Table of Contents

Inner joins are great for finding customers with Orders and for finding Products that have been ordered. But we often want to find customers who have no orders or products that no one has ordered. These are sometimes called unmatched queries since we are looks for customers in the customer table who have no matching rows in the order headers table. We will see several ways to do this. For now we will look at two approaches: (1) using the outer join and (2) using subqueries.

1. Unmatched queries using outer join

In the previous document we used the outer join to find employees with and without an assigned department. A variation on the outer join is a query to display only those employees who have no assigned department. Be careful to select the proper column for testing against null. With these tests, you do not want to use the join with the Using (col) syntax because you have to specify the exact column you are looking for. Compare the following two queries. We want departments with no employees.

Demo 01: Unmatched rows. Departments which do not have any employees

```
Select z_em_emp.d_id as "em_emp.d_id"
, z_em_dept.d_id as "em_dept.d_id"
, d_name
From z_em_dept
LEFT JOIN z_em_emp on z_em_dept.d_id = z_em_emp.d_id
Where z_em_emp.d_id IS NULL
;
+-------------+--------------+-----------+
| em_emp.d_id | em_dept.d_id | d_name    |
+-------------+--------------+-----------+
|        NULL |          200 | Marketing |
+-------------+--------------+-----------+
```

Demo 02: Unmatched rows. Take care which attribute you test. Since we are retrieving all data from the department table we will not have nulls in the department table id attribute.

```
Select z_em_emp.d_id  as "em_emp.d_id"
, z_em_dept.d_id as "em_dept.d_id"
, d_name
From z_em_dept
LEFT JOIN z_em_emp on z_em_dept.d_id = z_em_emp.d_id
Where z_em_dept.d_id IS NULL
;
Empty set (0.00 sec)
```

Still looking for departments which do not have any employees. Compare the following two demos.

Demo 03: Using a column name join. In the standard you cannot qualify the column used in this join and we get back no departments without employees. Note that this does not present as an error; we simply get no rows returned.

```
Select d_id
, d_name
From z_em_dept
LEFT JOIN z_em_emp using(d_id)
Where d_id IS NULL;
Empty set (0.00 sec)
```

Demo 04: MySQL allows qualification of the joining column D_ID to specify the table name. Since this is a non-standard extension you might with to stay with the condition join

```
Select z_em_emp.d_id as "em_emp.d_id"
, z_em_dept.d_id as "em_dept.d_id"
, d_name
From z_em_dept
LEFT JOIN z_em_emp using(d_id)
Where z_em_emp.d_id IS NULL;
+-------------+--------------+-----------+
| em_emp.d_id | em_dept.d_id | d_name    |
+-------------+--------------+-----------+
|        NULL |          200 | Marketing |
+-------------+--------------+-----------+
```

Demo 05: If you have troubles with setting up this type of query, run the query without the null filter first and examine the columns for the rows you want to return. Here we can see that we want to test the z_em_emp.d_id column for nulls.

```
Select z_em_emp.d_id as "em_emp.d_id"
, z_em_dept.d_id as "em_dept.d_id"
, d_name
From z_em_dept
Left join z_em_emp on z_em_dept.d_id = z_em_emp.d_id;
+-------------+--------------+---------------+
| em_emp.d_id | em_dept.d_id | d_name        |
+-------------+--------------+---------------+
|         100 |          100 | Manufacturing |
|         150 |          150 | Accounting    |
|         150 |          150 | Accounting    |
|        NULL |          200 | Marketing     |
|         250 |          250 | Research      |
+-------------+--------------+---------------+
```

2. Queries using the Altgeld mart tables

Demo 06: Customers without orders. We can use the Using (col) syntax here since the filter column is not the join column.

```
Select cust_id
, cust_name_last
, ord_id
From a_oe.customers
Left join a_oe.order_headers using(cust_id)
Where cust_id between 404900 and 409030
And ord_id IS NULL
Order by cust_id;
```

```
+---------+---------------+--------+
| cust_id | cust_name_last | ord_id |
+---------+---------------+--------+
|  409010 | Morris         |   NULL |
|  409020 | Max            |   NULL |
+---------+---------------+--------+
2 rows in set (0.00 sec)
```

Demo 07: If we try to find orders with no customers, we have no rows returned. Our database is set up to reject any order that is not associated with a customer. This would be a good query to run on poorly designed databases to locate orphaned rows.

```
Select CS.cust_id
, CS.cust_name_last
, OH.ord_id
From a_oe.order_headers OH
Left join a_oe.customers CS on CS.cust_id = OH.cust_id
Where OH.cust_id is null
;
```

```
Empty set (0.00 sec)
```

