Table of Contents

These are the tables we are using. Note that we have employees with no projects and a department with no employees and employees with no department.

z_em_dept

d_id	d_name
100	Manufacturing
150	Accounting
200	Marketing
250	Research

z_em_emp

e_id	e_name	d_id
1	Jones	150
2	Martin	150
3	Gates	250
4	Anders	100
5	Bossy	NULL
6	Perkins	NULL

z_em_empproj

p_id	e_id
ORDB-10	3
ORDB-10	5
Q4-SALES	2
Q4-SALES	4
ORDB-10	2
Q4-SALES	5

1. Syntax for outer joins (Left, Right)

Outer joins can use the syntax Left Join or Right Join. A left outer join written as

```
From tblA LEFT JOIN tblB
```

will include all rows from table tblA and any matching rows from tblB. The table to the left of the phrase Left Join will have all of its rows returned.

A right outer join written as

```
From tblA RIGHT JOIN tblB
```

will include all rows from table tblB and any matching rows from tblA. The table to the right of the phrase Right Join will have all of its rows returned.

The outer joins are not symmetric.

The word OUTER is optional; you can use Left Outer Join or Left Join.

You will still need to identify the joining columns and code the join phrase.

MySQL does not yet support the Full Outer join syntax that you might know from another dbms.

Demo 01: All departments; employees of those departments if they exist.

```
select D.d_id, d_name, e_id, e_name
from a_testbed.z_em_dept D
LEFT JOIN a_testbed.z_em_emp E on D.d_id = E.d_id
order by e_id;
+------+---------------+------+--------+
| d_id | d_name        | e_id | e_name |
+------+---------------+------+--------+
|  200 | Marketing     | NULL | NULL   |
|  150 | Accounting    |    1 | Jones  |
|  150 | Accounting    |    2 | Martin |
|  250 | Research      |    3 | Gates  |
|  100 | Manufacturing |    4 | Anders |
+------+---------------+------+--------+
```

Demo 02: All employees; assigned departments if they exist. Outer joins are not commutative

```
select d_id, d_name, e_id, e_name
from a_testbed.z_em_emp E
LEFT JOIN a_testbed.z_em_dept D using(d_id)
order by e_id;
+------+---------------+------+---------+
| d_id | d_name        | e_id | e_name  |
+------+---------------+------+---------+
|  150 | Accounting    |    1 | Jones   |
|  150 | Accounting    |    2 | Martin  |
|  250 | Research      |    3 | Gates   |
|  100 | Manufacturing |    4 | Anders  |
| NULL | NULL          |    5 | Bossy   |
| NULL | NULL          |    6 | Perkins |
+------+---------------+------+---------+
```

Demo 03: All employees; assigned departments if they exist.

```
select d_id, d_name, e_id, e_name
from a_testbed.z_em_dept D
RIGHT JOIN a_testbed.z_em_emp  E using(d_id)
order by e_id;
+------+---------------+------+---------+
| d_id | d_name        | e_id | e_name  |
+------+---------------+------+---------+
|  150 | Accounting    |    1 | Jones   |
|  150 | Accounting    |    2 | Martin  |
|  250 | Research      |    3 | Gates   |
|  100 | Manufacturing |    4 | Anders  |
| NULL | NULL          |    5 | Bossy   |
| NULL | NULL          |    6 | Perkins |
+------+---------------+------+---------+
```

Demo 04: Three table outer join. This is all of the departments and their employees if there are any in the department and the projects if the employees have a project.

```
select d_id, d_name, e_id, e_name, p_id
from a_testbed.z_em_dept
LEFT JOIN a_testbed.z_em_emp using(d_id)
LEFT JOIN a_testbed.z_em_empproj using(e_id)
order by e_id;
+------+---------------+------+--------+----------+
| d_id | d_name        | e_id | e_name | p_id     |
+------+---------------+------+--------+----------+
|  200 | Marketing     | NULL | NULL   | NULL     |
|  150 | Accounting    |    1 | Jones  | NULL     |
|  150 | Accounting    |    2 | Martin | Q4-SALES |
|  150 | Accounting    |    2 | Martin | ORDB-10  |
|  250 | Research      |    3 | Gates  | ORDB-10  |
|  100 | Manufacturing |    4 | Anders | Q4-SALES |
+------+---------------+------+--------+----------+
```

