Join Statements

Introduction

In this section, you will learn about several types of `JOIN` statements. Joins are the primary mechanism for combining data from multiple tables. In order to do this, you define the common attribute(s) between tables in order for them to be combined.

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 almost all industry cases, rather than just working with a single table you will generally need data from multiple tables. Doing this requires the use of **joins** using shared columns from the two tables. For example, here's a diagram of a mock customer relationship management (CRM) database.

Connecting to the Database

As usual, you'll start by connecting to the database.

> In the cell below, type the code to import `sqlite` and `pandas` with the standard alias. Then in the next cell create a connection to the database `data.sqlite` and asign it to a variable:

```
# replace this comment with the code to import the libraries
import sqlite3
import pandas as pd

# replace this comment with the code to create a connection to the
database data.database
conn = sqlite3.connect('data.sqlite')
```

```
import sqlite3
import pandas as pd
conn = sqlite3.connect('data.sqlite')
```

Displaying Product Details Along with Order Details

Let's say you need to generate a report that includes details about products from orders. To do that, we would need to take data from multiple tables in a single statement. To do this we will use `JOIN`.

> In the cell below, type the query to select all records from `orderdetails` and `products` and join them using thier common key `productCode` and display the first 10.

```
# replace None with the query to join orderdetails and proucts on
productCode
query = '''
    SELECT * FROM orderdetails
    JOIN products
    ON orderdetails.productCode = products.productCode
    LIMIT 10;
'''
pd.read_sql(query, conn)
```

```
   orderNumber productCode  quantityOrdered  priceEach
orderLineNumber  \
0        10100    S18_1749               30     136.00
3
1        10100    S18_2248               50      55.09
2
2        10100    S18_4409               22      75.46
4
3        10100    S24_3969               49      35.29
1
4        10101    S18_2325               25     108.06
4
5        10101    S18_2795               26     167.06
1
6        10101    S24_1937               45      32.53
3
7        10101    S24_2022               46      44.35
2
8        10102    S18_1342               39      95.55
2
9        10102    S18_1367               41      43.13
1

  productCode                                productName   productLine
\
0    S18_1749                  1917 Grand Touring Sedan  Vintage Cars

1    S18_2248                       1911 Ford Town Car  Vintage Cars

2    S18_4409        1932 Alfa Romeo 8C2300 Spider Sport  Vintage Cars
```

```
3    S24_3969           1936 Mercedes Benz 500k Roadster  Vintage Cars

4    S18_2325                  1932 Model A Ford J-Coupe  Vintage Cars

5    S18_2795                     1928 Mercedes-Benz SSK  Vintage Cars

6    S24_1937                1939 Chevrolet Deluxe Coupe  Vintage Cars

7    S24_2022  1938 Cadillac V-16 Presidential Limousine  Vintage Cars

8    S18_1342                       1937 Lincoln Berline  Vintage Cars

9    S18_1367   1936 Mercedes-Benz 500K Special Roadster  Vintage Cars


  productScale              productVendor  \
0         1:18  Welly Diecast Productions
1         1:18    Motor City Art Classics
2         1:18              Exoto Designs
3         1:24          Red Start Diecast
4         1:18      Autoart Studio Design
5         1:18       Gearbox Collectibles
6         1:24    Motor City Art Classics
7         1:24    Classic Metal Creations
8         1:18    Motor City Art Classics
9         1:18        Studio M Art Models

                                  productDescription  quantityInStock
\
0  This 1:18 scale replica of the 1917 Grand Tour...               2724

1  Features opening hood, opening doors, opening ...                540

2  This 1:18 scale precision die cast replica fea...               6553

3  This model features grille-mounted chrome horn...               2081

4  This model features grille-mounted chrome horn...               9354

5  This 1:18 replica features grille-mounted chro...                548

6  This 1:24 scale die-cast replica of the 1939 C...               7332

7  This 1:24 scale precision die cast replica of ...               2847

8  Features opening engine cover, doors, trunk, a...               8693

9  This 1:18 scale replica is constructed of heav...               8635
```

```
   buyPrice    MSRP
0     86.70  170.00
1     33.30   60.54
2     43.26   92.03
3     21.75   41.03
4     58.48  127.13
5     72.56  168.75
6     22.57   33.19
7     20.61   44.80
8     60.62  102.74
9     24.26   53.91
```

Expected Output

Compared to the Individual Tables:

`orderdetails` Table:

