, >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

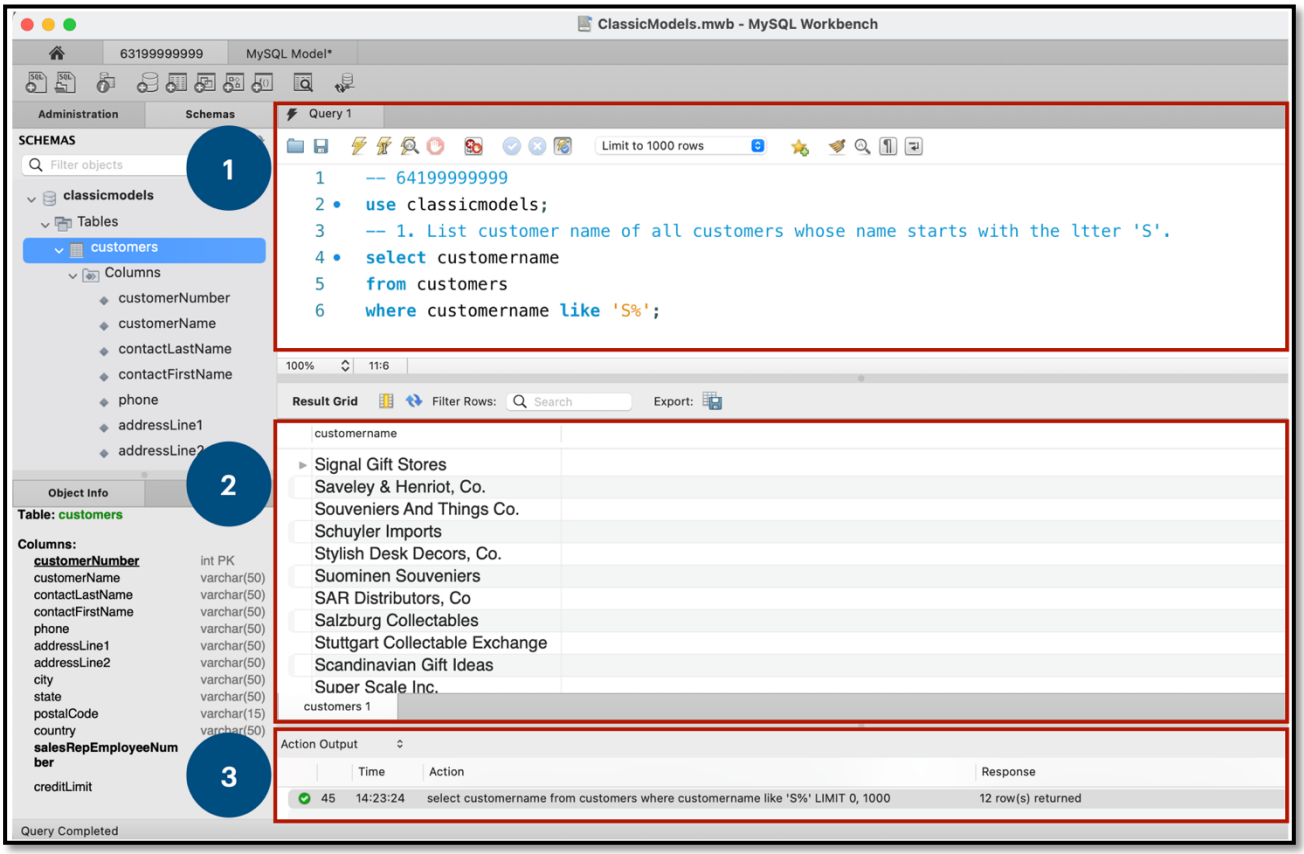
Note: The MySQL error code 1064 is a syntax error. This means the reason there's a problem is because MySQL doesn't understand what you're asking it to do.

Switch to SQL Editor

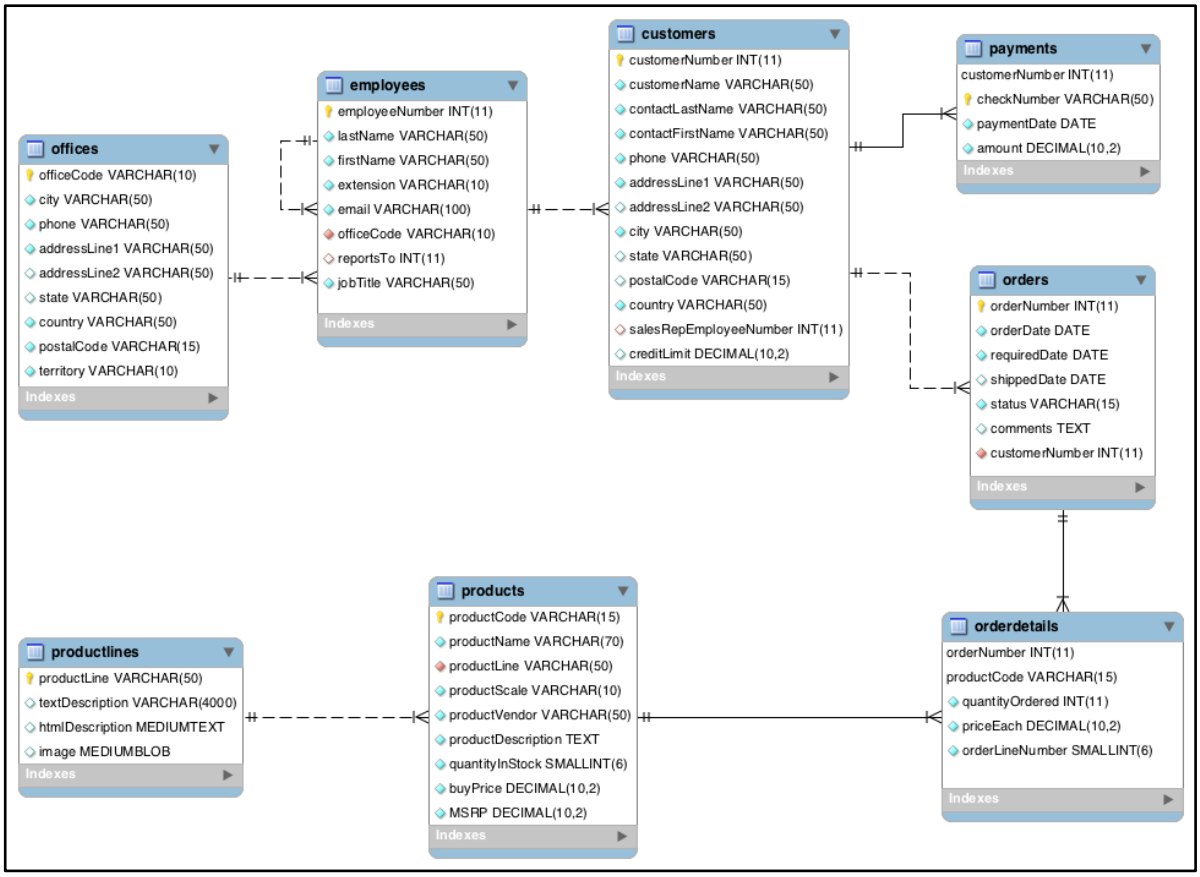
- You should specify the classicmodels database before writing SQL statements using the following command:
  USE db_name;