Demo 08: What is the product name and list price for the products that are not selling? These would be products in the products table that do not appear on any order.

```
Select catg_id  as catg
, prod_id  as p_id
, prod_desc as product
, prod_list_price as price
From a_prd.products
Left join  a_oe.order_details   using (prod_id)
Where ord_id is null
Order by catg_id, prod_id
;
+------+------+-----------------------------------------------+--------+
| catg | p_id | product                                       | price  |
+------+------+-----------------------------------------------+--------+
| APL  | 1126 | Low Energy washer Dryer combo                 | 850.00 |
| APL  | 4569 | Sized for the apartment                       | 349.95 |
| GFD  | 5000 | Cello bag of mixed fingerling potatoes        |  12.50 |
| GFD  | 5001 | Dundee Ginger Preserve 12 oz jar              |   5.00 |
| HW   | 1160 | Stand Mixer with attachments                  | 149.99 |
| HW   | 4575 | GE model 34PG98                               |  49.95 |
| MUS  | 2234 | Charles Mingus - Pithecanthropus Erectus      |  15.88 |
| MUS  | 2337 | John Coltrane - Blue Train                    |  15.87 |
| MUS  | 2487 | Stanley Turrentine - Don't Mess With Mr. T    |   9.45 |
| MUS  | 2933 | David Newman - I Remember Brother Ray         |  12.45 |
| MUS  | 2987 | Stanley Turrentine - Ballads                  |  15.87 |
| PET  | 1142 | Bird seed mix with sunflowers                 |   2.50 |
| PET  | 1143 | Bird seed mix with more sunflower seeds       |   2.50 |
| PET  | 4567 | Our highest end cat tree- you gotta see this  | 549.99 |
| PET  | 4568 | Satin four-poster cat bed                     | 549.99 |
+------+------+-----------------------------------------------+--------+
15 rows in set (0.00 sec)
```

Demo 09: Do we have any products with no inventory? This is an example of a question that needs clarification. For this query we will consider this to be either no inventory row at all or an inventory level of zero.

```
Select catg_id
, a_prd.products.prod_id
, prod_name as product
, quantity_on_hand
From a_prd.products
Left join a_prd.inventory on a_prd.products.prod_id = a_prd.inventory.prod_id
Where a_prd.inventory.prod_id is null
Or quantity_on_hand = 0
Order by quantity_on_hand , catg_id, prod_id
;
+---------+---------+--------------------+------------------+
| catg_id | prod_id | product            | quantity_on_hand |
+---------+---------+--------------------+------------------+
| APL     |    1126 | WasherDryer        |             NULL |
| APL     |    4569 | Mini Dryer         |             NULL |
| GFD     |    5000 | Fingerling Potatoes |             NULL |
| GFD     |    5001 | Ginger Preserve    |             NULL |
. . . rows omitted
| PET     |    4567 | Deluxe Cat Tree    |             NULL |
| PET     |    4568 | Deluxe Cat Bed     |             NULL |
| PET     |    4576 | Cosmo cat nip      |             NULL |
| PET     |    4577 | Cat leash          |             NULL |
| SPG     |    1050 | Stationary bike    |                0 |
+---------+---------+--------------------+------------------+
26 rows in set (0.00 sec)
```

3. Unmatched queries using subqueries

Some people find this syntax easier to understand. We are looking for data where we have a value in one table and we do not have that value in another table.

Demo 10: This is the Customers without orders query done using a subquery. (This is not including the cust_id range filter we had before) This filters for customer id values that are not in the order headers table- that would be customers with no orders.

```
Select cust_id
, cust_name_last
From a_oe.customers
Where cust_id NOT IN  (
    Select  cust_id
    From a_oe.order_headers
    )
;
+---------+----------------+
| cust_id | cust_name_last |
+---------+----------------+
|  400801 | Washington     |
|  402110 | Coltrane       |
|  402120 | McCoy          |
|  402500 | Jones          |
|  403500 | Stevenson      |
|  403750 | O'Leary        |
|  403760 | O'Leary        |
```

```
|  404150 | Dancer         |
|  404180 | Shay           |
|  404890 | Kelley         |
|  409010 | Morris         |
|  409020 | Max            |
+---------+----------------+
12 rows in set (0.00 sec)
```