In the cell below, type the code to select all records from `orderdetails` and display the first `10`

```
# replace None with the query to display the first 10 records in
orderdetails
query = """SELECT * FROM orderdetails LIMIT 10"""
pd.read_sql(query, conn)
```

```
   orderNumber productCode  quantityOrdered  priceEach
orderLineNumber
0        10100    S18_1749               30     136.00
3
1        10100    S18_2248               50      55.09
2
2        10100    S18_4409               22      75.46
4
3        10100    S24_3969               49      35.29
1
4        10101    S18_2325               25     108.06
4
5        10101    S18_2795               26     167.06
1
6        10101    S24_1937               45      32.53
3
7        10101    S24_2022               46      44.35
2
8        10102    S18_1342               39      95.55
2
```

```
9        10102    S18_1367                41      43.13
1
```

Expected Output

`products` Table:

In the cell below, type the code to select all records from `products` and display the first `10`

```
# replace None with the query to display the first 10 records in
products
query = None
pd.read_sql(query, conn)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
c:\Users\rurig\anaconda3\envs\learn-env\lib\site-packages\pandas\io\
sql.py in execute(self, *args, **kwargs)
   1680         try:
-> 1681             cur.execute(*args, **kwargs)
   1682             return cur

ValueError: operation parameter must be str

The above exception was the direct cause of the following exception:

DatabaseError                             Traceback (most recent call
last)
<ipython-input-29-12c934d1a0ed> in <module>
      1 # replace None with the query to display the first 10 records
in products
      2 query = None
----> 3 pd.read_sql(query, conn)

c:\Users\rurig\anaconda3\envs\learn-env\lib\site-packages\pandas\io\
sql.py in read_sql(sql, con, index_col, coerce_float, params,
parse_dates, columns, chunksize)
    481
    482     if isinstance(pandas_sql, SQLiteDatabase):
--> 483         return pandas_sql.read_query(
    484             sql,
    485             index_col=index_col,

c:\Users\rurig\anaconda3\envs\learn-env\lib\site-packages\pandas\io\
```

```
sql.py in read_query(self, sql, index_col, coerce_float, params,
parse_dates, chunksize)
   1725
   1726         args = _convert_params(sql, params)
-> 1727         cursor = self.execute(*args)
   1728         columns = [col_desc[0] for col_desc in
cursor.description]
   1729

c:\Users\rurig\anaconda3\envs\learn-env\lib\site-packages\pandas\io\
sql.py in execute(self, *args, **kwargs)
   1691
   1692             ex = DatabaseError(f"Execution failed on sql
'{args[0]}': {exc}")
-> 1693             raise ex from exc
   1694
   1695     @staticmethod

DatabaseError: Execution failed on sql 'None': operation parameter
must be str
```

Expected Output

The `USING` clause

A more concise way to join the tables, if the column name is identical, is the `USING` clause. Rather then saying `ON tableA.column = tableB.column` we can simply say `USING(column)`. Again, this only works if the column is **identically named** for both tables.

