Introduction to SQL

Shaping result sets, and using joins

Jon Flanders
@jonflanders

pluralsight
hardcore developer training

What you will learn...

What you will learn...

- How to shape a single table result set, and then how to join two tables together for a combined result set

Shaping a Result

Shaping a Result

- In the last module, I showed you how to get a result from a single Table using SELECT

Shaping a Result

- In the last module, I showed you how to get a result from a single Table using SELECT
- Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions

Shaping a Result

- **In the last module, I showed you how to get a result from a single Table using SELECT**
- **Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions**
  - Expressions that evaluate to true on a particular row cause that row to be part of the result set

Shaping a Result

- **In the last module, I showed you how to get a result from a single Table using SELECT**
- **Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions**
  - Expressions that evaluate to true on a particular row cause that row to be part of the result set
- **In addition to restricting the result, another common SQL task is to sort the result**

Shaping a Result

- **In the last module, I showed you how to get a result from a single Table using SELECT**
- **Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions**
  - Expressions that evaluate to true on a particular row cause that row to be part of the result set
- **In addition to restricting the result, another common SQL task is to sort the result**
- **You also might want to transform the data so it can be more easily consumed**

Shaping a Result

- **In the last module, I showed you how to get a result from a single Table using SELECT**
- **Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions**
  - Expressions that evaluate to true on a particular row cause that row to be part of the result set
- **In addition to restricting the result, another common SQL task is to sort the result**
- **You also might want to transform the data so it can be more easily consumed**
  - Consumed by the end-user or an application

Shaping a Result

- **In the last module, I showed you how to get a result from a single Table using SELECT**
- **Using the WHERE clause, I showed you how to restrict the results from that Table by using expressions**
  - Expressions that evaluate to true on a particular row cause that row to be part of the result set
- **In addition to restricting the result, another common SQL task is to sort the result**
- **You also might want to transform the data so it can be more easily consumed**
  - Consumed by the end-user or an application
- **SQL provides native capabilities that should be used before resorting to other means**

ORDER BY

ORDER BY

- The ORDER BY clause allows you to sort the result set

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**
  - Multiple columns are separated by commas

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**
  - Multiple columns are separated by commas
- **Ascending (ASC) order is the default**

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**
  - Multiple columns are separated by commas
- **Ascending (ASC) order is the default**
  - You can add the keyword DESC to cause the ordering to be descending

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**
  - Multiple columns are separated by commas
- **Ascending (ASC) order is the default**
  - You can add the keyword DESC to cause the ordering to be descending

```
Who are all the people in my
contact list, ordered by last name?
```

ORDER BY

- **The ORDER BY clause allows you to sort the result set**
  - ORDER BY comes after the WHERE clause, but the WHERE clause isn't required
- **You specify one or more of the columns from the result set**
  - Multiple columns are separated by commas
- **Ascending (ASC) order is the default**
  - You can add the keyword DESC to cause the ordering to be descending

Who are all the people in my contact list, ordered by last name?

```
SELECT p.person_first_name,p.persons_last_name
        FROM person p
        ORDER BY p.person_last_name
```

Set Functions

Set Functions

- SQL includes a number of built-in functions to provide additional functionality

Set Functions

- SQL includes a number of built-in functions to provide additional functionality
- Some of those functions help to turn column data from Tables into computed values

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**
  - This rule has an exception – stay tuned

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**
  - This rule has an exception – stay tuned
- **These functions help add additional interesting questions we can ask of the data**

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**
  - This rule has an exception – stay tuned
- **These functions help add additional interesting questions we can ask of the data**

```
How many contacts do I have?
```

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**
  - This rule has an exception – stay tuned
- **These functions help add additional interesting questions we can ask of the data**

```
How many contacts do I have?
```

```
Who is the contact that I've
interacted with the least?
```

Set Functions

- **SQL includes a number of built-in functions to provide additional functionality**
- **Some of those functions help to turn column data from Tables into computed values**
  - You pass the column name as the parameter to the function
- **You use these functions instead of columns in the select list**
  - This rule has an exception – stay tuned
- **These functions help add additional interesting questions we can ask of the data**

```
How many contacts do I have?
```

```
Who is the contact that I've
interacted with the least?
```

```
What is the average number of times
I've contacted people in my contact
list?
```

