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.

Connect to the Database

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

In [1]:
```
# Your code here; import the necessary packages
import sqlite3
import pandas as pd
```

In [2]:
```
# Your code here; create the connection
conn = sqlite3.Connection("data/data.sqlite")
```

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 [11]:
```
# Your code here
q1 = """
```

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

pd.read_sql(q1, conn)
```

Out[11]:

	customerNumber	contactLastName	contactFirstName
0	141	Freyre	Diego

Select the Total Number of Orders for Each Product Name

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

In [445…

```
# Your code here
q2 = """
SELECT
    productName
    , SUM(quantityOrdered)
FROM products
JOIN orderdetails
    ON products.productCode = orderdetails.productCode
GROUP BY products.productName
ORDER BY productName
;
"""


pd.read_sql(q2, conn)
```

Out[445]:

	productName	SUM(quantityOrdered)
0	18th Century Vintage Horse Carriage	907
1	18th century schooner	1011
2	1900s Vintage Bi-Plane	940
3	1900s Vintage Tri-Plane	1009
4	1903 Ford Model A	883
...	...	...
104	The Mayflower	898
105	The Queen Mary	896
106	The Schooner Bluenose	934
107	The Titanic	952
108	The USS Constitution Ship	1020

109 rows × 2 columns

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 [534…

```
# Your code here
# Hint: because one of the tables we'll be joining has duplicate customer numbers, you s
q3 = '''
WITH mergedTables AS (
    SELECT DISTINCT
        *
        , f.city AS cities
        , f.officeCode AS officeCodes
    FROM customers a
    JOIN orders b
        ON a.customerNumber = b.customerNumber
    JOIN orderdetails c
        ON b.orderNumber = c.orderNumber
    JOIN products d
        ON c.productCode = d.productCode
    JOIN employees e
        ON a.salesRepEmployeeNumber = e.employeeNumber
    JOIN offices f
        ON e.officeCode = f.officeCode
    )

SELECT DISTINCT
    productName
    , COUNT(DISTINCT customerNumber)
FROM mergedTables
GROUP BY productName
ORDER BY COUNT(DISTINCT customerNumber) DESC
;
'''

pd.read_sql(q3, conn)
```

Out[534]:

	productName	COUNT(DISTINCT customerNumber)
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

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 [473…

```
# Your code here
q4 = '''
WITH mergedTables AS (
    SELECT
        *
        , f.city AS cities
        , f.officeCode AS officeCodes
    FROM customers a
    JOIN orders b
        ON a.customerNumber = b.customerNumber
    JOIN orderdetails c
        ON b.orderNumber = c.orderNumber
    JOIN products d
        ON c.productCode = d.productCode
    JOIN employees e
        ON a.salesRepEmployeeNumber = e.employeeNumber
    JOIN offices f
        ON e.officeCode = f.officeCode
    )

SELECT *
FROM (
    SELECT
        employeeNumber
        , productName
        , lastName
        , firstName
        , cities
        , officeCodes
        , COUNT(DISTINCT customerNumber)
    FROM mergedTables
    GROUP BY productName
    HAVING COUNT(DISTINCT customerNumber) < 20
    )
GROUP BY employeeNumber
;
'''
pd.read_sql(q4, conn)
```

Out[473]:

	employeeNumber	productName	lastName	firstName	cities	officeCodes	COUNT(DISTINCT customerNumber)
0	1166	1949 Jaguar XK 120	Thompson	Leslie	San Francisco	1	18

1	1370	1958 Chevy Corvette Limited Edition	Hernandez	Gerard	Paris	4	19

Select the Employee Number, First Name, Last Name, and Number of Customers for Employees Whose Customers Have an Average Credit Limit Over 15K

In [533…

```
# Your code here
q5 = '''
WITH mergedTables AS (
    SELECT
        *
        , f.city AS cities
        , f.officeCode AS officeCodes
    FROM customers a
    JOIN orders b
        ON a.customerNumber = b.customerNumber
    JOIN orderdetails c
        ON b.orderNumber = c.orderNumber
    JOIN products d
        ON c.productCode = d.productCode
    JOIN employees e
        ON a.salesRepEmployeeNumber = e.employeeNumber
    JOIN offices f
        ON e.officeCode = f.officeCode
    )

SELECT
    employeeNumber
    , lastName
    , firstName
    , COUNT(employeeNumber) as creditLimitOver15K
FROM (
    SELECT
        employeeNumber
        , customerNumber
        , creditLimit
        , AVG(creditLimit)
        , productName
        , lastName
        , firstName
        , cities
        , officeCodes
    FROM mergedTables
    GROUP BY
        employeeNumber
        , creditLimit
    ORDER BY creditLimit DESC
    )
WHERE creditLimit > 15000
GROUP BY employeeNumber
;
'''
pd.read_sql(q5, conn)
```

Out[533]:

	employeeNumber	lastName	firstName	creditLimitOver15K
0	1165	Jennings	Leslie	6
1	1166	Thompson	Leslie	5
2	1188	Firrelli	Julie	6

3	1216	Patterson	Steve	6
4	1286	Tseng	Foon Yue	6
5	1323	Vanauf	George	8
6	1337	Bondur	Loui	6
7	1370	Hernandez	Gerard	7
8	1401	Castillo	Pamela	10
9	1501	Bott	Larry	8
10	1504	Jones	Barry	9
11	1611	Fixter	Andy	5
12	1612	Marsh	Peter	5
13	1621	Nishi	Mami	5
14	1702	Gerard	Martin	5

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!