Ordering and Limiting Data with SQL

Introduction

In this lesson, you'll learn how to sort your query results using an `ORDER BY` clause, and an additional technique for filtering the results of your SQL queries using the `ORDER BY` clause combined with the `LIMIT` clause.

Objectives

You will be able to:

- Order the results of your queries by using `ORDER BY` (`ASC` & `DESC`)
- Limit the number of records returned by a query using `LIMIT`
- Write SQL queries to filter and order results

The Data

Once again we'll be using the Northwind database, containing records related to employees, orders, products, etc.

Below we import the relevant packages and connect to the database:

```
# import pandas module
import pandas as pd
# import sqlite3
import sqlite3
# creating a connection
conn = sqlite3.connect("data.sqlite")
```

In this lesson we'll focus on the `products` table:

```
pd.read_sql("""SELECT * FROM products;""", conn)
```

```
    productCode                               productName
productLine  \
0      S10_1678  1969 Harley Davidson Ultimate Chopper   Motorcycles

1      S10_1949               1952 Alpine Renault 1300  Classic Cars

2      S10_2016                  1996 Moto Guzzi 1100i   Motorcycles

3      S10_4698   2003 Harley-Davidson Eagle Drag Bike   Motorcycles

4      S10_4757                    1972 Alfa Romeo GTA  Classic Cars
```

```
..          ...                                        ...            ...

105   S700_3505                              The Titanic          Ships

106   S700_3962                            The Queen Mary          Ships

107   S700_4002                American Airlines: MD-11S         Planes

108    S72_1253                          Boeing X-32A JSF         Planes

109    S72_3212                               Pont Yacht          Ships


    productScale              productVendor  \
0           1:10             Min Lin Diecast
1           1:10    Classic Metal Creations
2           1:10   Highway 66 Mini Classics
3           1:10           Red Start Diecast
4           1:10    Motor City Art Classics
..           ...                        ...
105        1:700   Carousel DieCast Legends
106        1:700  Welly Diecast Productions
107        1:700        Second Gear Diecast
108         1:72    Motor City Art Classics
109         1:72       Unimax Art Galleries

                                   productDescription quantityInStock
\
0    This replica features working kickstand, front...            7933

1    Turnable front wheels; steering function; deta...            7305

2    Official Moto Guzzi logos and insignias, saddl...            6625

3    Model features, official Harley Davidson logos...            5582

4    Features include: Turnable front wheels; steer...            3252

..                                                 ...             ...

105  Completed model measures 19 1/2 inches long, 9...            1956

106  Exact replica. Wood and Metal. Many extras inc...            5088

107  Polished finish. Exact replia with official lo...            8820

108  10" Wingspan with retractable landing gears.Co...            4857

109  Measures 38 inches Long x 33 3/4 inches High. ...             414
```

```
    buyPrice    MSRP
0      48.81   95.70
1      98.58  214.30
2      68.99  118.94
3      91.02  193.66
4      85.68  136.00
..       ...     ...
105    51.09  100.17
106    53.63   99.31
107    36.27   74.03
108    32.77   49.66
109    33.30   54.60

[110 rows x 9 columns]
```

`ORDER BY`

By default, SQL query results will be returned in the order that they are in the database. This order may or may not have any relevance to your business case.

If, instead, you want to specify the sort order of the returned rows based on the values in the data, you use the `ORDER BY` clause.

A standard query with `ORDER BY` looks like this:

```
SELECT column(s)
  FROM table_name
 ORDER BY column_name (sort_order);
```