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



The ER diagram for the classicmodels.

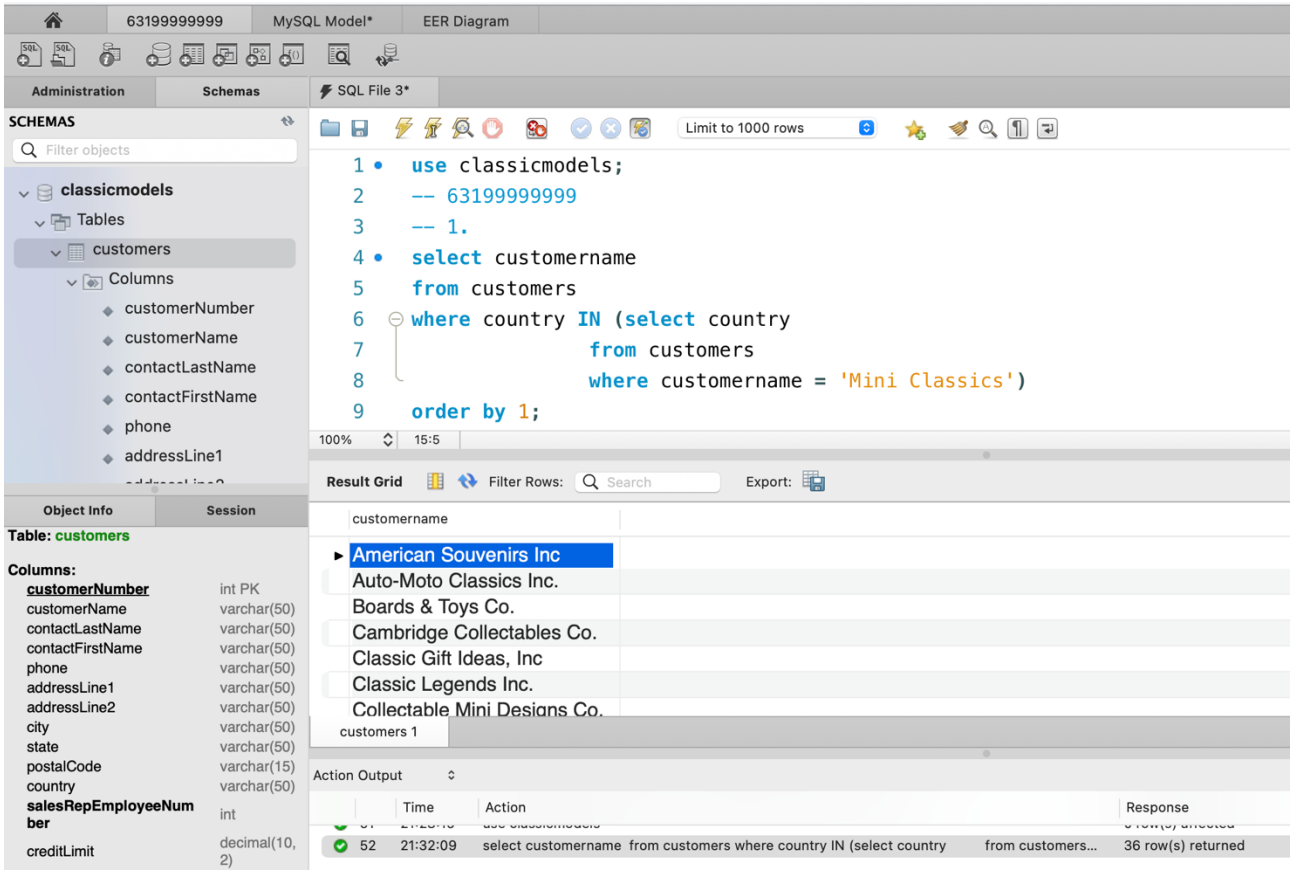


Note: The MSRP is “Manufacturer's suggested retail price” (ราคาขายปลีกแนะนำของผู้ผลิต).