Demo 05: Three table outer join. This is all of the employees and their departments if they have one and their projects if they have one

```
select e_id, e_name,d_id, d_name, p_id
from a_testbed.z_em_emp
LEFT JOIN a_testbed.z_em_dept using(d_id)
LEFT JOIN a_testbed.z_em_empproj using(e_id)
order by e_id;
```

```
+------+---------+------+---------------+----------+
| e_id | e_name  | d_id | d_name        | p_id     |
+------+---------+------+---------------+----------+
|    1 | Jones   |  150 | Accounting    | NULL     |
|    2 | Martin  |  150 | Accounting    | Q4-SALES |
|    2 | Martin  |  150 | Accounting    | ORDB-10  |
|    3 | Gates   |  250 | Research      | ORDB-10  |
|    4 | Anders  |  100 | Manufacturing | Q4-SALES |
|    5 | Bossy   | NULL | NULL          | ORDB-10  |
|    5 | Bossy   | NULL | NULL          | Q4-SALES |
|    6 | Perkins | NULL | NULL          | NULL     |
+------+---------+------+---------------+----------+
```

Demo 06: Suppose we want to see all employees and their departments if they have one and the names of their projects if they have one. The following query does not do that. We start with an outer join but then use an inner join which eliminates employees with no projects.

```
select E.e_id, e_name, D.d_id, d_name, p_id
from a_testbed.z_em_emp  E
LEFT JOIN a_testbed.z_em_dept D on D.d_id = E.d_id
JOIN a_testbed.z_em_empproj P on E.e_id = P.e_id
order by E.e_id;
+------+--------+------+---------------+----------+
| e_id | e_name | d_id | d_name        | p_id     |
+------+--------+------+---------------+----------+
|    2 | Martin |  150 | Accounting    | Q4-SALES |
|    2 | Martin |  150 | Accounting    | ORDB-10  |
|    3 | Gates  |  250 | Research      | ORDB-10  |
|    4 | Anders |  100 | Manufacturing | Q4-SALES |
|    5 | Bossy  | NULL | NULL          | Q4-SALES |
|    5 | Bossy  | NULL | NULL          | ORDB-10  |
+------+--------+------+---------------+----------+
```

2. Queries using altgeld_mart tables

2.1. Customers and orders

Demo 07: Customers with orders. This uses an inner join. The customer_id filter is to reduce the volume of output.

```
select customer_id, customer_name_last, order_id
from customer.customers
JOIN orderEntry.orderHeaders using(customer_id)
where customer_id between 404900 and 409030
order by customer_id, order_id;
+-------------+--------------------+----------+
| customer_id | customer_name_last | order_id |
+-------------+--------------------+----------+
|      404900 | Williams           |      520 |
|      404950 | Morris             |      110 |
|      404950 | Morris             |      408 |
|      404950 | Morris             |      411 |
|      404950 | Morris             |      535 |
|      404950 | Morris             |      540 |
|      404950 | Morris             |     4510 |
|      405000 | Day                |      116 |
|      408770 | Clay               |      405 |
|      409030 | Mazur              |      128 |
|      409030 | Mazur              |      130 |
|      409030 | Mazur              |      324 |
+-------------+--------------------+----------+
```

Demo 08: Customers with and without orders. This uses an outer join; Customers Left Join Order Headers. That means we get customers with orders and if the customer has several orders, that customer gets multiple lines in the result set.
We also get rows for the customers in this customer_id range who have no orders and the column for their order id value is null- these customers each get one row.

```
select customer_id, customer_name_last, order_id
from customer.customers
LEFT JOIN orderEntry.orderHeaders using(customer_id)
where customer_id between 404900 and 409030
order by customer_id, order_id;
+-------------+--------------------+----------+
| customer_id | customer_name_last | order_id |
+-------------+--------------------+----------+
|      404900 | Williams           |      520 |
|      404950 | Morris             |      110 |
|      404950 | Morris             |      408 |
|      404950 | Morris             |      411 |
|      404950 | Morris             |      535 |
|      404950 | Morris             |      540 |
|      404950 | Morris             |     4510 |
|      405000 | Day                |      116 |
|      408770 | Clay               |      405 |
|      408777 | Morise             |     NULL |
|      409010 | Morris             |     NULL |
|      409020 | Max                |     NULL |
|      409030 | Mazur              |      128 |
|      409030 | Mazur              |      130 |
|      409030 | Mazur              |      324 |
+-------------+--------------------+----------+
15 rows in set (0.00 sec)
```

Demo 09: Now consider this join. I change the join to a right join. The result set is the same as the inner join used previously. Why?

```
select customer_id, customer_name_last, order_id
from customer.customers
RIGHT JOIN orderEntry.orderHeaders using(customer_id)
where customer_id between 404900 and 409030
order by customer_id, order_id;
+-------------+--------------------+----------+
| customer_id | customer_name_last | order_id |
+-------------+--------------------+----------+
|      404900 | Williams           |      520 |
|      404950 | Morris             |      110 |
|      404950 | Morris             |      408 |
|      404950 | Morris             |      411 |
|      404950 | Morris             |      535 |
|      404950 | Morris             |      540 |
|      404950 | Morris             |     4510 |
|      405000 | Day                |      116 |
|      408770 | Clay               |      405 |
|      409030 | Mazur              |      128 |
|      409030 | Mazur              |      130 |
|      409030 | Mazur              |      324 |
+-------------+--------------------+----------+
12 rows in set (0.03 sec)
```

