Table of Contents

[1. Associating tables on conditions other than equality 1](#_Toc330937911)

[2. Self-Joins 3](#_Toc330937912)

[3. Legacy comma style inner join 5](#_Toc330937913)

Many of our joins are joining tables by matching the fk and pk of two tables and doing an equality match. There are a few more join conditions that may be useful.

1. Associating tables on conditions other than equality

We have a table to supply a customer's credit rating based on their credit limit. This holds descriptive terms for various credit levels assigned to a customer. But there is no relationship defined between these two tables since the credit levels do not generally exactly match the values in the credit ratings table. This type of join is often used for a lookup up table that is based on a range of values.

1. Displaying oe\_credit-ratings. These are integer values and the ranges do not overlap.

Select \*

From a\_oe.credit\_ratings

;

+-----------+------------+-----------+

| low\_limit | high\_limit | rating |

+-----------+------------+-----------+

| 0 | 1000 | Standard |

| 1001 | 2000 | Good |

| 2001 | 5000 | High |

| 5001 | 10000 | Excellent |

| 10001 | 99999 | Superior |

+-----------+------------+-----------+

1. Displaying the rating for customers. Notice that the joining clause uses a Between operator instead of equality.

Select cust\_id, credit\_limit, rating

From a\_oe.customers

Join a\_oe.credit\_ratings on credit\_limit between low\_limit and high\_limit

Order by cust\_id

;

+---------+--------------+-----------+

| cust\_id | credit\_limit | rating |

+---------+--------------+-----------+

| 400300 | 6000 | Excellent |

| 400801 | 750 | Standard |

| 401250 | 750 | Standard |

| 401890 | 1750 | Good |

| 402100 | 750 | Standard |

| 402110 | 750 | Standard |

| 402120 | 750 | Standard |

| 403000 | 6000 | Excellent |

| 403010 | 6000 | Excellent |

| 403050 | 6000 | Excellent |

| 403100 | 6000 | Excellent |

| 403500 | 6000 | Excellent |

| 403750 | 6000 | Excellent |

| 403760 | 6000 | Excellent |

| 404000 | 3500 | High |

| 404100 | 3500 | High |

| 404150 | 3500 | High |

| 404180 | 3500 | High |

| 404890 | 1750 | Good |

+---------+--------------+-----------+

. . . rows omitted

1. We can also write this query using the following syntax which filters the Cartesian product.

Select cust\_id, credit\_limit, rating

From a\_oe.customers

, a\_oe.credit\_ratings

Where credit\_limit between low\_limit and high\_limit

Order by cust\_id

;

Pay close attention to this query. This is using a Cartesian product- we have two tables in the From clause separated by a comma. The Where clause is supplying the test to associate these two tables and I get 31 rows returned in my data set. If I omit the Where clause then I am doing a Cartesian product with no filter for associating rows and I get back 165 rows in the result set ( 5 rows in the credit\_ratings table times 33 rows in the customer tables). And 134 of these rows are meaningless.

Select cust\_id, credit\_limit, rating

From a\_oe.customers, a\_oe.credit\_ratings

Order by cust\_id;

Suppose I use following filter to find customers with an Excellent credit rating and I get a result set that shows all of my customers!

Select cust\_id, credit\_limit, rating

From a\_oe.customers, a\_oe.credit\_ratings

Where rating in ('Excellent')

Order by cust\_id;

Cartesian products are not always a mistake but they need to be examined closely.

1. This uses a join that involves two attributes to check if any items were sold at more than their list price.

Select a\_prd.products.prod\_id, quoted\_price, prod\_list\_price, ord\_id

From a\_oe.order\_details od

Join a\_prd.products

on od.prod\_id = a\_prd.products.prod\_id

and quoted\_price > prod\_list\_price

;

+---------+--------------+-----------------+--------+

| prod\_id | quoted\_price | prod\_list\_price | ord\_id |

+---------+--------------+-----------------+--------+

| 1010 | 175.00 | 150.00 | 120 |

| 1010 | 175.00 | 150.00 | 121 |

| 1010 | 175.00 | 150.00 | 390 |

| 1010 | 195.00 | 150.00 | 395 |

| 1100 | 205.00 | 49.99 | 301 |

+---------+--------------+-----------------+--------+

1. Self-Joins

You can join a table to itself. You need to use a table alias to distinguish the two copies of the table involved in the join. The following is the traditional self-join of employees and their managers