Let's say we wanted to order the `products` table by the product name:

```
pd.read_sql("""
    SELECT * FROM products
    ORDER BY productName;
""", conn)
```

```
    productCode                          productName   productLine  \
0      S18_3136  18th Century Vintage Horse Carriage  Vintage Cars
1      S24_2011                18th century schooner         Ships
2      S24_2841             1900s Vintage Bi-Plane         Planes
3      S24_4278           1900s Vintage Tri-Plane          Planes
4      S18_3140                 1903 Ford Model A  Vintage Cars
..          ...                                ...           ...
105   S700_1938                     The Mayflower         Ships
106   S700_3962                   The Queen Mary          Ships
107   S700_1138              The Schooner Bluenose         Ships
108   S700_3505                       The Titanic         Ships
109   S700_2610          The USS Constitution Ship         Ships

    productScale              productVendor  \
```

```
0           1:18          Red Start Diecast
1           1:24   Carousel DieCast Legends
2           1:24      Autoart Studio Design
3           1:24       Unimax Art Galleries
4           1:18       Unimax Art Galleries
..           ...                        ...
105        1:700        Studio M Art Models
106        1:700  Welly Diecast Productions
107        1:700      Autoart Studio Design
108        1:700   Carousel DieCast Legends
109        1:700          Red Start Diecast

                                    productDescription quantityInStock
\
0    Hand crafted diecast-like metal horse carriage...             5992

1    All wood with canvas sails. Many extras includ...             1898

2    Hand crafted diecast-like metal bi-plane is re...             5942

3    Hand crafted diecast-like metal Triplane is Re...             2756

4     Features opening trunk,  working steering system             3913

..                                                  ...              ...

105  Measures 31 1/2 inches Long x 25 1/2 inches Hi...              737

106  Exact replica. Wood and Metal. Many extras inc...             5088

107  All wood with canvas sails. Measures 31 1/2 in...             1897

108  Completed model measures 19 1/2 inches long, 9...             1956

109  All wood with canvas sails. Measures 31 1/2" L...             7083


    buyPrice    MSRP
0      60.74  104.72
1      82.34  122.89
2      34.25   68.51
3      36.23   72.45
4      68.30  136.59
..       ...     ...
105    43.30   86.61
106    53.63   99.31
107    34.00   66.67
108    51.09  100.17
109    33.97   72.28

[110 rows x 9 columns]
```

Ascending and Descending Sort Order

When using `ORDER BY`, the default is to order in *ascending* order. This aligns with "alphabetical order" for text data like `productName`.

You can specify `ASC` (ascending) at the end of the `ORDER BY` clause and retrieve the same result:

```
pd.read_sql("""
    SELECT * FROM products
    ORDER BY productName ASC;
""", conn)
```

```
    productCode                          productName   productLine  \
0      S18_3136  18th Century Vintage Horse Carriage  Vintage Cars
1      S24_2011                18th century schooner         Ships
2      S24_2841              1900s Vintage Bi-Plane        Planes
3      S24_4278             1900s Vintage Tri-Plane        Planes
4      S18_3140                   1903 Ford Model A  Vintage Cars
..          ...                                 ...           ...
105   S700_1938                      The Mayflower         Ships
106   S700_3962                     The Queen Mary         Ships
107   S700_1138                The Schooner Bluenose         Ships
108   S700_3505                        The Titanic         Ships
109   S700_2610           The USS Constitution Ship         Ships

    productScale              productVendor  \
0           1:18          Red Start Diecast
1           1:24   Carousel DieCast Legends
2           1:24      Autoart Studio Design
3           1:24       Unimax Art Galleries
4           1:18       Unimax Art Galleries
..           ...                        ...
105        1:700        Studio M Art Models
106        1:700  Welly Diecast Productions
107        1:700      Autoart Studio Design
108        1:700   Carousel DieCast Legends
109        1:700          Red Start Diecast

                                  productDescription quantityInStock
\
0    Hand crafted diecast-like metal horse carriage...            5992

1    All wood with canvas sails. Many extras includ...            1898

2    Hand crafted diecast-like metal bi-plane is re...            5942

3    Hand crafted diecast-like metal Triplane is Re...            2756

4     Features opening trunk,  working steering system            3913
```

```
..                                                  ...              ...

105  Measures 31 1/2 inches Long x 25 1/2 inches Hi...              737

106  Exact replica. Wood and Metal. Many extras inc...             5088

107  All wood with canvas sails. Measures 31 1/2 in...             1897

108  Completed model measures 19 1/2 inches long, 9...             1956

109  All wood with canvas sails. Measures 31 1/2" L...             7083


    buyPrice    MSRP
0      60.74  104.72
1      82.34  122.89
2      34.25   68.51
3      36.23   72.45
4      68.30  136.59
..       ...     ...
105    43.30   86.61
106    53.63   99.31
107    34.00   66.67
108    51.09  100.17
109    33.97   72.28

[110 rows x 9 columns]
```

