SQL Subqueries - Lab Assignment #2

Introduction

Now that you've seen how subqueries work, it's time to get some practice writing them! Not all of the queries will require subqueries, but all will be a bit more complex and require some thought and review about aggregates, grouping, ordering, filtering, joins and subqueries. Good luck!

Objectives

You will be able to:

- Write subqueries to decompose complex queries

CRM Database ERD

Once again, here's the schema for the CRM database you'll continue to practice with.

No description has been provided for this image

Connect to the Database

As usual, start by importing the necessary packages and connecting to the database `data2.sqlite` in the data folder.

In [16]:
```
# Run this cell without changes
import sqlite3
import pandas as pd

conn = sqlite3.Connection("data/data.sqlite")
```

In [4]:
```
# Your code here; create the connection
conn = sqlite3.Connection("data/data.sqlite")
```

Q.1 Write an Equivalent Query using a Subquery

The following query works using a `JOIN`. Rewrite it so that it uses a subquery

instead.

```
SELECT
    customerNumber,
    contactLastName,
    contactFirstName
FROM customers
JOIN orders
    USING(customerNumber)
WHERE orderDate = '2003-01-31'
;
```

In [25]:
```
# Your code here
q1 = """
SELECT c.customerNumber,
       c.contactLastName,
       c.contactFirstName
FROM customers c
WHERE EXISTS (
    SELECT 1
    FROM orders o
    WHERE o.customerNumber = c.customerNumber
      AND o.orderDate = '2003-01-31'
);
"""
pd.read_sql(q1, conn)
```

Out[25]:

	customerNumber	contactLastName	contactFirstName
0	141	Freyre	Diego

Q.2 Select the Total Number of Orders for Each Product Name

Sort the results by the total number of items sold for that product.

In [26]:
```
# Your code here
q2 = """
SELECT p.productName,
       (
         SELECT COUNT(DISTINCT od.orderNumber)
         FROM orderdetails od
         WHERE od.productCode = p.productCode
       ) AS totalOrders,
       (
         SELECT SUM(od.quantityOrdered)
         FROM orderdetails od
         WHERE od.productCode = p.productCode
       ) AS totalItemsSold
```

```
FROM products p
ORDER BY totalItemsSold DESC;
"""

q2_result = pd.read_sql(q2, conn)
q2_result
```

Out[26]:

	productName	totalOrders	totalItemsSold
0	1992 Ferrari 360 Spider red	53	1808.0
1	1937 Lincoln Berline	28	1111.0
2	American Airlines: MD-11S	28	1085.0
3	1941 Chevrolet Special Deluxe Cabriolet	28	1076.0
4	1930 Buick Marquette Phaeton	28	1074.0
...	...	...	...
105	1911 Ford Town Car	25	832.0
106	1936 Mercedes Benz 500k Roadster	25	824.0
107	1970 Chevy Chevelle SS 454	25	803.0
108	1957 Ford Thunderbird	24	767.0
109	1985 Toyota Supra	0	NaN

110 rows × 3 columns

Q.3 Select the Product Name and the Total Number of People Who Have Ordered Each Product

Sort the results in descending order.

A quick note on the SQL `SELECT DISTINCT` statement:

The `SELECT DISTINCT` statement is used to return only distinct values in the specified column. In other words, it removes the duplicate values in the column from the result set.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the unique values. If you apply the `DISTINCT` clause to a column that has `NULL`, the `DISTINCT` clause will keep only one NULL and eliminates the other. In other words, the DISTINCT clause treats all `NULL` "values" as the same value.