1. Employees and managers . Note the back ticks on the Order by keys

Select concat(m.emp\_id, ' ' , m.name\_last) as "Manager"

, concat(e.emp\_id, ' ' , e.name\_last) as "Supervises"

From a\_emp.employees e

Left join a\_emp.employees m on m.emp\_id = e.emp\_mng

Order by `Manager`, `Supervises`

;

+------------+------------+

| Manager | Supervises |

+------------+------------+

| NULL | 100 King |

| 100 King | 101 Koch |

| 100 King | 102 D'Haa |

| 100 King | 145 Russ |

| 100 King | 146 Partne |

| 100 King | 201 Harts |

| 101 Koch | 108 Green |

| 101 Koch | 162 Holme |

| 101 Koch | 200 Whale |

| 101 Koch | 203 Mays |

| 101 Koch | 205 Higgs |

| 102 D'Haa | 103 Hunol |

| 103 Hunol | 104 Ernst |

| 108 Green | 109 Fiet |

| 108 Green | 110 Chen |

| 145 Russ | 150 Tuck |

| 145 Russ | 155 Hiller |

| 145 Russ | 207 Russ |

| 146 Partne | 160 Dorna |

| 146 Partne | 161 Dewal |

| 205 Higgs | 204 King |

| 205 Higgs | 206 Geitz |

+------------+------------+

22 rows in set (0.00 sec)

This is another self-join. The following query returns pairs of employees who have the same job id. We are joining on the job id and also on an inequality between the employees' ids. If we do not add that second joining condition, then each employee would be paired with themselves (since the job id values would match). The output shows one row if there are two employees with the same job id; and three rows if there are three employees with the same job id due to the pair matching .

1. Pairing Employees who have the same job id

Select emp\_1.job\_id

, emp\_1.emp\_id as Emp1, emp\_2.emp\_id as Emp2

From a\_emp.employees emp\_1

Join a\_emp.employees emp\_2

on emp\_1.job\_id = emp\_2.job\_id

and emp\_1.emp\_id < emp\_2.emp\_id

Order by emp\_1.job\_id, emp\_1.emp\_id, emp\_2.emp\_id

;

+--------+------+------+

| job\_id | Emp1 | Emp2 |

+--------+------+------+

| 8 | 150 | 155 |

| 8 | 150 | 207 |

| 8 | 155 | 207 |

| 16 | 101 | 108 |

| 16 | 101 | 161 |

| 16 | 101 | 162 |

| 16 | 101 | 200 |

| 16 | 101 | 203 |

| 16 | 101 | 205 |

| 16 | 108 | 161 |

| 16 | 108 | 162 |

| 16 | 108 | 200 |

| 16 | 108 | 203 |

| 16 | 108 | 205 |

| 16 | 161 | 162 |

| 16 | 161 | 200 |

| 16 | 161 | 203 |

| 16 | 161 | 205 |

| 16 | 162 | 200 |

| 16 | 162 | 203 |

| 16 | 162 | 205 |

| 16 | 200 | 203 |

| 16 | 200 | 205 |

| 16 | 203 | 205 |

| 32 | 104 | 109 |

| 32 | 104 | 110 |

| 32 | 104 | 160 |

| 32 | 104 | 204 |

| 32 | 104 | 206 |

| 32 | 109 | 110 |

| 32 | 109 | 160 |

| 32 | 109 | 204 |

| 32 | 109 | 206 |

| 32 | 110 | 160 |

| 32 | 110 | 204 |

| 32 | 110 | 206 |

| 32 | 160 | 204 |

| 32 | 160 | 206 |

| 32 | 204 | 206 |

| 64 | 102 | 103 |

| 64 | 102 | 146 |

| 64 | 103 | 146 |

+--------+------+------+

42 rows in set (0.00 sec)

1. Finding employees who earn more than other employees. This has a lot of rows of output

Select

e1.emp\_id, e1.salary ,' earns more than '

, e2.emp\_id ,e2.salary

From a\_emp.employees e1 ,

a\_emp.employees e2

Where e1.salary > e2.salary

Order by e1.salary desc, e1.emp\_id

;

The output starts with employee 101 who has the highest salary and is matched with all other employees. The next set of rows starts with employee 162 who has the next highest salary. The last set of rows starts with employee 103 who earns more than only the employee(s) with the lowest salary- in our data set that is employee 150 . Note there is no set of rows that start with this employee id.