Alternatively, if you wanted to use a *descending* order (opposite of alphabetical order for text data), you can use `DESC`:

```
pd.read_sql("""
    SELECT * FROM products
    ORDER BY productName DESC;
""", conn)

    productCode                            productName   productLine  \
0     S700_2610           The USS Constitution Ship         Ships
1     S700_3505                     The Titanic         Ships
2     S700_1138                The Schooner Bluenose         Ships
3     S700_3962                   The Queen Mary         Ships
4     S700_1938                    The Mayflower         Ships
..          ...                                ...           ...
105    S18_3140                  1903 Ford Model A  Vintage Cars
106    S24_4278              1900s Vintage Tri-Plane        Planes
107    S24_2841               1900s Vintage Bi-Plane        Planes
108    S24_2011                18th century schooner         Ships
109    S18_3136  18th Century Vintage Horse Carriage  Vintage Cars

    productScale              productVendor  \
```

```
0          1:700          Red Start Diecast
1          1:700   Carousel DieCast Legends
2          1:700      Autoart Studio Design
3          1:700  Welly Diecast Productions
4          1:700        Studio M Art Models
..           ...                         ...
105         1:18       Unimax Art Galleries
106         1:24       Unimax Art Galleries
107         1:24      Autoart Studio Design
108         1:24   Carousel DieCast Legends
109         1:18          Red Start Diecast

                                    productDescription quantityInStock
\
0    All wood with canvas sails. Measures 31 1/2" L...             7083

1    Completed model measures 19 1/2 inches long, 9...             1956

2    All wood with canvas sails. Measures 31 1/2 in...             1897

3    Exact replica. Wood and Metal. Many extras inc...             5088

4    Measures 31 1/2 inches Long x 25 1/2 inches Hi...              737

..                                                 ...              ...

105   Features opening trunk,  working steering system             3913

106  Hand crafted diecast-like metal Triplane is Re...             2756

107  Hand crafted diecast-like metal bi-plane is re...             5942

108  All wood with canvas sails. Many extras includ...             1898

109  Hand crafted diecast-like metal horse carriage...             5992


    buyPrice    MSRP
0      33.97   72.28
1      51.09  100.17
2      34.00   66.67
3      53.63   99.31
4      43.30   86.61
..       ...     ...
105    68.30  136.59
106    36.23   72.45
107    34.25   68.51
108    82.34  122.89
109    60.74  104.72

[110 rows x 9 columns]
```

Custom Sorting

You are not limited to sorting by alphabetical order. For example, if you wanted to sort based on the length of the `productDescription`, that would look like this:

```
pd.read_sql("""
    SELECT
        productName, length(productDescription) AS description_length
    FROM
        products
    ORDER BY
        description_length;
""", conn)
```

```
                                  productName  description_length
0            1928 British Royal Navy Airplane                  28
1                              P-51-D Mustang                  45
2                           1903 Ford Model A                  48
3                         1904 Buick Runabout                  48
4                 1930 Buick Marquette Phaeton                  48
..                                         ...                 ...
105  1936 Mercedes-Benz 500K Special Roadster                 361
106                           2002 Suzuki XREO                 380
107                       The Schooner Bluenose                390
108                       1996 Moto Guzzi 1100i                391
109      2003 Harley-Davidson Eagle Drag Bike                  494

[110 rows x 2 columns]
```

You can also sort by something without selecting it. Here is a query that has the same order as the query above, but doesn't actually select the description length:

```
pd.read_sql("""
    SELECT productName FROM products
    ORDER BY length(productDescription);
""", conn)
```

```
                                  productName
0            1928 British Royal Navy Airplane
1                              P-51-D Mustang
2                           1903 Ford Model A
3                         1904 Buick Runabout
4                 1930 Buick Marquette Phaeton
..                                         ...
105  1936 Mercedes-Benz 500K Special Roadster
106                           2002 Suzuki XREO
107                       The Schooner Bluenose
108                       1996 Moto Guzzi 1100i
109      2003 Harley-Davidson Eagle Drag Bike
```

```
[110 rows x 1 columns]
```

Sorting By Multiple Columns

You are also not limited to sorting by one column at a time. Typically this is most useful if some rows have repeated values in a given and you want a "tiebreaker" value from another column.