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

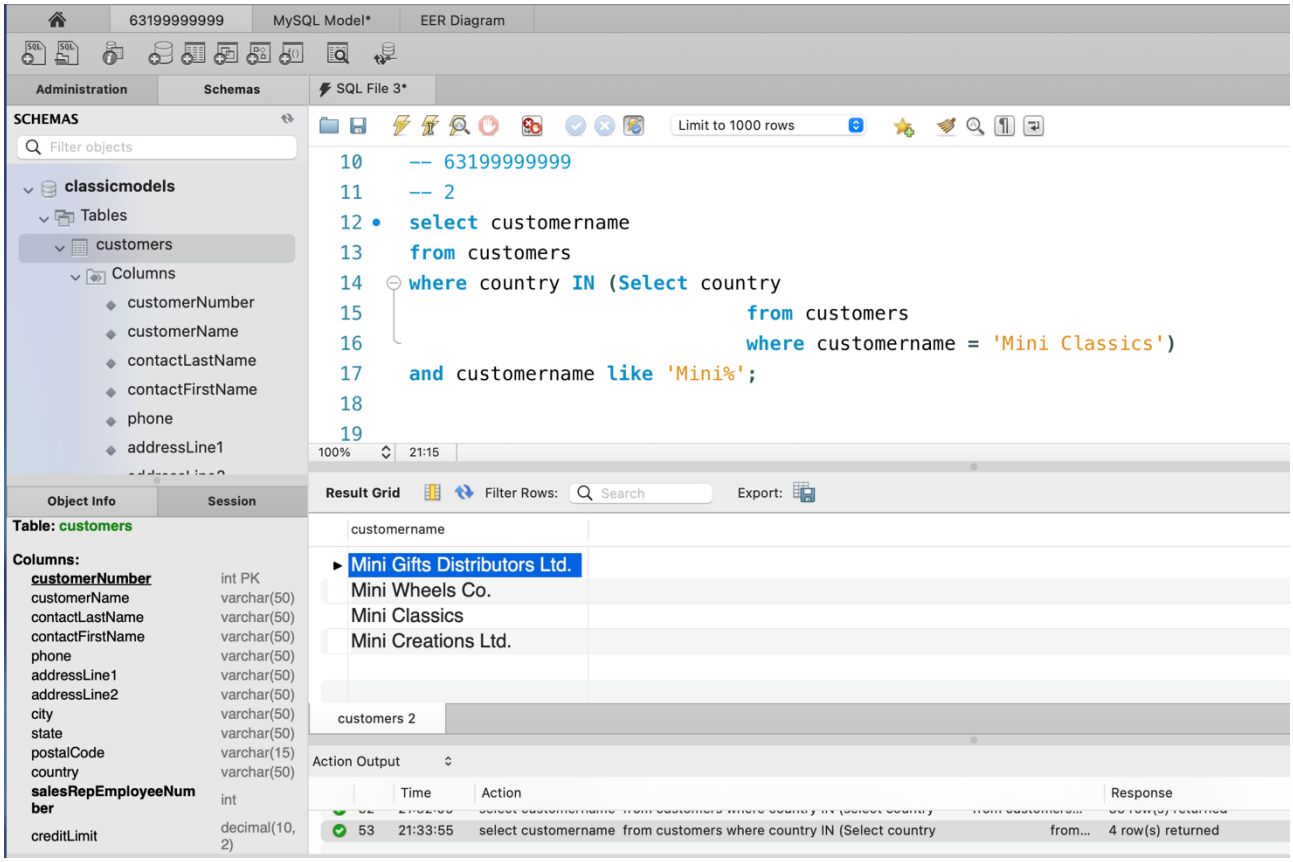
3.1 List the customer name of all customers who live in the same country of customer named “Mini Classics”. Sort the results in ascending order by customer name.

```
-- 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.
```



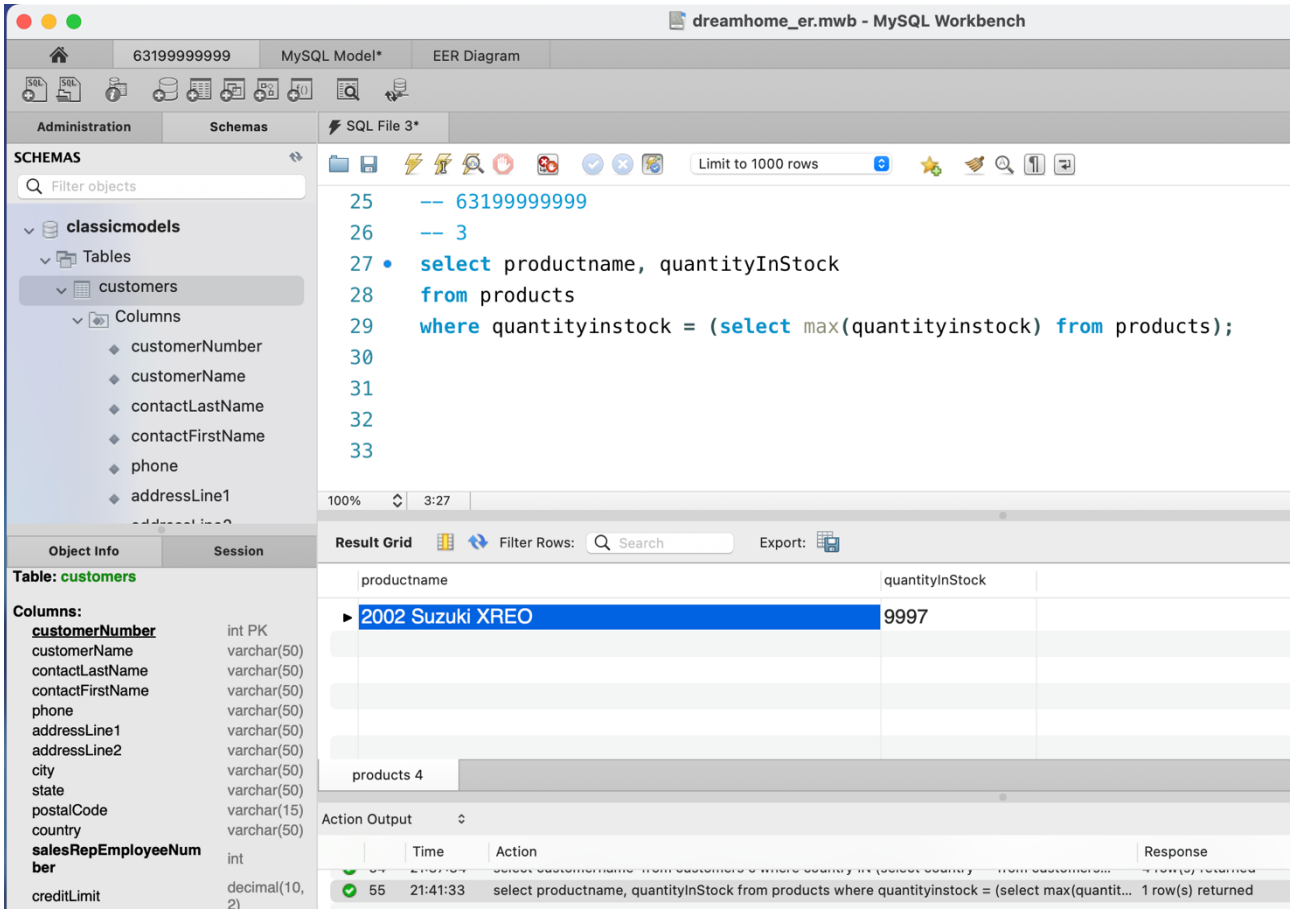
3.2 List the customer name of all customers who live in the same country of customer named “Mini Classics” and their customer names start with “Mini”.

```
--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
```