> In the cell below, type the query to select all records in `orderdetails` and `products` and join them on `productCode` with the `USING()` clause, and return the first 10 records:

```
# replace None with the query to join orderdetails and proucts on
productCode with the using() clause
query = """
    SELECT * FROM orderdetails
    JOIN products USING(productCode)
    LIMIT 10;
"""
pd.read_sql(query, conn)

   orderNumber productCode  quantityOrdered  priceEach
orderLineNumber  \
0        10100    S18_1749               30     136.00
3
```

```
1        10100    S18_2248                50      55.09
2
2        10100    S18_4409                22      75.46
4
3        10100    S24_3969                49      35.29
1
4        10101    S18_2325                25     108.06
4
5        10101    S18_2795                26     167.06
1
6        10101    S24_1937                45      32.53
3
7        10101    S24_2022                46      44.35
2
8        10102    S18_1342                39      95.55
2
9        10102    S18_1367                41      43.13
1

                                    productName   productLine
productScale  \
0                   1917 Grand Touring Sedan  Vintage Cars
1:18
1                        1911 Ford Town Car  Vintage Cars
1:18
2        1932 Alfa Romeo 8C2300 Spider Sport  Vintage Cars
1:18
3           1936 Mercedes Benz 500k Roadster  Vintage Cars
1:24
4                  1932 Model A Ford J-Coupe  Vintage Cars
1:18
5                     1928 Mercedes-Benz SSK  Vintage Cars
1:18
6                1939 Chevrolet Deluxe Coupe  Vintage Cars
1:24
7  1938 Cadillac V-16 Presidential Limousine  Vintage Cars
1:24
8                       1937 Lincoln Berline  Vintage Cars
1:18
9   1936 Mercedes-Benz 500K Special Roadster  Vintage Cars
1:18

               productVendor  \
0  Welly Diecast Productions
1    Motor City Art Classics
2              Exoto Designs
3          Red Start Diecast
4      Autoart Studio Design
5       Gearbox Collectibles
```

```
6    Motor City Art Classics
7    Classic Metal Creations
8    Motor City Art Classics
9        Studio M Art Models

                                    productDescription  quantityInStock
\
0  This 1:18 scale replica of the 1917 Grand Tour...              2724

1  Features opening hood, opening doors, opening ...               540

2  This 1:18 scale precision die cast replica fea...              6553

3  This model features grille-mounted chrome horn...              2081

4  This model features grille-mounted chrome horn...              9354

5  This 1:18 replica features grille-mounted chro...               548

6  This 1:24 scale die-cast replica of the 1939 C...              7332

7  This 1:24 scale precision die cast replica of ...              2847

8  Features opening engine cover, doors, trunk, a...              8693

9  This 1:18 scale replica is constructed of heav...              8635


   buyPrice    MSRP
0     86.70  170.00
1     33.30   60.54
2     43.26   92.03
3     21.75   41.03
4     58.48  127.13
5     72.56  168.75
6     22.57   33.19
7     20.61   44.80
8     60.62  102.74
9     24.26   53.91
```

Expected Output

More Aliasing

You can also assign tables an **alias** by entering an alternative shorthand name. This is slightly different than the previous lesson where we introduced aliases for column names, since now we are aliasing *tables*.

When aliasing columns the goal is usually to improve readability by giving something a more specific or easier-to-read name. For example, `name AS employee_name`, `AVG(AVG) AS average_batting_average`, or `COUNT(*) AS num_products`.

When aliasing tables the goal is usually to shorten the name, in order to shorten the overall query. So typically you'll see examples that alias a longer table name to a one-character or two-character shorthand. For example, `orderdetails AS od` or `products AS p`. (It is also possible to use aliases to clarify what exactly is in a table, like how aliases are used for columns, just less common.)

The following query produces the same result as the previous ones, using aliases `od` and `p` for `orderdetails` and `products`, respectively:

In the following cell, type the following code to demonstrate the use of aliasing:

```
# replace None with the query to demonstrate aliasing
query = """
    SELECT * FROM orderdetails AS od
    JOIN products AS p
    ON od.productCode = p.productCode
    LIMIT 10;
"""
pd.read_sql(query, conn)
```

```
   orderNumber productCode  quantityOrdered  priceEach
orderLineNumber  \
0        10100    S18_1749               30     136.00
3
1        10100    S18_2248               50      55.09
2
2        10100    S18_4409               22      75.46
4
3        10100    S24_3969               49      35.29
1
4        10101    S18_2325               25     108.06
4
5        10101    S18_2795               26     167.06
1
6        10101    S24_1937               45      32.53
3
7        10101    S24_2022               46      44.35
2
8        10102    S18_1342               39      95.55
2
9        10102    S18_1367               41      43.13
1

  productCode                                productName   productLine
\
0    S18_1749                   1917 Grand Touring Sedan  Vintage Cars
```

```
1    S18_2248                          1911 Ford Town Car  Vintage Cars
2    S18_4409        1932 Alfa Romeo 8C2300 Spider Sport  Vintage Cars
3    S24_3969           1936 Mercedes Benz 500k Roadster  Vintage Cars
4    S18_2325                  1932 Model A Ford J-Coupe  Vintage Cars
5    S18_2795                     1928 Mercedes-Benz SSK  Vintage Cars
6    S24_1937                1939 Chevrolet Deluxe Coupe  Vintage Cars
7    S24_2022  1938 Cadillac V-16 Presidential Limousine  Vintage Cars
8    S18_1342                       1937 Lincoln Berline  Vintage Cars
9    S18_1367   1936 Mercedes-Benz 500K Special Roadster  Vintage Cars


  productScale               productVendor  \
0         1:18  Welly Diecast Productions
1         1:18    Motor City Art Classics
2         1:18               Exoto Designs
3         1:24           Red Start Diecast
4         1:18      Autoart Studio Design
5         1:18       Gearbox Collectibles
6         1:24    Motor City Art Classics
7         1:24    Classic Metal Creations
8         1:18    Motor City Art Classics
9         1:18        Studio M Art Models

                                  productDescription  quantityInStock
\
0  This 1:18 scale replica of the 1917 Grand Tour...              2724
1  Features opening hood, opening doors, opening ...               540
2  This 1:18 scale precision die cast replica fea...              6553
3  This model features grille-mounted chrome horn...              2081
4  This model features grille-mounted chrome horn...              9354
5  This 1:18 replica features grille-mounted chro...               548
6  This 1:24 scale die-cast replica of the 1939 C...              7332
7  This 1:24 scale precision die cast replica of ...              2847
8  Features opening engine cover, doors, trunk, a...              8693
```

```
9  This 1:18 scale replica is constructed of heav...              8635


   buyPrice    MSRP
0     86.70  170.00
1     33.30   60.54
2     43.26   92.03
3     21.75   41.03
4     58.48  127.13
5     72.56  168.75
6     22.57   33.19
7     20.61   44.80
8     60.62  102.74
9     24.26   53.91
```