Computational Functions

Computational Functions

Operator	Description

Computational Functions

Operator	Description
COUNT	Returns the count of the column specified – note that COUNT **includes** NULL values when used with *

Computational Functions

Operator	Description
COUNT	Returns the count of the column specified – note that COUNT **includes** NULL values when used with *
MAX	Returns the maximum value of the column specified – does not include NULL values

Computational Functions

Operator	Description
COUNT	Returns the count of the column specified – note that COUNT **includes** NULL values when used with *
MAX	Returns the maximum value of the column specified – does not include NULL values
MIN	Returns the minimum value of the column specified – does not include NULL values

Computational Functions

Operator	Description
COUNT	Returns the count of the column specified – note that COUNT **includes** NULL values when used with *
MAX	Returns the maximum value of the column specified – does not include NULL values
MIN	Returns the minimum value of the column specified – does not include NULL values
AVG	Returns the average value for the column specified – does not include NULL values and only works on numeric columns

Computational Functions

Operator	Description
COUNT	Returns the count of the column specified – note that COUNT **includes** NULL values when used with *
MAX	Returns the maximum value of the column specified – does not include NULL values
MIN	Returns the minimum value of the column specified – does not include NULL values
AVG	Returns the average value for the column specified – does not include NULL values and only works on numeric columns
SUM	Returns the sum of the values for the column specified – does not include NULL values and only works on numeric columns

Result Set Qualifiers - Review

Result Set Qualifiers - Review

- By default, a query will return ALL rows from the Table listed after the FROM clause

Result Set Qualifiers - Review

- By default, a query will return ALL rows from the Table listed after the FROM clause
- There are two ways to limit the number of rows in a simple SELECT Statement

Result Set Qualifiers - Review

- By default, a query will return ALL rows from the Table listed after the FROM clause
- There are two ways to limit the number of rows in a simple SELECT Statement
- One is to add additional clauses after the FROM clause

Result Set Qualifiers - Review

- By default, a query will return ALL rows from the Table listed after the FROM clause
- There are two ways to limit the number of rows in a simple SELECT Statement
- One is to add additional clauses after the FROM clause
- Another is to use the DISTINCT qualifier before the select list

Result Set Qualifiers - Review

- By default, a query will return ALL rows from the Table listed after the FROM clause
- There are two ways to limit the number of rows in a simple SELECT Statement
- One is to add additional clauses after the FROM clause
- Another is to use the DISTINCT qualifier before the select list
  - This limits the result set to all the distinct values rather than all the values

Result Set Qualifiers - Review

- **By default, a query will return ALL rows from the Table listed after the FROM clause**
- **There are two ways to limit the number of rows in a simple SELECT Statement**
- **One is to add additional clauses after the FROM clause**
- **Another is to use the DISTINCT qualifier before the select list**
  - This limits the result set to all the distinct values rather than all the values
  - ALL is also a result set qualifier – but it is always implied

Result Set Qualifiers - Review

- **By default, a query will return ALL rows from the Table listed after the FROM clause**
- **There are two ways to limit the number of rows in a simple SELECT Statement**
- **One is to add additional clauses after the FROM clause**
- **Another is to use the DISTINCT qualifier before the select list**
  - This limits the result set to all the distinct values rather than all the values
  - ALL is also a result set qualifier – but it is always implied

```
What are all the unique first
names among my contacts?
```

Result Set Qualifiers - Review

- **By default, a query will return ALL rows from the Table listed after the FROM clause**
- **There are two ways to limit the number of rows in a simple SELECT Statement**
- **One is to add additional clauses after the FROM clause**
- **Another is to use the DISTINCT qualifier before the select list**
  - This limits the result set to all the distinct values rather than all the values
  - ALL is also a result set qualifier – but it is always implied

What are all the unique first names among my contacts?

```
SELECT DISTINCT p.person_first_name
  FROM person p;
```

Qualifiers and Set Functions

Qualifiers and Set Functions

- You can add the DISTINCT or ALL qualifier to a column passed to a Set Function

Qualifiers and Set Functions

- **You can add the DISTINCT or ALL qualifier to a column passed to a Set Function**
  - ALL is the implicit default here, as it is when it is at the beginning of the select list

Qualifiers and Set Functions

- **You can add the DISTINCT or ALL qualifier to a column passed to a Set Function**
  - ALL is the implicit default here, as it is when it is at the beginning of the select list
- **The Set Function is run against the DISTINCT result set of that column rather than all values**

Qualifiers and Set Functions

- **You can add the DISTINCT or ALL qualifier to a column passed to a Set Function**
  - ALL is the implicit default here, as it is when it is at the beginning of the select list
- **The Set Function is run against the DISTINCT result set of that column rather than all values**

What is the count of unique first names among my contacts?

