Table of Contents

1. Unmatched queries using outer join 1
2. Queries using the Altgeld mart tables 2
3. Unmatched queries using subqueries 3
4. What can go wrong? 4

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 looking 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. If we do not qualify the column used in this join, 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 have an outer join from customers left join order headers to get customers with and without orders and then we filter for just rows where the order id in the order headers table is null.

```
select CS.customer_id
, CS.customer_name_last
, OH.order_id
from customer.customers  CS
left join orderEntry.orderHeaders OH on CS.customer_id = OH.customer_id
where CS.customer_id between 404900 and 409030
and OH.order_id IS NULL
order by CS.customer_id;
+-------------+--------------------+----------+
| customer_id | customer_name_last | order_id |
+-------------+--------------------+----------+
|      408777 | Morise             |     NULL |
|      409010 | Morris             |     NULL |
|      409020 | Max                |     NULL |
+-------------+--------------------+----------+
```

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.customer_id
, cs.customer_name_last
, oh.order_id
from orderEntry.orderHeaders oh
Left Join customer.customers cs on cs.customer_id = oh.customer_id
where oh.customer_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 product.products
Left Join orderEntry.orderDetails   using (prod_id)
where order_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)
```

3. Unmatched queries using subqueries

The unmatched pattern we have used gets customer with and without orders and then filters away the customers with orders.

Some people find the subquery syntax easier to understand. The subquery approach says to find rows where we have a value in one table and we do not have that value in another table. For example, customers in the customer table but that customer is not in the order-headers table.

Demo 09: These are customers without orders query done using a subquery.
The subquery gets the customer_id for customers with orders and then the main query filters those out using the NOT IN test. That gives us the customers with no orders.

```
select customer_id, customer_name_last
from customer.customers
where customer_id between 404900 and 409030
and  customer_id NOT IN
  (
   select customer_id
   from orderEntry.orderHeaders
   );
```

```
+-------------+--------------------+
| customer_id | customer_name_last |
+-------------+--------------------+
|      408777 | Morris             |
|      409010 | Morris             |
|      409020 | Max                |
+-------------+--------------------+
```

Notice that we are comparing the customer_id in the customers table with a list of customer_id values from the orderHeaders table. We compare customer_id to customer_id. This is similar to the join logic.

Demo 10: 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 for prod_id 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 product.products
where prod_id NOT IN  (
   select prod_id
   from orderEntry.orderDetails
   )
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. Remember that in the orderHeaders table, the employee who took the order is referred to as the sales_rep. First do a left join to see what the data looks like.

Demo 11: Left join Employees to Orders

```
select emp_id, name_last, order_id
from employee.employees
Left join orderEntry.orderHeaders on emp_id = sales_rep_id;
+--------+-----------+----------+
| emp_id | name_last | 0rder_id |
+--------+-----------+----------+
|    100 | King      |     NULL |
|    101 | Koch      |     NULL |
|    102 | D'Haa     |     NULL |
```

```
|    103 | Hunol     |     NULL |
. . . rows omitted
|    145 | Russ      |      312 |
|    145 | Russ      |      405 |
|    145 | Russ      |      505 |
|    145 | Russ      |      540 |
|    146 | Partne    |     NULL |
. . . rows omitted
```

Some employees have served as a sales rep and some have taken more than one order. We could add a filter to find the rows where the Order_id is null.

Demo 12: 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 employee.employees
Left join orderEntry.orderHeaders on emp_id = sales_rep_id
where order_id is null;
+--------+-----------+
| emp_id | name_last |
+--------+-----------+
|    100 | King      |
|    101 | Koch      |
|    102 | D'Haa     |
|    103 | Hunol     |
. . . rows omitted
|    146 | Partne    |
. . . rows omitted
```

What if we try this with a subquery?

Demo 13: Subquery version 1 We filter for employee id values that are not in the appropriate column (sales_rep_id) 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 employee.employees
where emp_id NOT IN  (
  select sales_rep_id
  from orderEntry.orderHeaders  );
Empty set (0.00 sec)
```

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

Demo 14: Subquery version 2- Eliminate the nulls from the subquery.

```
select emp_id, name_last
from employee.employees
where emp_id NOT IN  (
  select sales_rep_id
  from orderEntry.orderHeaders
  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.