Expected Output

Note that just like with column aliases, the `AS` keyword is optional in SQLite. So, instead of `FROM orderdetails AS od` you could write `FROM orderdetails od` with the same outcome.

It is somewhat more common to see `AS` used with column aliases and skipped with table aliases, but again, you'll want to check the syntax rules of your particular type of SQL as well as style guidelines from your employer to know which syntax to use in a professional setting.

`LEFT JOIN`S

By default a `JOIN` is an `INNER JOIN`, or the intersection between two tables. In other words, the `JOIN` between orders and products is only for `productCodes` that are in both the `orderdetails` and `products` tables. If a product had yet to be ordered (and wasn't in the `orderdetails` table) then it would also not be in the result of the `JOIN`.

The `LEFT JOIN` keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side if there is no match.

There are many other types of joins, displayed below. Of these, SQLite does not support outer joins, but it is good to be aware of as more powerful versions of SQL such as PostgreSQL support these additional functions.

For example, the statement

```
SELECT * FROM products LEFT JOIN orderdetails
```

would return all products, even those that hadn't been ordered. You can imagine that all products in inventory should have a description in the product table, but perhaps not every product is represented in the orderdetails table.

In the cell below, type the query to select all records from `products` and join them with all records in `orderdetails` on `productcode` using `LEFT JOIN`, then execute the query and store it in a dataframe named `df`:

```
# replace this comment with the code to create the specified query
q = """SELECT * FROM products LEFT JOIN orderdetails
USING(productCode)"""
# replace this comment with the code to execute the query and store it
in a dataframe named df
df = pd.read_sql(q, conn)
print("Number of records returned:", len(df))
print("Number of records where order details are null:",
len(df[df.orderNumber.isnull()]))
```

```
Number of records returned: 2997
Number of records where order details are null: 1
```

Expected Output

Let's take a look at the one record that has null values in the order details:

```
# run this cell with no changes to view the one record with null
values
df[df.orderNumber.isnull()]
```

```
     productCode        productName   productLine productScale  \
1122    S18_3233  1985 Toyota Supra  Classic Cars         1:18

                  productVendor  \
1122  Highway 66 Mini Classics

                                      productDescription
quantityInStock  \
1122  This model features soft rubber tires, working...
7733

      buyPrice    MSRP  orderNumber  quantityOrdered  priceEach  \
1122     57.01  107.57          NaN              NaN        NaN

      orderLineNumber
1122              NaN
```

Expected Output

As you can see, it's a rare occurrence, but there is one product that has yet to be ordered.

Primary Versus Foreign Keys

Another important consideration when performing joins is to think more about the key or column you are joining on. As you'll see in upcoming lessons, this can lead to interesting behavior if the join value is not unique in one or both of the tables. In all of the above examples, you joined two tables using the **primary key**. The primary key(s) of a table are those column(s) which uniquely identify a row. You'll also see this designated in our schema diagram with the asterisk (*).

You can also join tables using **foreign keys** which are not the primary key for that particular table, but rather another table. For example, `employeeNumber` is the primary key for the employees table and corresponds to the `salesRepEmployeeNumber` of the customers table. In the customers table, `salesRepEmployeeNumber` is only a foreign key, and is unlikely to be a unique identifier, as it is likely that an employee serves multiple customers. As such, in the resulting view `employeeNumber` would no longer be a unique field.