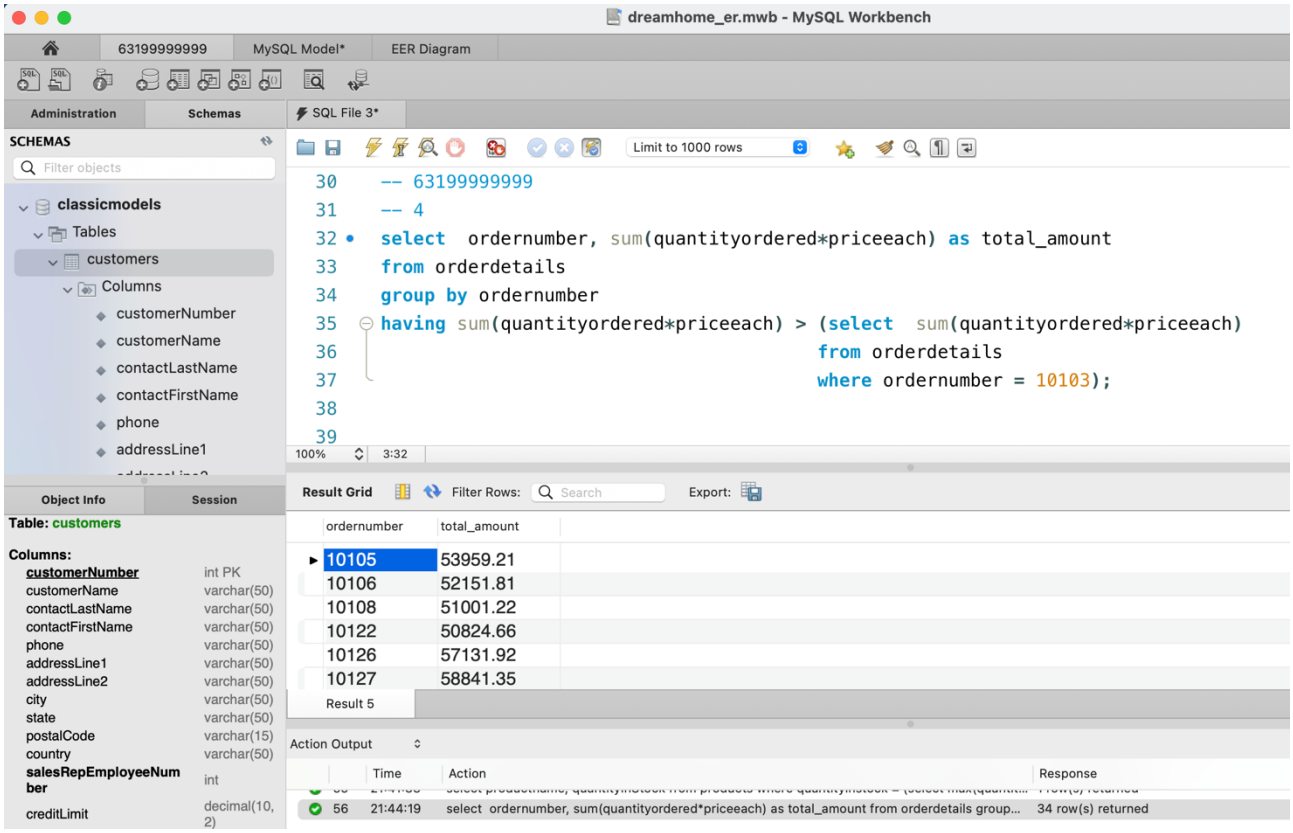
3.3 List the product name and quantity in stock of the product that has the maximum quantities in stock.