For example, this query sorts by `productVendor`, then `productName`:

```
pd.read_sql("""
    SELECT
          productVendor, productName, MSRP FROM products
    ORDER BY
          productVendor, productName;
""", conn)
```

```
                  productVendor                productName    MSRP
0         Autoart Studio Design     1900s Vintage Bi-Plane   68.51
1         Autoart Studio Design  1932 Model A Ford J-Coupe  127.13
2         Autoart Studio Design  1937 Horch 930V Limousine   65.75
3         Autoart Studio Design   1962 Volkswagen Microbus  127.79
4         Autoart Studio Design           1968 Ford Mustang  194.57
..                          ...                         ...     ...
105  Welly Diecast Productions         1968 Dodge Charger  117.44
106  Welly Diecast Productions         1969 Corvair Monza  151.08
107  Welly Diecast Productions         1969 Dodge Charger  115.16
108  Welly Diecast Productions  1971 Alpine Renault 1600s   61.23
109  Welly Diecast Productions             The Queen Mary   99.31

[110 rows x 3 columns]
```

The ordering of the columns in the `ORDER BY` clause is important. The earlier ones should be the ones requiring a "tiebreaker", not the later ones. If we switch around the order of the previous query, for example, you'll notice that it doesn't really look sorted by `productVendor` at all:

```
pd.read_sql("""
    SELECT
        productVendor, productName, MSRP FROM products
    ORDER BY
        productName, productVendor;
""", conn)
```

```
                  productVendor                              productName
MSRP
0            Red Start Diecast  18th Century Vintage Horse Carriage
104.72
1     Carousel DieCast Legends                 18th century schooner
```

```
122.89
2        Autoart Studio Design                1900s Vintage Bi-Plane
68.51
3         Unimax Art Galleries              1900s Vintage Tri-Plane
72.45
4         Unimax Art Galleries                    1903 Ford Model A
136.59
..                           ...                                 ...
...
105        Studio M Art Models                        The Mayflower
86.61
106  Welly Diecast Productions                       The Queen Mary
99.31
107      Autoart Studio Design                 The Schooner Bluenose
66.67
108   Carousel DieCast Legends                          The Titanic
100.17
109          Red Start Diecast            The USS Constitution Ship
72.28

[110 rows x 3 columns]
```

Another way of thinking about which column should come first after `ORDER BY`, is that the fewer unique values the column has, the earlier it should appear.

To find out how many unique values there are in a column, we can use the `DISTINCT` keyword in addition to `COUNT`. Below are some examples:

```
pd.read_sql("""
    SELECT
          COUNT(DISTINCT productVendor) AS num_product_vendors,
          COUNT(DISTINCT productName) AS num_product_names
    FROM
          products;
""", conn)

   num_product_vendors  num_product_names
0                   13                110
```

This result aligns with what we observed earlier. There are many more unique product names than product vendors, so it makes sense to sort by vendors first, then names.

Type Casting

Sometimes when you use `ORDER BY`, you'll get an odd result like this. We are trying to sort by `quantityInStock` to get a list of products from the least to most number in stock right now:

```
pd.read_sql("""
    SELECT
```

```
        productName, quantityInStock
    FROM
        products
    ORDER BY
        quantityInStock;
""", conn)
```

```
                          productName quantityInStock
0          1970 Chevy Chevelle SS 454            1005
1    Diamond T620 Semi-Skirted Tanker            1016
2                   1969 Ford Falcon            1049
3           1957 Corvette Convertible            1249
4            1928 Ford Phaeton Deluxe             136
..                                ...             ...
105              2002 Chevy Corvette            9446
106    America West Airlines B757-200            9653
107                 1995 Honda Civic            9772
108                   P-51-D Mustang             992
109                 2002 Suzuki XREO            9997

[110 rows x 2 columns]
```

What is 136 doing there? What happened?

Let's take a closer look at the first 10 results (using pandas):

```
pd.read_sql("""
    SELECT productName, quantityInStock FROM products
    ORDER BY quantityInStock;
""", conn).head(10)
```

```
                       productName quantityInStock
0        1970 Chevy Chevelle SS 454            1005
1  Diamond T620 Semi-Skirted Tanker            1016
2                 1969 Ford Falcon            1049
3         1957 Corvette Convertible            1249
4          1928 Ford Phaeton Deluxe             136
5                1952 Citroen-15CV            1452
6          1960 BSA Gold Star DBD34              15
7                   1958 Setra Bus            1579
8    1962 City of Detroit Streetcar            1645
9                 1997 BMW F650 ST             178
```

The problem is that **we assumed that these values were stored as numbers, but apparently they are actually strings**. So they are being sorted in alphabetical order, so "1249", "136", and "1452" are correctly ordered based on comparing the first two characters ("12" comes before "13", which comes before "14").