Qualifiers and Set Functions

- **You can add the DISTINCT or ALL qualifier to a column passed to a Set Function**
  - ALL is the implicit default here, as it is when it is at the beginning of the select list
- **The Set Function is run against the DISTINCT result set of that column rather than all values**

What is the count of unique first names among my contacts?

```
SELECT Count(DISTINCT p.person_first_name)
  FROM person p;
```

GROUP BY

GROUP BY

- Normally, a SET Function has to be the only item in the select list

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

- **GROUP BY lets you put all the subsets together, and link those subsets up to the column(s) specified in the GROUP BY clause**

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

- **GROUP BY lets you put all the subsets together, and link those subsets up to the column(s) specified in the GROUP BY clause**
  - So to appear in the GROUP BY clause, the column must be in the select list

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

- **GROUP BY lets you put all the subsets together, and link those subsets up to the column(s) specified in the GROUP BY clause**
  - So to appear in the GROUP BY clause, the column must be in the select list

```
What is the count of every
unique first names among my
contacts?
```

GROUP BY

- **Normally, a SET Function has to be the only item in the select list**
  - Including other columns wouldn't make sense anyway
- **If you could run the function against a subset of the set, you could get a "sub" result**

- **GROUP BY lets you put all the subsets together, and link those subsets up to the column(s) specified in the GROUP BY clause**
  - So to appear in the GROUP BY clause, the column must be in the select list

```
What is the count of every
unique first names among my
contacts?
```

```
SELECT Count(DISTINCT p.person_first_name),
       p.person_first_name
  FROM person p
GROUP BY p.person_first_name;
```

HAVING

HAVING

- The HAVING clause works against the result set returned with a GROUP BY clause in the same way that the WHERE clause works against the result set returned from a simple SELECT

HAVING

- **The HAVING clause works against the result set returned with a GROUP BY clause in the same way that the WHERE clause works against the result set returned from a simple SELECT**

What is the count of unique first name among my contacts that appears at least 5 times?

HAVING

- The HAVING clause works against the result set returned with a GROUP BY clause in the same way that the WHERE clause works against the result set returned from a simple SELECT

What is the count of unique first name among my contacts that appears at least 5 times?

```
SELECT
Count(DISTINCT p.person_first_name) as
NameCount,
p.person_first_name
FROM person p
GROUP BY p.person_first_name
HAVING NameCount >= 5;
```

JOINS

JOINS

- A join in SQL is creating a new result set from two or more tables

JOINS

- A join in SQL is creating a new result set from two or more tables
- Joins are made by extending the FROM clause to include all of the tables you wish to query

JOINS

- A join in SQL is creating a new result set from two or more tables
- Joins are made by extending the FROM clause to include all of the tables you wish to query
- The vast majority of the time, you will also extend the WHERE clause to include an equality expression that includes columns from each of the joined tables

JOINS

- **A join in SQL is creating a new result set from two or more tables**
- **Joins are made by extending the FROM clause to include all of the tables you wish to query**
- **The vast majority of the time, you will also extend the WHERE clause to include an equality expression that includes columns from each of the joined tables**
  - Most often these columns will be primary or foreign keys

JOINS

- **A join in SQL is creating a new result set from two or more tables**
- **Joins are made by extending the FROM clause to include all of the tables you wish to query**
- **The vast majority of the time, you will also extend the WHERE clause to include an equality expression that includes columns from each of the joined tables**
  - Most often these columns will be primary or foreign keys
  - Using the relations between the Tables to answer more complex questions

CROSS

CROSS

- The simplest type of JOIN is perhaps the least useful

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**
  - All the rows and columns from both tables

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**
  - All the rows and columns from both tables
- **It isn't very useful because there isn't any connection between the rows**

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**
  - All the rows and columns from both tables
- **It isn't very useful because there isn't any connection between the rows**
- **CROSS keyword is optional**

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**
  - All the rows and columns from both tables
- **It isn't very useful because there isn't any connection between the rows**
- **CROSS keyword is optional**

What are all the first names
and email addresses I have?

