Table of Contents

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

Demo 01: This uses a join that involves two attributes to check if any items were sold at more than their list price. This does not use a Where clause- the testing is done in the join.

```
select
  PR.prod_id
, quoted_price
, prod_list_price
, order_id
from oe_orderDetails OD
join prd_products PR On OD.prod_id = PR.prod_id
and quoted_price > prod_list_price;
```

PROD_ID	QUOTED_PRICE	PROD_LIST_PRICE	ORDER_ID
1010	175	150	390
1010	195	150	395
1010	175	150	550
1010	175	150	551
1010	175	150	609
1010	175	150	2120
1010	175	150	2121
1100	205	49.99	301
1150	7.25	4.99	223
1152	55.25	55	540
1152	55.25	55	2508

2. 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

Demo 02: Employees and managers . Note that the first row here has no Manager. Employee 100 is at the top of the chart.

```
select
  M.emp_id || ' ' || M.name_last As "Manager"
, E.emp_id || ' ' || E.name_last As "Supervises"
from emp_employees E
Left Join emp_employees M On m.emp_id = e.emp_mng
order by "Manager", "Supervises";
```

Manager	Supervises
	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
. . . rows omitted
```

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.

Demo 03: Pairing Employees who have the same job id

```
select emp_1.job_id
, emp_1.emp_id, emp_2.emp_id
from emp_employees emp_1
join 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     EMP_ID     EMP_ID
---------- ---------- ----------
         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 selected
```

Demo 04: 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 ' as " "
,  E2.emp_id ,E2.salary
from emp_employees E1 ,
     emp_employees E2
where E1.salary > E2.salary
order by  E1.salary desc, E2.salary desc;
```

The output starts with employee 161 who has the highest salary and is matched with all other employees. The next set of rows starts with employee 100 who has the next highest salary. The last set of rows starts with employee 150 who earns more than only the employee(s) with the lowest salary- in our data set that is employee 201 . Note there is no set of rows that start with this employee id.

```
    EMP_ID     SALARY                      EMP_ID     SALARY
---------- ---------- ----------------- ---------- ----------
       161     120000  earns more than         100     100000
       161     120000  earns more than         204      99090
       161     120000  earns more than         101      98005
       161     120000  earns more than         162      98000
       161     120000  earns more than         206      88954
       161     120000  earns more than         146      88954
       161     120000  earns more than         205      75000
       161     120000  earns more than         103      69000
       161     120000  earns more than         160      65000
       161     120000  earns more than         104      65000
       161     120000  earns more than         109      65000
       161     120000  earns more than         200      65000
       161     120000  earns more than         203      64450
       161     120000  earns more than         108      62000
       161     120000  earns more than         102      60300
       161     120000  earns more than         110      60300
       161     120000  earns more than         145      59000
       161     120000  earns more than         207      30000
       161     120000  earns more than         155      29000
       161     120000  earns more than         150      20000
       161     120000  earns more than         201      15000
       100     100000  earns more than         204      99090
       100     100000  earns more than         101      98005
       100     100000  earns more than         162      98000
       100     100000  earns more than         206      88954
       100     100000  earns more than         146      88954
       100     100000  earns more than         205      75000
. . .  rows omitted for many employees
       207      30000  earns more than         155      29000
       207      30000  earns more than         150      20000
       207      30000  earns more than         201      15000
       155      29000  earns more than         150      20000
       155      29000  earns more than         201      15000
       150      20000  earns more than         201      15000

223 rows selected.
```

3. Legacy comma 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 in 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.

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.

Demo 05: A Column Name style inner join of orders and order details

```
select
  customer_id
, order_id
, prod_id
, quantity_ordered * quoted_price As "ExtCost"
from oe_orderHeaders
join oe_orderDetails Using (order_id)
order by customer_id, order_id;
```

```
CUSTOMER_ID   ORDER_ID    PROD_ID    ExtCost
----------- ---------- ---------- ----------
     400300        378       1120       2250
     400300        378       1125       2250
     401250        106       1060     255.95
     401250        113       1080       22.5
     401250        119       1070        225
     401250        301       1100        205
. . .  rows omitted
```

Demo 06: Comma join: Using the join of orders and order details in the Where clause

```
select
  oe_orderHeaders.customer_id
, oe_orderHeaders.order_id
, oe_orderDetails.prod_id
, oe_orderDetails.quantity_ordered * oe_orderDetails.quoted_price As "ExtCost"
from oe_orderHeaders
   , oe_orderDetails
where oe_orderHeaders.order_id = oe_orderDetails.order_id
order by oe_orderHeaders.customer_id, oe_orderHeaders.order_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.

Oracle also supports an older outer join syntax in the Where clause which we do not discuss.