We want to treat these as numbers, not strings, so we can use `CAST` as part of the `ORDER BY` clause.

Specifically since these are whole numbers, we'll cast them as the `INTEGER` data type in SQLite:

```
pd.read_sql("""
    SELECT productName, quantityInStock FROM products
    ORDER BY CAST(quantityInStock AS INTEGER);
""", conn).head(10)
```

```
                productName quantityInStock
0  1960 BSA Gold Star DBD34              15
1         1968 Ford Mustang              68
2  1928 Ford Phaeton Deluxe             136
3          1997 BMW F650 ST             178
4               Pont Yacht             414
5        1911 Ford Town Car             540
6    1928 Mercedes-Benz SSK             548
7        F/A 18 Hornet 1/72             551
8        2002 Yamaha YZR M1             600
9             The Mayflower             737
```

That looks right for the first 10 rows! Let's check the full dataset:

```
pd.read_sql("""
    SELECT productName, quantityInStock FROM products
    ORDER BY CAST(quantityInStock AS INTEGER);
""", conn)
```

```
                        productName quantityInStock
0          1960 BSA Gold Star DBD34              15
1                 1968 Ford Mustang              68
2          1928 Ford Phaeton Deluxe             136
3                  1997 BMW F650 ST             178
4                       Pont Yacht             414
..                             ...             ...
105       1932 Model A Ford J-Coupe            9354
106             2002 Chevy Corvette            9446
107  America West Airlines B757-200            9653
108                1995 Honda Civic            9772
109                2002 Suzuki XREO            9997

[110 rows x 2 columns]
```

Looks good!

`LIMIT`

`LIMIT` is used to determine the number of records you want to return from a dataset.

A standard query with `LIMIT` would be:

```
SELECT column(s)
  FROM table_name
 LIMIT number;
```

You can use `LIMIT` to specify that you only want a certain number of results, in whatever order they appear in the database. This is kind of like using `.head()` to look at the first part of a dataframe:

```
pd.read_sql("""
    SELECT * FROM orders
    LIMIT 5;
""", conn)
```

```
  orderNumber   orderDate requiredDate shippedDate   status  \
0       10100  2003-01-06   2003-01-13  2003-01-10  Shipped
1       10101  2003-01-09   2003-01-18  2003-01-11  Shipped
2       10102  2003-01-10   2003-01-18  2003-01-14  Shipped
3       10103  2003-01-29   2003-02-07  2003-02-02  Shipped
4       10104  2003-01-31   2003-02-09  2003-02-01  Shipped

                  comments customerNumber
0                                     363
1  Check on availability.             128
2                                     181
3                                     121
4                                     141
```

This approach would also be relevant if you find yourself in a situation where the database is too large, so you can only select a certain number of rows at a time.

`LIMIT` with `ORDER BY`

`LIMIT` becomes much more powerful and useful when combined with `ORDER BY`.

Sometimes you are less interested in results that are filtered based on an *absolute* value like "greater than or equal to 5" or "starting with 'M'", and more interested in results that are filtered based on a *relative* value like "oldest", "youngest", "longest", "shortest", etc.

To do this kind of filtering, you first need to use `ORDER BY` to sort the values, then `LIMIT` to select only the top value(s) based on your sorting.

For example, if we wanted to select the 10 orders with the longest comments to start a customer service audit:

