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 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
----------    -----------    --------------
                      200    MARKETING
```

Demo 02:　　　Unmatched rows. This syntax will not work since the USING clause means that we cannot qualify D_ID to specify the table name. Note that this does not present as an error; we simply get no rows returned.

```
select D_ID as "EM_Emp.D_ID"
, D_ID as "EM_Dept.D_ID"
, D_Name
from z_em_Dept
LEFT JOIN z_em_Emp Using(D_ID)
where D_ID IS NULL;
```
```
no rows selected
```

Demo 03:　　　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
;
```
```
no rows selected
```

Demo 04: 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
----------- ------------ --------------
        150          150 Accounting
        150          150 Accounting
        250          250 Research
        100          100 Manufacturing
                     200 Marketing
```

2. Queries using the AltgeldMart tables

Demo 05: 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 cust_customers  CS
left join oe_orderHeaders OH on CS.customer_id = OH.customer_id
where OH.order_id IS NULL
order by CS.customer_id;
```

```
CUSTOMER_ID CUSTOMER_NAME_LAST          ORDER_ID
----------- ------------------------ ----------
     400801 Washington
     402110 Coltrane
     402120 McCoy
     402500 Jones
     403500 Stevenson
     403750 O'Leary
     403760 O'Leary
     404150 Dancer
     404180 Shay
. . . rows omitted
```

Demo 06: 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 oe_orderHeaders OH
Left Join cust_customers CS On CS.customer_id = OH.customer_id
where OH.customer_id Is Null;
```

```
no rows selected
```

Demo 07:   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
, prod_id
, prod_desc As product
, prod_list_price
from prd_products
Left Join oe_orderDetails Using (prod_id)
where order_id Is Null
order by catg_id, prod_id;
```

```
CATG_ID    PROD_ID PRODUCT                                            PROD_LIST_PRICE
------- ---------- -------------------------------------------------- ---------------
APL           1126 Low Energy Washer Dryer combo                                  850
APL           4569 Sized for the apartment                                     349.95
GFD           5000 Cello bag of mixed fingerling potatoes                        12.5
GFD           5001 Dundee Ginger Preserve 12 oz jar                                 5
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                                  2.5
PET           1143 Bird seed mix with more sunflower seeds                        2.5
PET           4567 Our highest end cat tree- you gotta see this                549.99
PET           4568 Satin four-poster cat bed                                   549.99
```

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 08:   These are customers without orders query done using a subquery.
The subquery gets the cust_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 cust_customers
where customer_id Not In (
  select customer_id
  from oe_orderHeaders
);
```

```
CUSTOMER_ID CUSTOMER_NAME_LAST
---------- ------------------------
    400801 Washington
    402110 Coltrane
    402120 McCoy
    402500 Jones
```

```
    403500 Stevenson
    403750 O'Leary
    403760 O'Leary
    404150 Dancer
    404180 Shay
. . .  rows omitted
```

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

Demo 09:   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
, prod_id
, prod_desc as product
, prod_list_price
from prd_products
where prod_id NOT IN  (
    select prod_id
    from oe_order_details)
order by catg_id, prod_id ;
```

```
CATG_ID    PROD_ID PRODUCT                                        PROD_LIST_PRICE
-------  ---------- --------------------------------------------- ---------------
APL          1126 Low Energy washer Dryer combo                              850
APL          4569 Sized for the apartment                                 349.95
GFD          5000 Cello bag of mixed fingerling potatoes                    12.5
GFD          5001 Dundee Ginger Preserve 12 oz jar                             5
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                             2.5
PET          1143 Bird seed mix with more sunflower seeds                   2.5
PET          4567 Our highest end cat tree- you gotta see this           549.99
PET          4568 Satin four-poster cat bed                               549.99
```

4. What can go wrong?

Suppose we want to find employees who are not associated with any orders. Remember that in the order headers 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 10:   Left join Employees to Orders

```
select
  emp_id
, name_last
, order_id
from emp_employees
Left Join oe_orderHeaders On emp_id = sales_rep_id
;
```

```
    EMP_ID NAME_LAST                ORDER_ID
---------- ------------------------ ----------
       100 King
       101 Koch
       102 D'Haa
       103 Hunol
       104 Ernst
       108 Green
       109 Fiet
       110 Chen
       145 Russ                            112
       145 Russ                            540
       145 Russ                           2505
       145 Russ                            405
       145 Russ                            130
       145 Russ                            312
. . . 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 11:   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 emp_employees
Left Join oe_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
       104 Ernst
       108 Green
       109 Fiet
       110 Chen
       146 Partne
. . . rows omitted
18 rows selected.
```

What if we try this with a subquery?

Demo 12:   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 emp_employees
where emp_id Not In (
  select sales_rep_id
  from oe_orderHeaders
);
```
```
no rows selected
```

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

Demo 13:   Subquery version 2-Eliminate the nulls from the subquery.

```
select
  emp_id
, name_last
from emp_employees
where emp_id Not In (
  select sales_rep_id
  from oe_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.