CROSS

- **The simplest type of JOIN is perhaps the least useful**
  - Also incredibly inefficient
- **The CROSS JOIN is what Maths calls a Cartesian Product**
  - All the rows and columns from both tables
- **It isn't very useful because there isn't any connection between the rows**
- **CROSS keyword is optional**

What are all the first names
and email addresses I have?

```
SELECT
p.person_first_name, e.email_address
  FROM person p, email_address e;
```

INNER JOIN

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**

INNER JOIN

- INNER JOIN is the expected “typical” join in a relational system
- Take all the rows from Table A

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows
- **INNER JOIN goes between table names in the FROM clause**

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows
- **INNER JOIN goes between table names in the FROM clause**
- **The ON clause follows the INNER JOIN clause**

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows
- **INNER JOIN goes between table names in the FROM clause**
- **The ON clause follows the INNER JOIN clause**
  - Boolean expression to match the key columns

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows
- **INNER JOIN goes between table names in the FROM clause**
- **The ON clause follows the INNER JOIN clause**
  - Boolean expression to match the key columns

```
What are my contacts' email
addresses?
```

INNER JOIN

- **INNER JOIN is the expected "typical" join in a relational system**
- **Take all the rows from Table A**
  - Find all the rows in Table B where a key in Table A is equal to a key in Table B
  - Make a new result set with all those rows
- **INNER JOIN goes between table names in the FROM clause**
- **The ON clause follows the INNER JOIN clause**
  - Boolean expression to match the key columns

What are my contacts' email addresses?

```
SELECT
p.person_first_name, p.person_last_name,
e.email_address
FROM person p INNER JOIN email_address e
ON p.person_id = e.email_address_person_id;
```

OUTER JOIN

OUTER JOIN

- The difference between the INNER JOIN and the OUTER JOIN relates to NULL

OUTER JOIN

- The difference between the INNER JOIN and the OUTER JOIN relates to NULL
- The INNER JOIN only joins against rows where there is a match in the joined table

OUTER JOIN

- The difference between the INNER JOIN and the OUTER JOIN relates to NULL
- The INNER JOIN only joins against rows where there is a match in the joined table
- OUTER JOIN works even when a row in one of the tables doesn't match with a row in the joined table

OUTER JOIN

- **The difference between the INNER JOIN and the OUTER JOIN relates to NULL**
- **The INNER JOIN only joins against rows where there is a match in the joined table**
- **OUTER JOIN works even when a row in one of the tables doesn't match with a row in the joined table**
  - i.e., even when there is no row that matches the WHERE clause expression

OUTER JOIN

- **The difference between the INNER JOIN and the OUTER JOIN relates to NULL**
- **The INNER JOIN only joins against rows where there is a match in the joined table**
- **OUTER JOIN works even when a row in one of the tables doesn't match with a row in the joined table**
  - i.e., even when there is no row that matches the WHERE clause expression
- **A FULL OUTER JOIN will return a result set with all the joined rows**

OUTER JOIN

- **The difference between the INNER JOIN and the OUTER JOIN relates to NULL**
- **The INNER JOIN only joins against rows where there is a match in the joined table**
- **OUTER JOIN works even when a row in one of the tables doesn't match with a row in the joined table**
  - i.e., even when there is no row that matches the WHERE clause expression
- **A FULL OUTER JOIN will return a result set with all the joined rows**
  - The rows from the first table will also be returned, matched with NULL values for the second table's columns

OUTER JOIN

- **The difference between the INNER JOIN and the OUTER JOIN relates to NULL**
- **The INNER JOIN only joins against rows where there is a match in the joined table**
- **OUTER JOIN works even when a row in one of the tables doesn't match with a row in the joined table**
  - i.e., even when there is no row that matches the WHERE clause expression
- **A FULL OUTER JOIN will return a result set with all the joined rows**
  - The rows from the first table will also be returned, matched with NULL values for the second table's columns
  - If there are any rows in the second table that match the expression, they are also returned – and the cells from the first table will be NULL

OUTER JOIN

OUTER JOIN

What are all my contacts and their email addresses, including the ones missing an email address and the ones with an email address but missing a contact name.?