`note` works similarly as `.head(10)`

```
pd.read_sql("""
    SELECT * FROM orders
    ORDER BY length(comments) DESC
```

```
    LIMIT 10;
""", conn) # works-similarly-as-.-head-10
```

```
  orderNumber   orderDate requiredDate shippedDate     status  \
0       10167  2003-10-23   2003-10-30                Cancelled
1       10179  2003-11-11   2003-11-17  2003-11-13  Cancelled
2       10253  2004-06-01   2004-06-09  2004-06-02  Cancelled
3       10173  2003-11-05   2003-11-15  2003-11-09    Shipped
4       10279  2004-08-09   2004-08-19  2004-08-15    Shipped
5       10377  2005-02-09   2005-02-21  2005-02-12    Shipped
6       10124  2003-05-21   2003-05-29  2003-05-25    Shipped
7       10230  2004-03-15   2004-03-24  2004-03-20    Shipped
8       10328  2004-11-12   2004-11-21  2004-11-18    Shipped
9       10367  2005-01-12   2005-01-21  2005-01-16   Resolved

                                             comments customerNumber
0  Customer called to cancel. The warehouse was n...            448
1  Customer cancelled due to urgent budgeting iss...            496
2  Customer disputed the order and we agreed to c...            201
3  Cautious optimism. We have happy customers her...            278
4  Cautious optimism. We have happy customers her...            141
5  Cautious optimism. We have happy customers her...            186
6  Customer very concerned about the exact color ...            112
7  Customer very concerned about the exact color ...            128
8  Customer very concerned about the exact color ...            278
9  This order was disputed and resolved on 2/1/20...            205
```

```
# Showing similarity
pd.read_sql("""
    SELECT * FROM orders
    ORDER BY length(comments) DESC;
""", conn).head(10)
```

```
  orderNumber   orderDate requiredDate shippedDate     status  \
0       10167  2003-10-23   2003-10-30                Cancelled
1       10179  2003-11-11   2003-11-17  2003-11-13  Cancelled
2       10253  2004-06-01   2004-06-09  2004-06-02  Cancelled
3       10173  2003-11-05   2003-11-15  2003-11-09    Shipped
4       10279  2004-08-09   2004-08-19  2004-08-15    Shipped
5       10377  2005-02-09   2005-02-21  2005-02-12    Shipped
6       10124  2003-05-21   2003-05-29  2003-05-25    Shipped
7       10230  2004-03-15   2004-03-24  2004-03-20    Shipped
8       10328  2004-11-12   2004-11-21  2004-11-18    Shipped
9       10367  2005-01-12   2005-01-21  2005-01-16   Resolved

                                             comments customerNumber
0  Customer called to cancel. The warehouse was n...            448
1  Customer cancelled due to urgent budgeting iss...            496
2  Customer disputed the order and we agreed to c...            201
3  Cautious optimism. We have happy customers her...            278
```

```
4  Cautious optimism. We have happy customers her...            141
5  Cautious optimism. We have happy customers her...            186
6  Customer very concerned about the exact color ...            112
7  Customer very concerned about the exact color ...            128
8  Customer very concerned about the exact color ...            278
9  This order was disputed and resolved on 2/1/20...            205
```

You can also combine `LIMIT` with `WHERE` if you want to apply two different kinds of filters (absolute and relative) at the same time:

```
pd.read_sql("""
    SELECT * FROM orders
    WHERE status = "Cancelled"
    ORDER BY length(comments) DESC
    LIMIT 10;
""", conn)
```

```
  orderNumber   orderDate requiredDate shippedDate     status  \
0       10167  2003-10-23   2003-10-30              Cancelled
1       10179  2003-11-11   2003-11-17  2003-11-13  Cancelled
2       10253  2004-06-01   2004-06-09  2004-06-02  Cancelled
3       10260  2004-06-16   2004-06-22              Cancelled
4       10262  2004-06-24   2004-07-01              Cancelled
5       10248  2004-05-07   2004-05-14              Cancelled

                                             comments customerNumber
0  Customer called to cancel. The warehouse was n...            448
1  Customer cancelled due to urgent budgeting iss...            496
2  Customer disputed the order and we agreed to c...            201
3  Customer heard complaints from their customers...            357
4  This customer found a better offer from one of...            141
5  Order was mistakenly placed. The warehouse not...            131
```

As you can see in the above query result, note that **the number after `LIMIT` does not guarantee that you will get that number of results**. That is only the upper limit of the results you might get. It appears that there are only 6 canceled orders, so even though we limited the results to 10, we only got 6.

If there were 0 canceled orders, we would get 0 results, regardless of what `LIMIT` says.

