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!

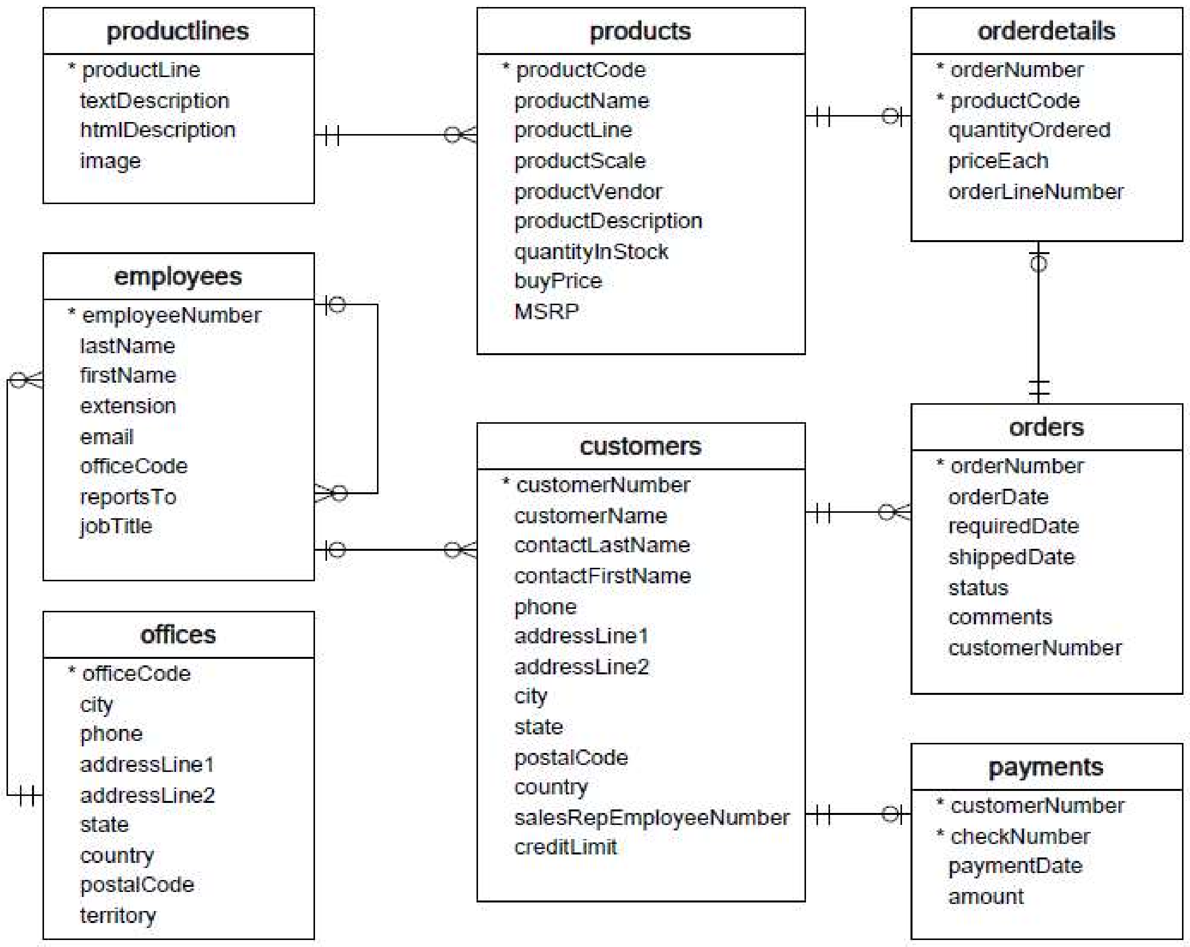
Objectives

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 [8]:
```
import sqlite3
import pandas as pd
```

In [9]:
```
conn = sqlite3.Connection("data/data2.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 [10]:
```
pd.read_sql('''
SELECT customers.customerNumber, customers.contactLastName, customers.contactFirs
FROM customers, orders
WHERE customers.customerNumber = orders.customerNumber
AND orders.orderDate = "2003-01-31";

''', conn)
```

Out[10]:
```
   customerNumber  contactLastName  contactFirstName   orderDate
0             141           Freyre             Diego  2003-01-31
```

Select the Total Number of Orders for Each Product Name

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

In [11]:
```
pd.read_sql('''
SELECT COUNT(*) as count, p.productName
FROM orderdetails o, products p
WHERE o.productCode = p.productCode
GROUP BY o.productCode;
''', conn)
```

Out[11]:

	count	productName
0	28	1969 Harley Davidson Ultimate Chopper
1	28	1952 Alpine Renault 1300
2	28	1996 Moto Guzzi 1100i
3	28	2003 Harley-Davidson Eagle Drag Bike
4	28	1972 Alfa Romeo GTA
...	...	...
104	27	The Titanic
105	27	The Queen Mary
106	28	American Airlines: MD-11S
107	28	Boeing X-32A JSF
108	27	Pont Yacht

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 [20]:
```
pd.read_sql('''
SELECT productName, COUNT(DISTINCT quantityOrdered) AS TotalCustomers
FROM products
INNER JOIN orderdetails
ON products.productCode = orderdetails.productCode
GROUP BY productName
ORDER BY TotalCustomers DESC

''', conn)
```

Out[20]:

	productName	TotalCustomers
0	1992 Ferrari 360 Spider red	28
1	The Titanic	22
2	2002 Suzuki XREO	22
3	1956 Porsche 356A Coupe	22
4	1937 Lincoln Berline	22
...	...	...
104	2001 Ferrari Enzo	15
105	1969 Corvair Monza	15
106	2002 Chevy Corvette	14
107	1958 Setra Bus	14
108	1957 Ford Thunderbird	14

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 [13]:
```
pd.read_sql('''
SELECT employeeNumber,firstName, lastName, Offices.city , Employees.officeCode
FROM Employees
JOIN Offices
ON Employees.officeCode = Offices.officeCode
JOIN Customers
ON Employees.employeeNumber = Customers.salesRepEmployeeNumber
GROUP BY employeeNumber,firstName, lastName, Offices.city , Employees.officeCode
HAVING COUNT(customerNumber)< 20;

''', conn)
```

Out[13]:

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

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

In [14]:
```
pd.read_sql('''
SELECT employeeNumber, firstName, lastName, COUNT(customerNumber)
FROM Employees
INNER JOIN Customers
ON Employees.employeeNumber = Customers.salesRepEmployeeNumber
GROUP BY employeeNumber, firstName, lastName
HAVING AVG(creditLimit) > 15000;



''', conn)
```

Out[14]:

	employeeNumber	firstName	lastName	COUNT(customerNumber)
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!