OUTER JOIN

What are all my contacts and their email addresses, including the ones missing an email address and the ones with an email address but missing a contact name.?

```
SELECT
p.person_first_name, p.person_last_name,
e.email_address
FROM person p FULL OUTER JOIN email_address e
ON p.person_id = e.email_address_person_id;
```

OUTER JOIN

What are all my contacts and their email addresses, including the ones missing an email address and the ones with an email address but missing a contact name.?

```
SELECT
p.person_first_name, p.person_last_name,
e.email_address
FROM person p FULL OUTER JOIN email_address e
ON p.person_id = e.email_address_person_id;
```

person_first_name	person_last_name	email_address
Jon	Flanders	jon@...
Fritz	Onion	fritz@...
Shannon	Ahern	NULL
NULL	NULL	aaron@...

LEFT OUTER JOIN

LEFT OUTER JOIN

- Using LEFT OUTER JOIN means that only the rows from the table on the left of the LEFT OUTER JOIN clause will be returned

LEFT OUTER JOIN

- **Using LEFT OUTER JOIN means that only the rows from the table on the left of the LEFT OUTER JOIN clause will be returned**
  - Rows that are not matched will have NULL for the columns from the right-hand side table

LEFT OUTER JOIN

LEFT OUTER JOIN

What are my contacts and their email addresses, including those I don't have an email for?

LEFT OUTER JOIN

What are my contacts and their email addresses, including those I don't have an email for?

```
SELECT
  p.person_first_name, p.person_last_name,
  e.email_address
FROM person p LEFT OUTER JOIN email_address e
ON p.person_id = e.email_address_person_id;
```

LEFT OUTER JOIN

What are my contacts and their email addresses, including those I don't have an email for?

```
SELECT
  p.person_first_name, p.person_last_name,
  e.email_address
FROM person p LEFT OUTER JOIN email_address e
ON p.person_id = e.email_address_person_id;
```

person_first_name	person_last_name	email_address
Jon	Flanders	jon@...
Fritz	Onion	fritz@...
Shannon	Ahern	NULL

RIGHT OUTER JOIN

RIGHT OUTER JOIN

- RIGHT OUTER JOIN returns all the rows from the table on the right-hand side of the JOIN clause

RIGHT OUTER JOIN

- RIGHT OUTER JOIN returns all the rows from the table on the right-hand side of the JOIN clause
- NULL values for rows that don't have a match in the left-hand side table

RIGHT OUTER JOIN

RIGHT OUTER JOIN

What are the email addresses I have, including those emails I don't have a person for?

RIGHT OUTER JOIN

What are the email addresses I have, including those emails I don't have a person for?

```
SELECT
  p.person_first_name, p.person_last_name,
    e.email_address
FROM person p RIGHT OUTER JOIN email_address e
  ON p.person_id = e.email_address_person_id;
```

RIGHT OUTER JOIN

What are the email addresses I have, including those emails I don't have a person for?

```
SELECT
  p.person_first_name, p.person_last_name,
    e.email_address
FROM person p RIGHT OUTER JOIN email_address e
  ON p.person_id = e.email_address_person_id;
```

person_first_name	person_last_name	email_address
Jon	Flanders	jon@...
Fritz	Onion	fritz@...
NULL	NULL	aaron@...

SELF JOIN

SELF JOIN

- It is odd but sometimes it is useful to join a table against itself

SELF JOIN

- It is odd but sometimes it is useful to join a table against itself
- There isn't any specific syntax for this JOIN, but it is worth mentioning that the same table can be on both the left-hand side and the right-hand side of a JOIN clause

SUMMARY

SUMMARY

- **ORDER BY and GROUP BY can help you to shape your results to more easily answer complex questions from your data.**

SUMMARY

- ORDER BY and GROUP BY can help you to shape your results to more easily answer complex questions from your data.
- JOINS make the relational model come to life by associating tables together