In [27]:
```
# Your code here
```

```
# Hint: because one of the tables we'll be joining has duplicate custo

q3 = """
SELECT p.productName,
       COUNT(DISTINCT o.customerNumber) AS totalPeopleOrdered
FROM products p
JOIN orderdetails od ON p.productCode = od.productCode
JOIN orders o ON od.orderNumber = o.orderNumber
GROUP BY p.productName
ORDER BY totalPeopleOrdered DESC;
"""

q3_result = pd.read_sql(q3, conn)
q3_result
```

Out[27]:

	productName	totalPeopleOrdered
0	1992 Ferrari 360 Spider red	40
1	Boeing X-32A JSF	27
2	1972 Alfa Romeo GTA	27
3	1952 Alpine Renault 1300	27
4	1934 Ford V8 Coupe	27
...	...	...
104	1958 Chevy Corvette Limited Edition	19
105	2002 Chevy Corvette	18
106	1969 Chevrolet Camaro Z28	18
107	1952 Citroen-15CV	18
108	1949 Jaguar XK 120	18

109 rows × 2 columns

Q.4 Select the Employee Number, First Name, Last Name, City (of the office), and Office Code of the Employees Who Sold Products That Have Been Ordered by Fewer Than 20 people.

This problem is a bit tougher. To start, think about how you might break the problem up. Be sure that your results only list each employee once.

In [28]:

```
# Your code here

q4 = """
```

```
SELECT DISTINCT e.employeeNumber,
                e.firstName,
                e.lastName,
                off.city,
                e.officeCode
FROM employees e
JOIN customers c ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders o ON c.customerNumber = o.customerNumber
JOIN orderdetails od ON o.orderNumber = od.orderNumber
JOIN offices off ON e.officeCode = off.officeCode
WHERE od.productCode IN (
    SELECT od2.productCode
    FROM orderdetails od2
    JOIN orders o2 ON od2.orderNumber = o2.orderNumber
    GROUP BY od2.productCode
    HAVING COUNT(DISTINCT o2.customerNumber) < 20
);
"""

q4_result = pd.read_sql(q4, conn)
q4_result
```

Out[28]:

	employeeNumber	firstName	lastName	city	officeCode
0	1370	Gerard	Hernandez	Paris	4
1	1501	Larry	Bott	London	7
2	1337	Loui	Bondur	Paris	4
3	1166	Leslie	Thompson	San Francisco	1
4	1286	Foon Yue	Tseng	NYC	3
5	1612	Peter	Marsh	Sydney	6
6	1611	Andy	Fixter	Sydney	6
7	1401	Pamela	Castillo	Paris	4
8	1621	Mami	Nishi	Tokyo	5
9	1323	George	Vanauf	NYC	3
10	1165	Leslie	Jennings	San Francisco	1
11	1702	Martin	Gerard	Paris	4
12	1216	Steve	Patterson	Boston	2
13	1188	Julie	Firrelli	Boston	2
14	1504	Barry	Jones	London	7

Q.5 Select the Employee Number, First Name, Last Name,

and Number of Customers for Employees Whose Customers Have an Average Credit Limit Over 15K

In [29]:
```
# Your code here

q5 = """
SELECT e.employeeNumber,
       e.firstName,
       e.lastName,
       COUNT(c.customerNumber) AS numCustomers
FROM employees e
JOIN customers c ON e.employeeNumber = c.salesRepEmployeeNumber
GROUP BY e.employeeNumber, e.firstName, e.lastName
HAVING AVG(c.creditLimit) > 15000;

"""

q5_result = pd.read_sql(q5, conn)
q5_result
```

Out[29]:

	employeeNumber	firstName	lastName	numCustomers
0	1165	Leslie	Jennings	6
1	1166	Leslie	Thompson	6
2	1188	Julie	Firrelli	6
3	1216	Steve	Patterson	6
4	1286	Foon Yue	Tseng	7
5	1323	George	Vanauf	8
6	1337	Loui	Bondur	6
7	1370	Gerard	Hernandez	7
8	1401	Pamela	Castillo	10
9	1501	Larry	Bott	8
10	1504	Barry	Jones	9
11	1611	Andy	Fixter	5
12	1612	Peter	Marsh	5
13	1621	Mami	Nishi	5
14	1702	Martin	Gerard	6

Summary

In this lesson, you got to practice some more complex SQL queries, some of which required subqueries. There's still plenty more SQL to be had though; hope you've been enjoying some of these puzzles!