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



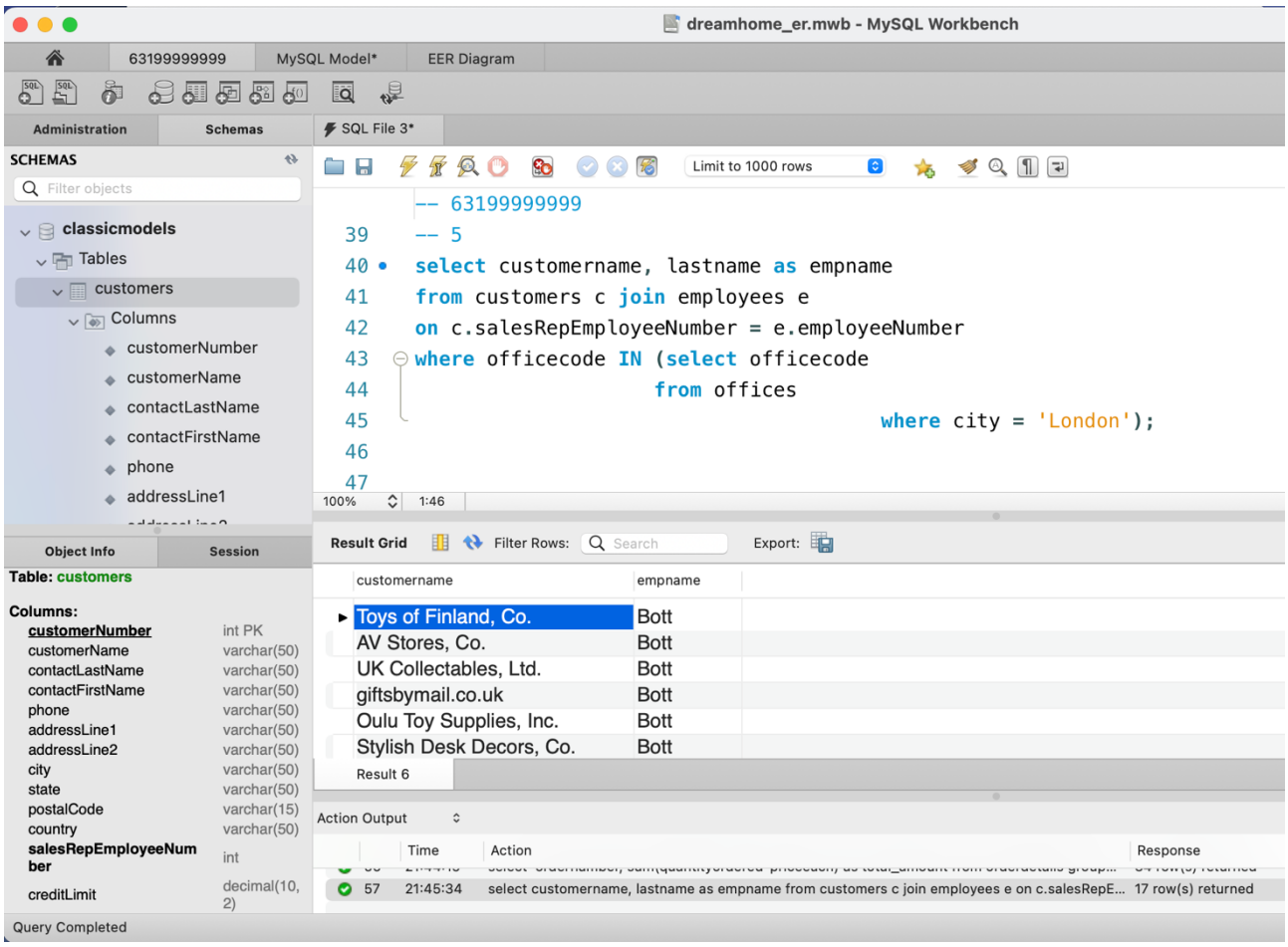
3.4 List the order number and the total amount of sales of all orders that their total amount of sales is more than the total amount of sales order number 10103. Name the total amount of sales column to “total_amount”.

```
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
```



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.
```

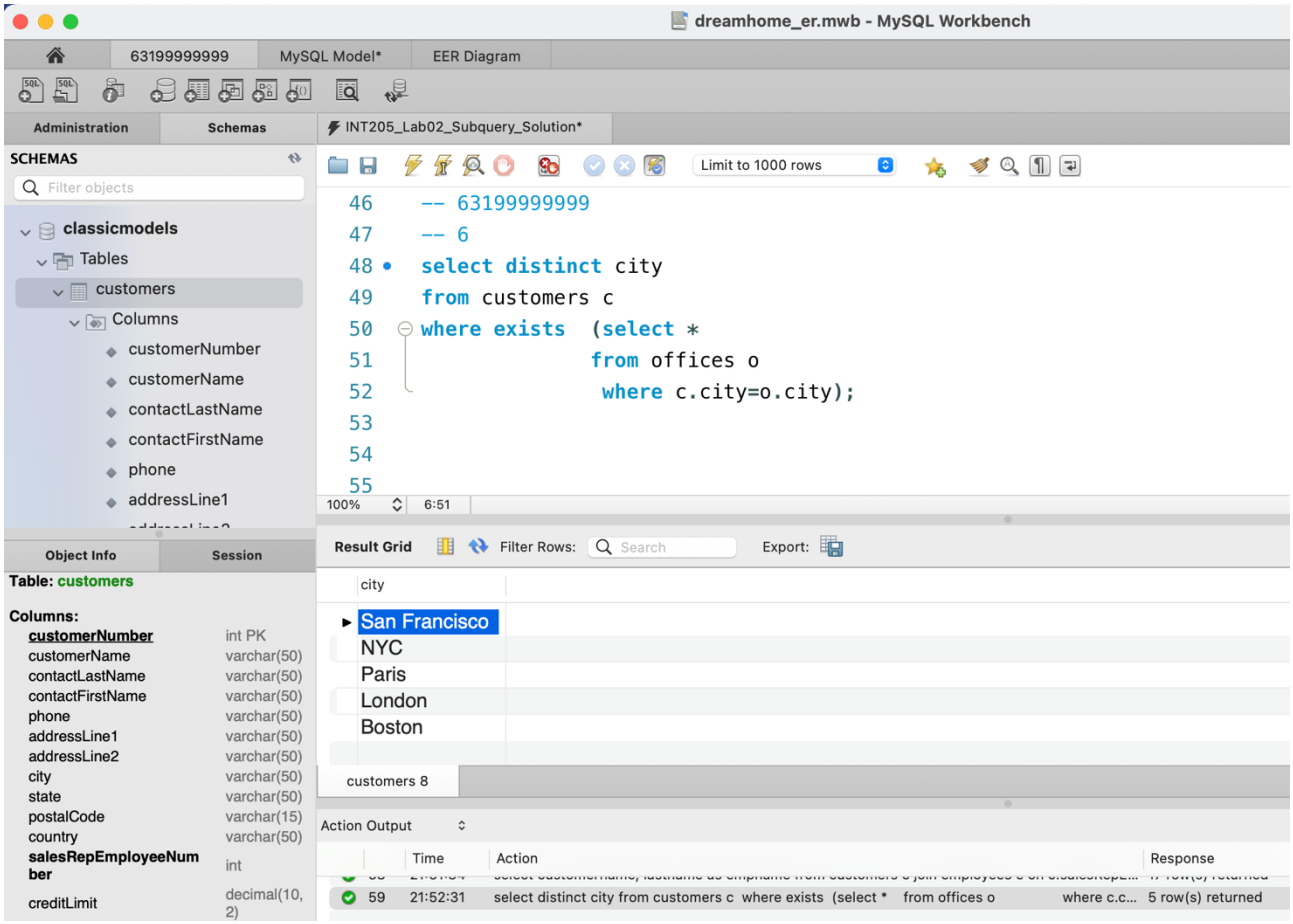


3.6 List all cities that are both an office location and a customer location. Remove the duplicate answer. [Write two SQL statements using each type of subquery => Type I: Nested Subquery and Type II: Correlated Subquery subquery]

