Join Statements - Lab

Introduction

In this lab, you'll practice your knowledge of `JOIN` statements, using various types of joins and various methods for specifying the links between them.

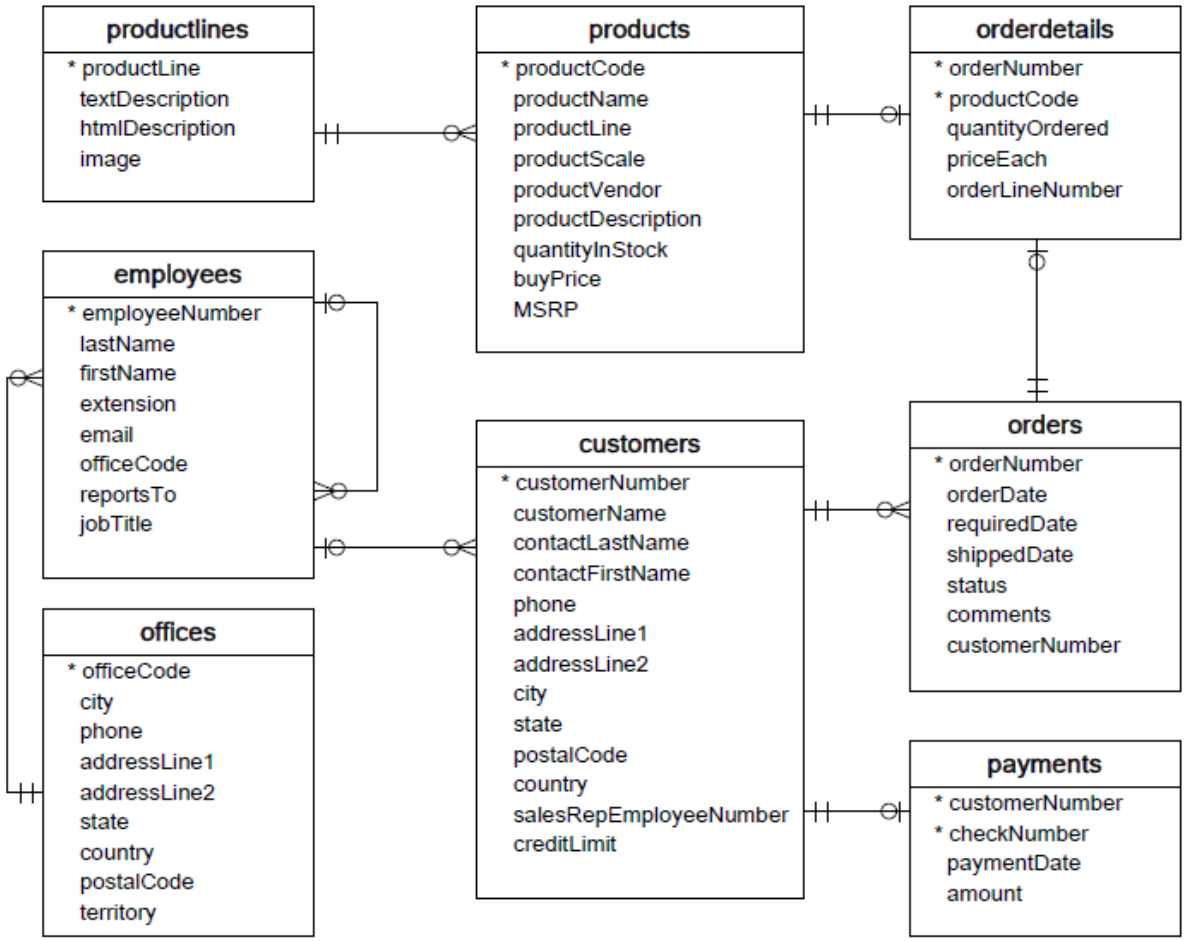
Objectives

You will be able to:

- Write SQL queries that make use of various types of joins
- Compare and contrast the various types of joins
- Discuss how primary and foreign keys are used in SQL
- Decide and perform whichever type of join is best for retrieving desired data

CRM ERD

In this lab, you'll use the same customer relationship management (CRM) database that you saw from the previous lesson.



Connecting to the Database

Import the necessary packages and connect to the database `'data.sqlite'` .

```
import sqlite3
import pandas as pd
```

```
conn = sqlite3.connect('data.sqlite')
```

Select the names of all employees in Boston

Hint: join the employees and offices tables. Select the first and last name.

```
q = """
SELECT firstName, lastName
```

```
FROM employees
JOIN offices
    USING(officeCode)
WHERE city = 'Boston'
;
"""
pd.read_sql(q, conn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	firstName	lastName
0	Julie	Firrelli
1	Steve	Patterson

Are there any offices that have zero employees?

Hint: Combine the employees and offices tables and use a group by. Select the office code, city, and number of employees.