> In the cell below, type the query to join `customers` using the alias `c` with `employees` using the alias `e` on the foreign keys `salesTepEmoloyeeNumber` and `employeeNumber` and order the result by `employeeNumber`, then type the code to execute the query:

```
# replace None with the query to select the desired records
q = """
    SELECT * FROM customers AS c
    JOIN employees AS e
    ON c.salesRepEmployeeNumber = e.employeeNumber
    ORDER BY employeeNumber;
"""
# replace this comment with the code to execute the query
pd.read_sql(q, conn)
```

```
    customerNumber                 customerName contactLastName  \
0              124  Mini Gifts Distributors Ltd.          Nelson
1              129               Mini Wheels Co.          Murphy
2              161         Technics Stores Inc.       Hashimoto
3              321      Corporate Gift Ideas Co.           Brown
4              450     The Sharp Gifts Warehouse           Frick
..             ...                           ...             ...
95             298               Vida Sport, Ltd           Holz
96             344                   CAF Imports       Fernandez
97             376        Precious Collectables            Urs
98             458    Corrida Auto Replicas, Ltd          Sommer
99             484    Iberia Gift Imports, Corp.            Roel

   contactFirstName            phone               addressLine1  \
0             Susan       4155551450            5677 Strong St.
1             Julie       6505555787  5557 North Pendale Street
2              Juri       6505556809          9408 Furth Circle
```

```
3             Julie       6505551386            7734 Strong St.
4               Sue       4085553659             3086 Ingle Ln.
..              ...              ...                       ...
95           Mihael      0897-034555          Grenzacherweg 237
96            Jesus  +34 913 728 555            Merchants House
97            Braun      0452-076555               Hauptstr. 29
98          Martín    (91) 555 22 82             C/ Araquil, 67
99      José Pedro    (95) 555 82 82              C/ Romero, 33

             addressLine2           city state postalCode  ...  \
0                           San Rafael    CA      97562  ...
1                         San Francisco    CA      94217  ...
2                            Burlingame    CA      94217  ...
3                         San Francisco    CA      94217  ...
4                              San Jose    CA      94217  ...
..                      ...          ...   ...        ...  ...
95                                Genève             1203  ...
96  27-30 Merchant's Quay          Madrid            28023  ...
97                                  Bern             3012  ...
98                                Madrid            28023  ...
99                               Sevilla            41101  ...

   salesRepEmployeeNumber  creditLimit  employeeNumber  lastName
firstName  \
0                    1165       210500            1165  Jennings
Leslie
1                    1165        64600            1165  Jennings
Leslie
2                    1165        84600            1165  Jennings
Leslie
3                    1165       105000            1165  Jennings
Leslie
4                    1165        77600            1165  Jennings
Leslie
..                    ...          ...             ...       ...
...
95                   1702       141300            1702    Gerard
Martin
96                   1702        59600            1702    Gerard
Martin
97                   1702            0            1702    Gerard
Martin
98                   1702       104600            1702    Gerard
Martin
99                   1702        65700            1702    Gerard
Martin

   extension                         email officeCode  reportsTo
jobTitle
0      x3291  ljennings@classicmodelcars.com          1       1143
```

```
Sales Rep
1      x3291  ljennings@classicmodelcars.com          1       1143
Sales Rep
2      x3291  ljennings@classicmodelcars.com          1       1143
Sales Rep
3      x3291  ljennings@classicmodelcars.com          1       1143
Sales Rep
4      x3291  ljennings@classicmodelcars.com          1       1143
Sales Rep
..       ...                             ...        ...        ...
...
95     x2312    mgerard@classicmodelcars.com          4       1102
Sales Rep
96     x2312    mgerard@classicmodelcars.com          4       1102
Sales Rep
97     x2312    mgerard@classicmodelcars.com          4       1102
Sales Rep
98     x2312    mgerard@classicmodelcars.com          4       1102
Sales Rep
99     x2312    mgerard@classicmodelcars.com          4       1102
Sales Rep

[100 rows x 21 columns]
```

Expected Output

Notice that this also returned both columns: `salesRepEmployeeNumber` and `employeeNumber`. These columns contain identical values so you would probably actually only want to select one or the other.

<-brian-added/>

Finally, it is important to close the connection.

```
conn.close()
```

Summary

In this lesson, you investigated joins. This included implementing the `ON` and `USING` clauses, aliasing table names, implementing `LEFT JOIN`, and using primary vs. foreign keys.