+--------+----------+-------------------+--------+----------+

| emp\_id | salary | earns more than | emp\_id | salary |

+--------+----------+-------------------+--------+----------+

| 101 | 98005.00 | earns more than | 207 | 65000.00 |

| 101 | 98005.00 | earns more than | 145 | 65000.00 |

| 101 | 98005.00 | earns more than | 203 | 44450.00 |

| 101 | 98005.00 | earns more than | 201 | 15000.00 |

| 101 | 98005.00 | earns more than | 104 | 50000.00 |

| 101 | 98005.00 | earns more than | 155 | 80000.00 |

| 101 | 98005.00 | earns more than | 110 | 30300.00 |

| 101 | 98005.00 | earns more than | 109 | 15000.00 |

| 101 | 98005.00 | earns more than | 146 | 88954.00 |

| 101 | 98005.00 | earns more than | 162 | 98000.00 |

| 101 | 98005.00 | earns more than | 160 | 15000.00 |

| 101 | 98005.00 | earns more than | 161 | 15000.00 |

| 101 | 98005.00 | earns more than | 103 | 9000.00 |

. . . rows omitted

| 205 | 15000.00 | earns more than | 150 | 6500.00 |

| 205 | 15000.00 | earns more than | 108 | 12000.00 |

| 205 | 15000.00 | earns more than | 103 | 9000.00 |

| 108 | 12000.00 | earns more than | 150 | 6500.00 |

| 108 | 12000.00 | earns more than | 103 | 9000.00 |

| 103 | 9000.00 | earns more than | 150 | 6500.00 |

+--------+----------+-------------------+--------+----------+

211 rows in set (0.00 sec)

1. Legacy comma style inner join

There is a traditional, legacy join that does the attribute matching in the Where clause. You will see this join in a lot of older code ( and a lot of code written now).

Logically this syntax does a Cartesian product and adds a filter for the records that match on the joining condition.

1. This is the join using the column name syntax

Select cust\_id

, oh.ord\_id

, prod\_id

, quantity\_ordered \* quoted\_price as "extprice"

From a\_oe.order\_headers oh

Join a\_oe.order\_details od on oh.ord\_id = od.ord\_id

Order by cust\_id, oh.ord\_id

;

+---------+--------+---------+----------+

| cust\_id | ord\_id | prod\_id | extprice |

+---------+--------+---------+----------+

| 400300 | 378 | 1120 | 2250.00 |

| 400300 | 378 | 1125 | 2250.00 |

| 401250 | 106 | 1060 | 255.95 |

| 401250 | 113 | 1080 | 22.50 |

| 401250 | 119 | 1070 | 225.00 |

| 401250 | 301 | 1100 | 205.00 |

| 401890 | 112 | 1110 | 99.98 |

| 401890 | 519 | 1020 | 64.75 |

| 401890 | 519 | 1110 | 49.99 |

. . . rows omitted

1. Using the join of orders and order details in the Where clause

Select oh.cust\_id

, oh.ord\_id

, od.prod\_id

, od.quantity\_ordered \* od.quoted\_price as "extprice"

From a\_oe.order\_headers oh

, a\_oe.order\_details od

Where oh.ord\_id = od.ord\_id

Order by oh.cust\_id, oh.ord\_id

;

The advantage of doing the join in the From clause is that it isolates the join issues from the Where clause filters. If you do the join in the Where clause then you need to take more care with other filters in the Where clause especially if you have both And and Or operators in the Where clause.

This join syntax is not allowed in this class for assignments. I want you to get used to using the more uniform join syntax using the Condition join or Column Name join..