If we loosen the query restriction on the `status` column (producing more results), we can see the `LIMIT` take effect again:

```
pd.read_sql("""
    SELECT * FROM orders
    WHERE status IN ("Cancelled", "Resolved")
    ORDER BY length(comments) DESC
    LIMIT 10;
""", conn)
```

```
  orderNumber   orderDate requiredDate shippedDate     status  \
0       10167  2003-10-23   2003-10-30              Cancelled
1       10179  2003-11-11   2003-11-17  2003-11-13  Cancelled
2       10253  2004-06-01   2004-06-09  2004-06-02  Cancelled
3       10367  2005-01-12   2005-01-21  2005-01-16   Resolved
4       10327  2004-11-10   2004-11-19  2004-11-13   Resolved
5       10164  2003-10-21   2003-10-30  2003-10-23   Resolved
6       10260  2004-06-16   2004-06-22              Cancelled
7       10262  2004-06-24   2004-07-01              Cancelled
8       10386  2005-03-01   2005-03-09  2005-03-06   Resolved
9       10248  2004-05-07   2004-05-14              Cancelled

                                            comments customerNumber
0  Customer called to cancel. The warehouse was n...            448
1  Customer cancelled due to urgent budgeting iss...            496
2  Customer disputed the order and we agreed to c...            201
3  This order was disputed and resolved on 2/1/20...            205
4  Order was disputed and resolved on 12/1/04. Th...            145
5  This order was disputed, but resolved on 11/1/...            452
6  Customer heard complaints from their customers...            357
7  This customer found a better offer from one of...            141
8  Disputed then Resolved on 3/15/2005. Customer ...            141
9  Order was mistakenly placed. The warehouse not...            131
```

`LIMIT` with Date/Time Data

`LIMIT` can be especially useful for finding the oldest and newest records in a dataset with date/time data.

For example, who were the first five unique customers to place an order using this system?

```
pd.read_sql("""
    SELECT DISTINCT customerNumber, orderDate FROM orders
    ORDER BY orderDate
    LIMIT 5;
""", conn)
```

```
  customerNumber   orderDate
0            363  2003-01-06
1            128  2003-01-09
2            181  2003-01-10
3            121  2003-01-29
4            141  2003-01-31
```

Of the orders that have not been shipped and have not been canceled, what are the 10 newest based on order date?

```
pd.read_sql("""
    SELECT * FROM orders
```

```
    WHERE
          shippedDate = "" AND status != "Cancelled"
    ORDER BY
          orderDate DESC
    LIMIT 10;
""", conn)
```

```
  orderNumber   orderDate requiredDate shippedDate      status  \
0       10424  2005-05-31   2005-06-08              In Process
1       10425  2005-05-31   2005-06-07              In Process
2       10422  2005-05-30   2005-06-11              In Process
3       10423  2005-05-30   2005-06-05              In Process
4       10420  2005-05-29   2005-06-07              In Process
5       10421  2005-05-29   2005-06-06              In Process
6       10414  2005-05-06   2005-05-13                 On Hold
7       10407  2005-04-22   2005-05-04                 On Hold
8       10401  2005-04-03   2005-04-14                 On Hold
9       10334  2004-11-19   2004-11-28                 On Hold

                                              comments customerNumber
0                                                                 141
1                                                                 119
2                                                                 157
3                                                                 314
4                                                                 282
5  Custom shipping instructions were sent to ware...              124
6  Customer credit limit exceeded. Will ship when...              362
7  Customer credit limit exceeded. Will ship when...              450
8  Customer credit limit exceeded. Will ship when...              328
9  The outstaniding balance for this customer exc...              144
```

What is the order that took the longest to fulfill, and how long did it take?

```
pd.read_sql("""
    SELECT *,
          julianday(shippedDate) - julianday(orderDate) AS
days_to_fulfill
    FROM
          orders
    WHERE
          shippedDate != ""
    ORDER BY
          days_to_fulfill DESC
    LIMIT 1;
""", conn)
```

```
  orderNumber   orderDate requiredDate shippedDate   status  \
0       10165  2003-10-22   2003-10-31  2003-12-26  Shipped

                                              comments customerNumber  \
```

```
0  This order was on hold because customers's cre...             148

   days_to_fulfill
0             65.0
```

```
# closing the connection
conn.close()
```

Summary

In this lesson, you expanded your SQL knowledge by learning how to sort and limit the results of your query using `ORDER BY` and `LIMIT`. You also saw how to customize the sorting behavior as well as incorporate `DISTINCT` and `WHERE` keywords to create more-sophisticated queries.