```
-- 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);
```

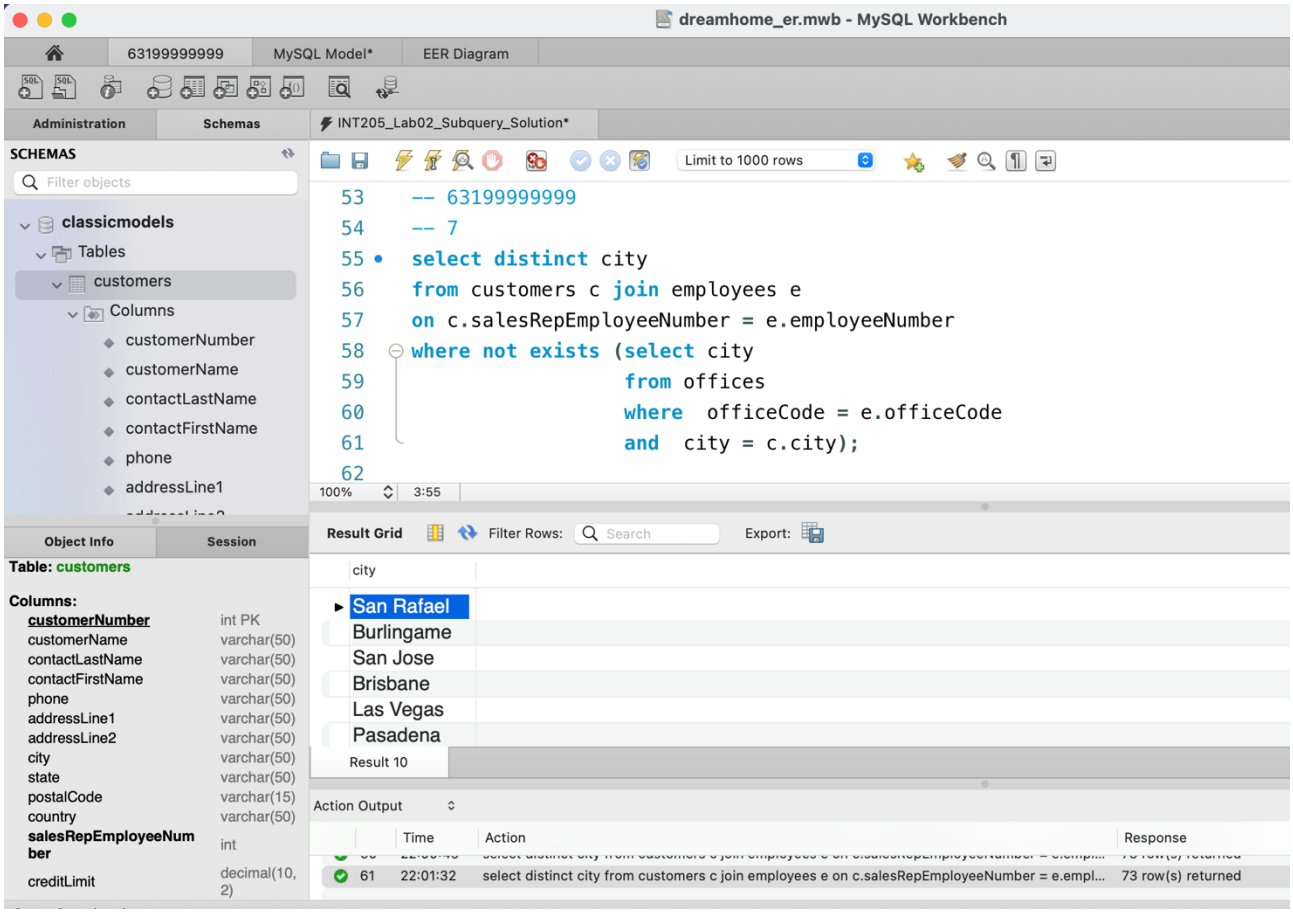
```
-- A4: Type II: Correlated Subquery
select DISTINCT city
from customers
where exists (select city
              from offices
              where offices.city = customers.city);
-- A5: Type II: Correlated Subquery
select distinct city
from offices o
where exists (select city
              from customers
              where city = o.city);
-- 5 rows returned.
```



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]

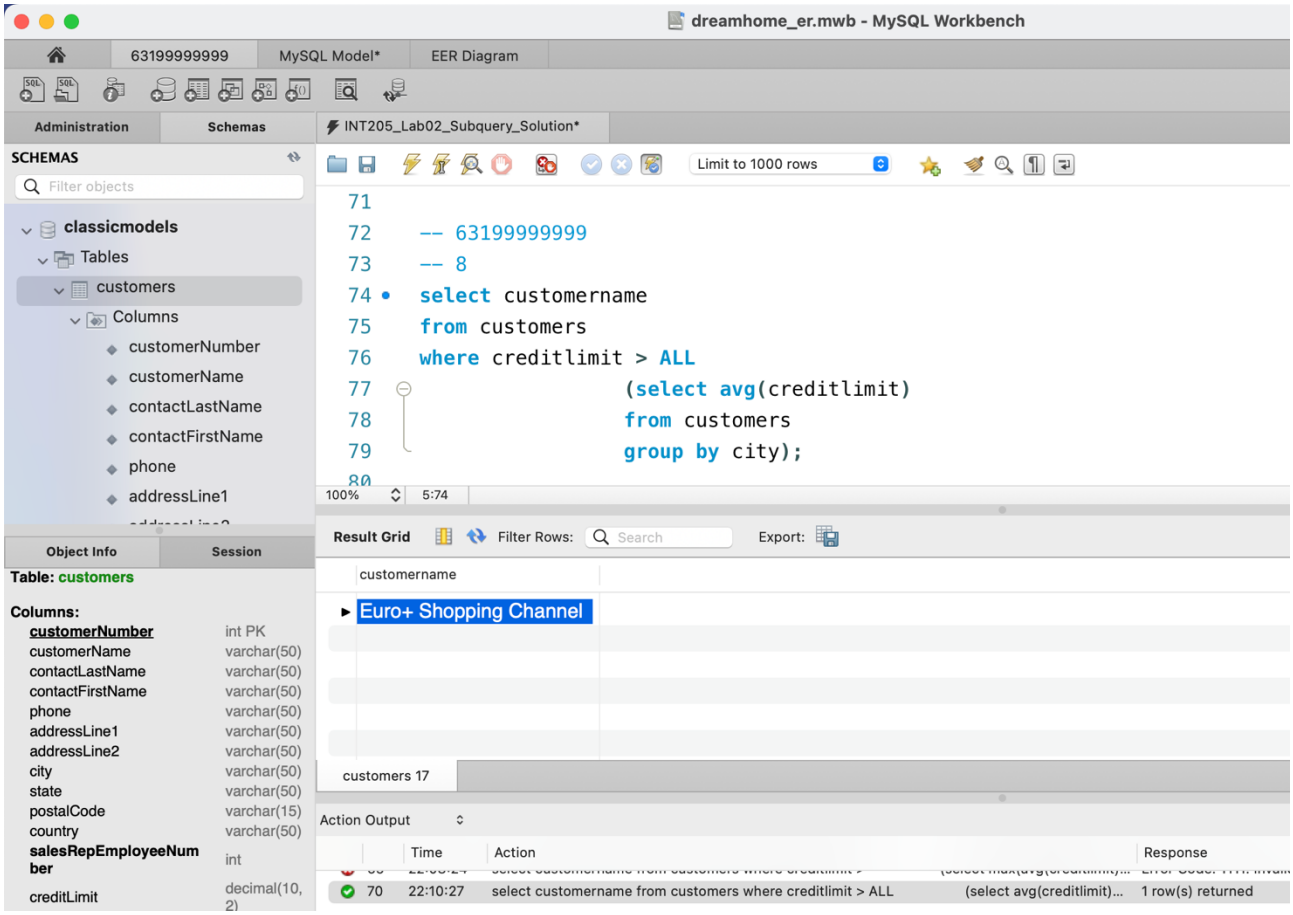
```
-- A1: Type II: Correlated Subquery
select distinct city
from customers c join employees e
on c.salesRepEmployeeNumber = e.employeeNumber
where not exists (select city
                  from offices
                  where  officeCode = e.officeCode
                  and  city = c.city);
```

