The UNION command

You've seen that joins combine two tables together by using a common column as a link. You've used different joins to combine two tables completely, to pull columns from one table to the other table, and to pull only matching records. In general, joins let you arrange columns from two different tables so that they are next to each other.

But what if you want to merge column values from two tables into a single column? For example, say there's one table that contains items 1-5 and their prices. There's a similar column in another table that contains items 6-10 and their prices. Although these tables have similar columns, they have different individual values. You can merge these columns so that you have one table with all 10 items and their prices. To do this in SQL, use a UNION command.

To see this a bit more clearly, take a look at the two tables below. Table 1 has prices for items 1-5. Table 2 has prices for items 6-10.

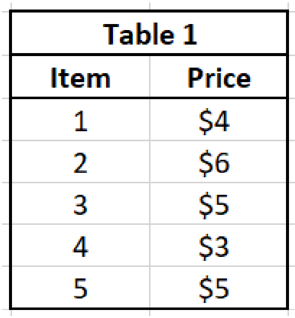


Table 1	
Item	Price
1	$4
2	$6
3	$5
4	$3
5	$5

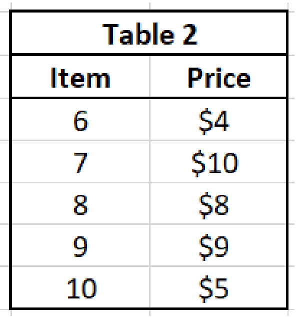
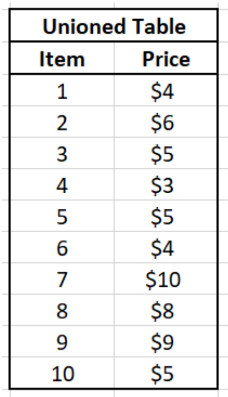


Table 2	
Item	Price
6	$4
7	$10
8	$8
9	$9
10	$5

Say you want to merge these two tables so that you have only one table with items 1-10 and their prices. If you **UNION** these two tables, you'll get the following resulting table:



Unioned Table	
Item	Price
1	$4
2	$6
3	$5
4	$3
5	$5
6	$4
7	$10
8	$8
9	$9
10	$5

So, while a join can be used to add columns from another table to a current table, a union can be used to merge columns from two different tables. In general, UNION is used to stack the results of two SELECT queries.

Union: A clause that combines the results of two SELECT queries

UNION

Although unions can merge columns from many tables, this course focuses on applying a union to two tables. To do this, you can use the UNION command, following the structure below.

```
SELECT column_name(s) FROM table1_name
UNION
SELECT column_name(s) FROM table2_name;
```

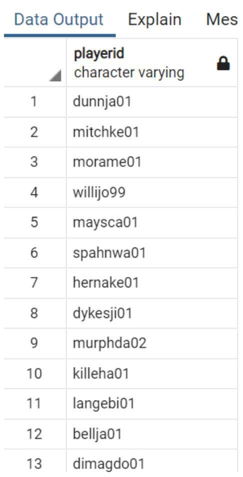
There are a few things to keep in mind when using UNION:

- Every **SELECT** clause in a **UNION** statement must contain the same number of columns. So, if you **SELECT** one column from Table 1, then you must **SELECT** one column from Table 2.
- All columns in a **UNION** statement must have the same data type.
- All columns in a **UNION** statement must have the same table position. So, if the first column in the statement is positioned as the third column in Table 1, then the second column in the statement must be positioned as the third column in Table 2.
- The name of the resulting output column is the same as the name of the Table 1 column.
- The **UNION** command returns only unique records. So, it excludes any duplicates from the output that occur when a value is in both tables.

To try this out, go to pgAdmin and open a new Query Tool for the baseball database. This database contains two tables: hof_inducted and hof_not_inducted. The hof_inducted table has all baseball players who have been inducted into the Hall of Fame. The hof_not_inducted table has all baseball players who haven't been inducted into the Hall of Fame. Say that you want to merge these tables so that you have a list of all players from both tables. Use the syntax structure for UNION to create a query that outputs all playerid values from these two tables. Your query should look like this:

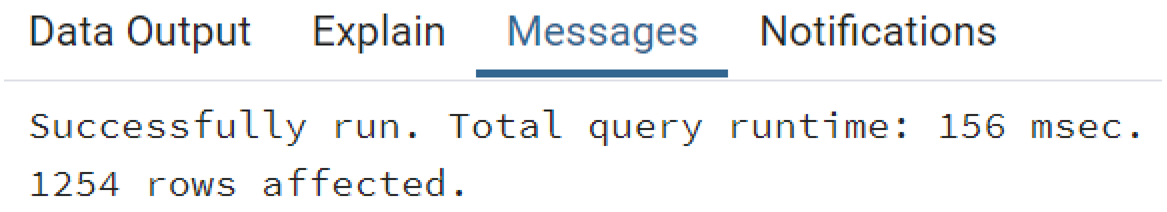
```
SELECT playerid FROM hof_inducted

UNION

SELECT playerid FROM hof_not_inducted;
```

Execute your query. Your output should look like the one below. The order that you see in your output may differ from the one below, because the results are displayed in a random order.



	playerid character varying
1	dunnja01
2	mitchke01
3	morame01
4	willijo99
5	maysca01
6	spahnwa01
7	hernake01
8	dykesji01
9	murphda02
10	killeha01
11	langebi01
12	bellja01
13	dimagdo01

You can also check how many rows are returned in the output. To do that, click the Messages tab. You'll see that this run returned 1,254 unique player IDs.



```
Successfully run. Total query runtime: 156 msec.
1254 rows affected.
```

UNION ALL

The UNION ALL command is just like the UNION command, except that it includes any duplicates in the output. The structure of this command is as follows:

SELECT column_name(s) FROM table1_name

UNION ALL

SELECT column_name(s) FROM table2_name;

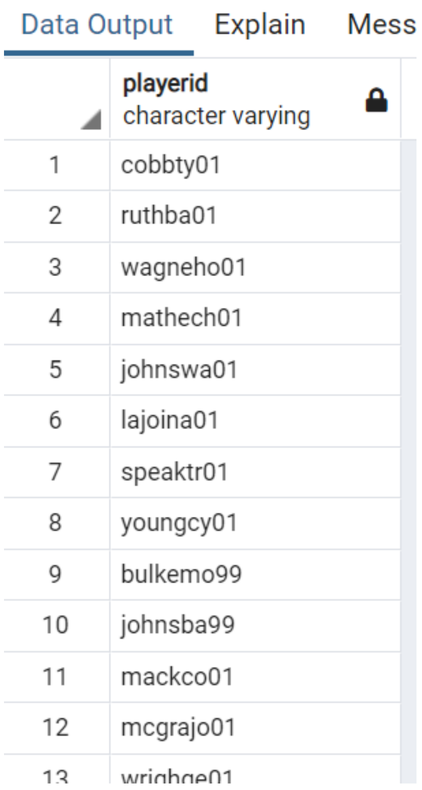
Use the hof_inducted and hof_not_inducted tables again to create a query that merges the playerid columns and keeps duplicate values in the output. Your query should look like this:

SELECT playerid FROM hof_inducted

UNION ALL

SELECT playerid FROM hof_not_inducted;

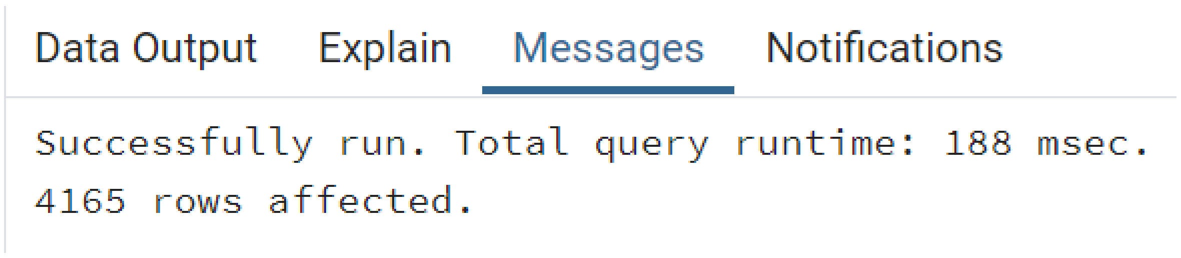
Execute this query to get the following output:



Data Output | Explain | Mess

	playerid character varying
1	cobbty01
2	ruthba01
3	wagneho01
4	mathech01
5	johnswa01
6	lajoina01
7	speaktr01
8	youngcy01
9	bulkemo99
10	johnsba99
11	mackco01
12	mcgrajo01
13	wrighge01

Double-check the Messages tab to see how many records are returned for this query.



```
Successfully run. Total query runtime: 188 msec.
4165 rows affected.
```

This time, there are 4,165 rows returned because this query includes duplicate player IDs. Recall that the UNION query returned only 1,254 records because it included only unique values.