```
# Note that COUNT(*) is not appropriate here because
# we are trying to count the _employees_ in each group.
# So instead we count by some attribute of an employee
# record. The primary key (employeeNumber) is a
# conventional way to do this
q = """
SELECT
    o.officeCode,
    o.city,
    COUNT(e.employeeNumber) AS n_employees
FROM offices AS o
LEFT JOIN employees AS e
    USING(officeCode)
GROUP BY officeCode
HAVING n_employees = 0
```

```
;
"""
pd.read_sql(q, conn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	officeCode	city	n_employees
0	27	Boston	0

Write 3 questions of your own and answer them

```
# Example question:
"""
How many customers are there per office?
"""

# Example question answer:
q = """
SELECT
    o.officeCode,
    o.city,
    COUNT(c.customerNumber) AS n_customers
FROM offices AS o
JOIN employees AS e
    USING(officeCode)
JOIN customers AS c
    ON e.employeeNumber = c.salesRepEmployeeNumber
GROUP BY officeCode
;
"""
pd.read_sql(q, conn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	officeCode	city	n_customers
0	1	San Francisco	12
1	2	Boston	12
2	3	NYC	15
3	4	Paris	29
4	5	Tokyo	5
5	6	Sydney	10
6	7	London	17

Level Up 1: Display the names of every individual product that each employee has sold

Hint: You will need to use multiple `JOIN` clauses to connect all the way from employee names to product names.

```
# We don't need to use aliases for the columns since they
# are conveniently already labeled as different kinds of
# names (firstName, lastName, productName)
q = """
SELECT firstName, lastName, productName
FROM employees AS e
JOIN customers AS c
    ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders
    USING(customerNumber)
JOIN orderdetails
    USING(orderNumber)
JOIN products
    USING(productCode)
```

```
;
"""
df = pd.read_sql(q, conn)
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	firstName	lastName	productName
0	Leslie	Jennings	1958 Setra Bus
1	Leslie	Jennings	1940 Ford Pickup Truck
2	Leslie	Jennings	1939 Cadillac Limousine
3	Leslie	Jennings	1996 Peterbilt 379 Stake Bed with Outrigger
4	Leslie	Jennings	1968 Ford Mustang
...	...	...	...
2991	Martin	Gerard	1954 Greyhound Scenicruiser
2992	Martin	Gerard	1950's Chicago Surface Lines Streetcar
2993	Martin	Gerard	Diamond T620 Semi-Skirted Tanker
2994	Martin	Gerard	1911 Ford Town Car
2995	Martin	Gerard	1936 Mercedes Benz 500k Roadster

2996 rows × 3 columns

Level Up 2: Display the number of products each employee has sold

Alphabetize the results by employee last name.

Hint: Use the `quantityOrdered` column from `orderDetails` . Also, think about how to group the data when some employees might have the same first or last name.

```
q = """
SELECT firstName, lastName, SUM(quantityOrdered) as total_products_sold
FROM employees AS e
JOIN customers AS c
    ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders
    USING(customerNumber)
JOIN orderdetails
    USING(orderNumber)
GROUP BY firstName, lastName
ORDER BY lastName
;
"""
pd.read_sql(q, conn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	firstName	lastName	total_products_sold
0	Loui	Bondur	6186
1	Larry	Bott	8205
2	Pamela	Castillo	9290
3	Julie	Firrelli	4227
4	Andy	Fixter	6246
5	Martin	Gerard	4180
6	Gerard	Hernandez	14231
7	Leslie	Jennings	11854

	firstName	lastName	total_products_sold
8	Barry	Jones	7486
9	Peter	Marsh	6632
10	Mami	Nishi	4923
11	Steve	Patterson	5561
12	Leslie	Thompson	4056
13	Foon Yue	Tseng	5016
14	George	Vanauf	7423

Level Up 3: Display the names employees who have sold more than 200 different products

Hint: this is different from the previous question because the quantity sold doesn't matter, only the number of different products

```
# Recall that HAVING is used for filtering after an aggregation
q = """
SELECT firstName, lastName, COUNT(productCode) as different_products_sold
FROM employees AS e
JOIN customers AS c
    ON e.employeeNumber = c.salesRepEmployeeNumber
JOIN orders
    USING(customerNumber)
JOIN orderdetails
    USING(orderNumber)
GROUP BY firstName, lastName
HAVING different_products_sold > 200
ORDER BY lastName
;
"""
pd.read_sql(q, conn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
```

```
        text-align: right;
    }

</style>
```

	firstName	lastName	different_products_sold
0	Larry	Bott	236
1	Pamela	Castillo	272
2	Gerard	Hernandez	396
3	Leslie	Jennings	331
4	Barry	Jones	220
5	George	Vanauf	211

Summary

Congrats! You practiced using join statements and leveraged your foreign keys knowledge!

Releases

No releases published

Packages

No packages published

Contributors 7

Languages

- **Jupyter Notebook** 100.0%