```
-- 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.
```



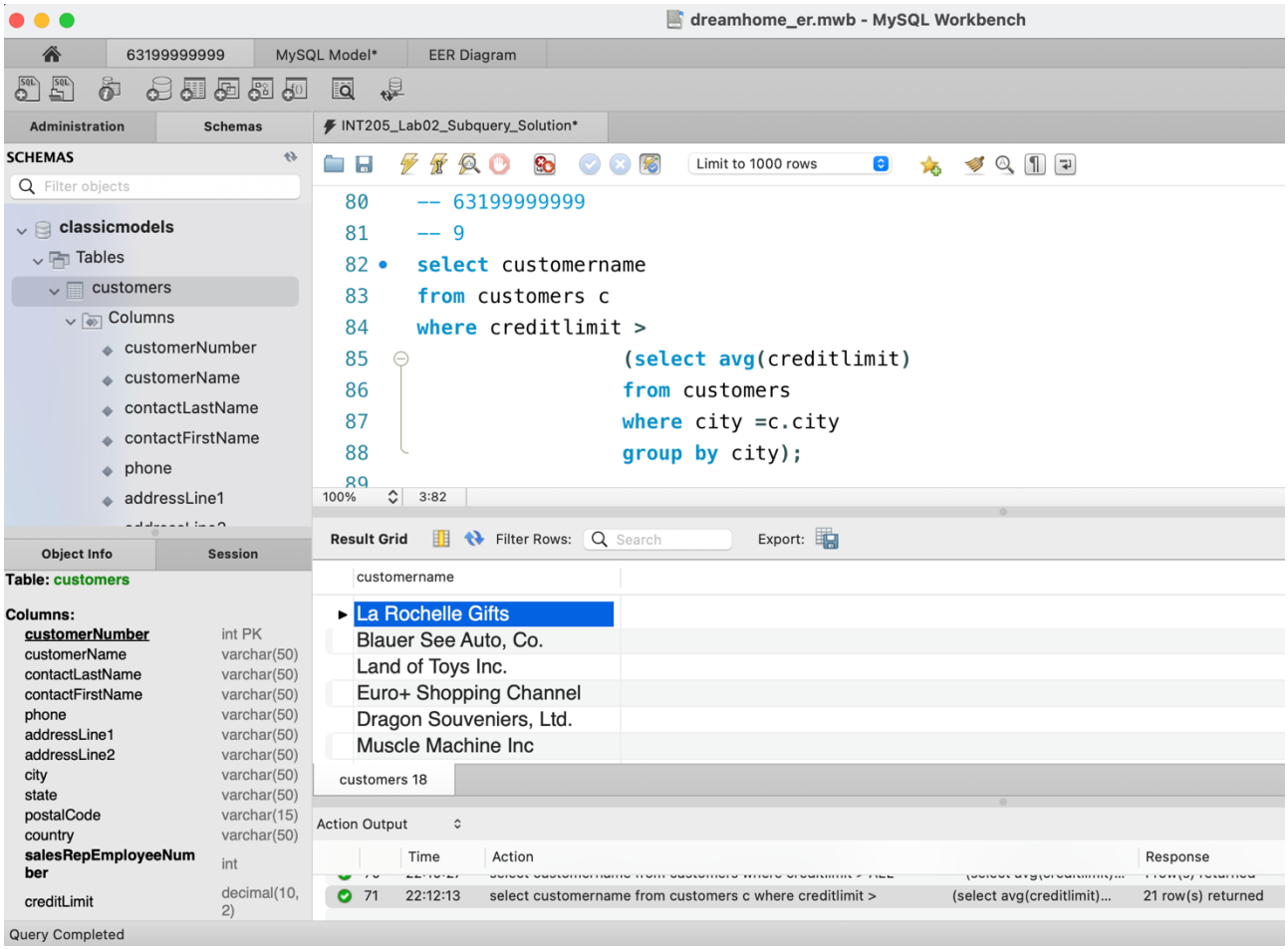
3.8 List the customer name of all customers who have a credit limit greater than all average credit limits of customers in each city.