Demo 11: What is the product name and list price for the products that are not selling? These would be products in the products table that do not appear on any order.
This query does not need table aliases since each part of the query is referencing a single table

```
Select catg_id  as catg
, prod_id  as p_id
, prod_desc as product
, prod_list_price as price
From a_prd.products
Where prod_id NOT IN  (
     Select  prod_id
     From a_oe.order_details
     )
Order by catg_id, prod_id
;
+------+------+-----------------------------------------------+--------+
| catg | p_id | product                                       | price  |
+------+------+-----------------------------------------------+--------+
| APL  | 1126 | Low Energy washer Dryer combo                 | 850.00 |
| APL  | 4569 | Sized for the apartment                       | 349.95 |
| GFD  | 5000 | Cello bag of mixed fingerling potatoes        |  12.50 |
| GFD  | 5001 | Dundee Ginger Preserve 12 oz jar              |   5.00 |
| HW   | 1160 | Stand Mixer with attachments                  | 149.99 |
| HW   | 4575 | GE model 34PG98                               |  49.95 |
| MUS  | 2234 | Charles Mingus - Pithecanthropus Erectus      |  15.88 |
| MUS  | 2337 | John Coltrane - Blue Train                    |  15.87 |
| MUS  | 2487 | Stanley Turrentine - Don't Mess With Mr. T    |   9.45 |
| MUS  | 2933 | David Newman - I Remember Brother Ray         |  12.45 |
| MUS  | 2987 | Stanley Turrentine - Ballads                  |  15.87 |
| PET  | 1142 | Bird seed mix with sunflowers                 |   2.50 |
| PET  | 1143 | Bird seed mix with more sunflower seeds       |   2.50 |
| PET  | 4567 | Our highest end cat tree- you gotta see this  | 549.99 |
| PET  | 4568 | Satin four-poster cat bed                     | 549.99 |
+------+------+-----------------------------------------------+--------+
15 rows in set (0.00 sec)
```

4. What can go wrong?

Suppose we want to find employees who are not associated with any orders. First do a left join to see what the data looks like.

Demo 12: Left join Employees to Orders

```
Select emp_id
, name_last
, ord_id
From emp_employees
Left join oe_order_headers on emp_id = sales_rep_id
;
```

```
+--------+-----------+--------+
| emp_id | name_last | Ord_id |
+--------+-----------+--------+
|    100 | King      |   NULL |
|    101 | Koch      |   NULL |
|    102 | D'Haa     |   NULL |
|    103 | Hunol     |   NULL |
|    104 | Ernst     |   NULL |
|    108 | Green     |   NULL |
|    109 | Fiet      |   NULL |
|    110 | Chen      |   NULL |
|    145 | Russ      |    112 |
|    145 | Russ      |    130 |
|    145 | Russ      |    312 |
|    145 | Russ      |    405 |
|    145 | Russ      |    505 |
|    145 | Russ      |    540 |
|    146 | Partne    |   NULL |
|    150 | Tuck      |    105 |
|    150 | Tuck      |    106 |
|    150 | Tuck      |    107 |
|    150 | Tuck      |    111 |

81 rows in set
```

We could add a filter to find the rows where the Ord_id is null.

Demo 13: Left join Employees to Orders with null order id. These are employees who are not associated with any order.

```
Select emp_id
, name_last
From a_emp.employees
Left join a_oe.order_headers on emp_id = sales_rep_id
Where ord_id is null;
+--------+-----------+
| emp_id | name_last |
+--------+-----------+
|    100 | King      |
|    101 | Koch      |
|    102 | D'Haa     |
|    103 | Hunol     |
|    104 | Ernst     |
|    108 | Green     |
|    109 | Fiet      |
|    110 | Chen      |
|    146 | Partne    |
|    160 | Dorna     |
|    161 | Dewal     |
|    162 | Holme     |
|    200 | Whale     |
|    201 | Harts     |
|    203 | Mays      |
|    204 | King      |
|    205 | Higgs     |
|    206 | Geitz     |
|    207 | Russ      |
+--------+-----------+
```

```
19 rows in set (0.00 sec)
```

What if we try this with a subquery?

Demo 14: Subquery version 1 We are filter for employee id that are not in the appropriate column in the order headers table. This returns no rows at all! Before you read on to the next demo try to figure out why this might happen. (What is the usual villain when a query goes bad?)

```
Select emp_id
, name_last
From a_emp.employees
Where emp_id NOT IN  (
     Select  sales_rep_id
     From a_oe.order_headers );
Empty set (0.00 sec)
```

Remember that a Not In () predicate returns no rows if there is a null in the list.

Demo 15: Subquery version 2-

```
Select emp_id
, name_last
From a_emp.employees
Where emp_id NOT IN  (
     Select  sales_rep_id
     From a_oe.order_headers
     Where sales_rep_id is not null)
;
Same result set as with the outer join
```

So now the question is: why did the other subqueries work? We were filtering on an attribute that was a not null attribute.