```
-- Type I: Nested Subquery
select customername
from customers
where creditlimit > ALL
        (select avg(creditlimit)
        from customers
        group by city);
-- 1 row returned.
```



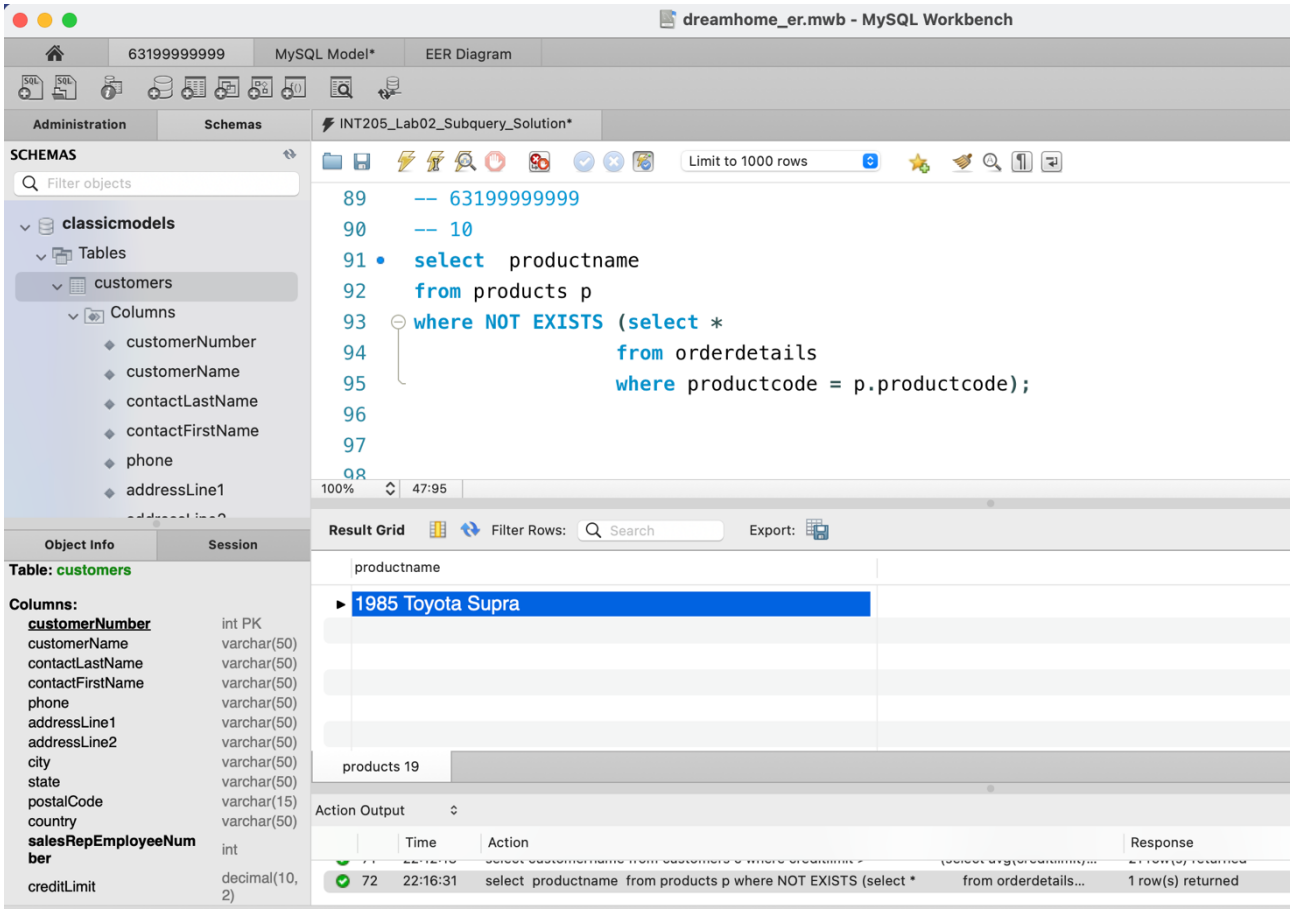
3.9 List the customer name of all customers who have a credit limit greater than their average credit limits of customers in their cities.

```
-- 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
```



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.
```


dreamhome_er.mwb - MySQL Workbench
63199999999
MySQL Model*
EER Diagram
Administration
Schemas
INT205_Lab02_Subquery_Solution*
SCHEMAS
Filter objects
classicmodels
Tables
customers
Columns
customerNumber
customerName
contactLastName
contactFirstName
phone
addressLine1
Limit to 1000 rows
-- 63199999999
-- 10
select productname
from products p
where NOT EXISTS (select *
from orderdetails
where productcode = p.productcode);
100%
47:95
Result Grid
Filter Rows:
Search
Export:
productname
1985 Toyota Supra
products 19
Object Info
Session
Table: customers
Columns:
customerNumber int PK
customerName varchar(50)
contactLastName varchar(50)
contactFirstName varchar(50)
phone varchar(50)
addressLine1 varchar(50)
addressLine2 varchar(50)
city varchar(50)
state varchar(50)
postalCode varchar(15)
country varchar(50)
salesRepEmployeeNumber int
creditLimit decimal(10, 2)
Action Output
Time
Action
Response
72 22:16:31 select productname from products p where NOT EXISTS (select * from orderdetails... 1 row(s) returned