In our database we have a foreign key in the order headers table that refers back to the customer table and to the cust_id in the customer table.

```
create table orderEntry.orderHeaders(
    order_id      int unsigned  not null
  , order_date    datetime      not null
  , customer_id   int unsigned  not null
. . .
  , constraint ord_cust_fk foreign key(customer_id) references
orderEntry.customers(customer_id)
. . . )
```

I also set the customer_id in the order headers table as Not null. This means that every row in the order headers table must have a value for the customer_id (it is Not null) and that customer_id in the order header must match a customer_id in the customers tables (foreign key reference) .

The outer join in this query is asking for all orders whether or not they match a customer. But our database is set up so that every order header row is matched with a customer. So it does not make sense to ask to see order headers rows that do not match a customer. In this case you should use an inner join. Using an outer join when it is logically impossible to return unmatched rows is inefficient. Someone reading your query would assume you have made a mistake someplace but they would not know what the mistake is- is the database badly designed and allows the entry of orders that do not belong to a customer (who pays for those orders?), or did you get the join order incorrect?

2.2. Products and orders

These are limited to products in the MUS category to reduce the volume of output

Demo 10: First an inner join- these show products which have been ordered- each product id must match a product id on an order detail row

```
select PR.prod_id, PR.prod_desc, PR.catg_id, OD.order_id
from product.products  PR
join orderEntry.orderDetails  OD  on PR.prod_id = OD.prod_id
where PR.catg_id  in ('MUS')
order by PR.prod_id;
+---------+-----------------------------------+---------+----------+
| prod_id | prod_desc                         | catg_id | order_id |
+---------+-----------------------------------+---------+----------+
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |      413 |
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |      525 |
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |      552 |
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |      715 |
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |     2218 |
|    2014 | Bix Beiderbecke - Tiger Rag       | MUS     |     3518 |
|    2412 | David Newman - Davey Blue         | MUS     |      525 |
|    2412 | David Newman - Davey Blue         | MUS     |     2225 |
|    2746 | Charles Mingus - Blues & Politics | MUS     |      525 |
|    2746 | Charles Mingus - Blues & Politics | MUS     |     2218 |
|    2747 | Charles Mingus - Blues & Roots    | MUS     |      520 |
|    2947 | Ornette Coleman - Sound Grammer   | MUS     |      525 |
|    2947 | Ornette Coleman - Sound Grammer   | MUS     |     2225 |
|    2984 | John Coltrane - Lush Life         | MUS     |      413 |
|    2984 | John Coltrane - Lush Life         | MUS     |      552 |
|    2984 | John Coltrane - Lush Life         | MUS     |      715 |
|    2984 | John Coltrane - Lush Life         | MUS     |     3518 |
+---------+-----------------------------------+---------+----------+
17 rows in set (0.07 sec)
```

Demo 11: How many products do we have in the MUS category?
We have 11 products; looking at the previous result set, 6 of these products were sold (Several were sold on more than one order.)

```
select PR.prod_id, PR.prod_desc, PR.catg_id
from product.products  PR
where catg_id in ('MUS')
order by PR.prod_id;
+---------+---------------------------------------------+---------+
| prod_id | prod_desc                                   | catg_id |
+---------+---------------------------------------------+---------+
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |
|    2234 | Charles Mingus - Pithecanthropus Erectus    | MUS     |
|    2337 | John Coltrane - Blue Train                  | MUS     |
|    2412 | David Newman - Davey Blue                   | MUS     |
|    2487 | Stanley Turrentine - Don't Mess With Mr. T  | MUS     |
|    2746 | Charles Mingus - Blues & Politics           | MUS     |
|    2747 | Charles Mingus - Blues & Roots              | MUS     |
|    2933 | David Newman - I Remember Brother Ray       | MUS     |
|    2947 | Ornette Coleman - Sound Grammer             | MUS     |
|    2984 | John Coltrane - Lush Life                   | MUS     |
|    2987 | Stanley Turrentine - Ballads                | MUS     |
+---------+---------------------------------------------+---------+
11 rows in set (0.00 sec)
```

Demo 12: We can use an outer join to get both ordered and un-ordered products. I have highlighted the rows where the order id is null; those are the products that were never sold.

```
select PR.prod_id, prod_desc, catg_id, order_id
from product.products  PR
LEFT JOIN orderEntry.orderDetails   OD  on PR.prod_id = OD.prod_id
where catg_id in ('MUS')
order by PR.prod_id;
+---------+---------------------------------------------+---------+----------+
| prod_id | prod_desc                                   | catg_id | order_id |
+---------+---------------------------------------------+---------+----------+
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      413 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      525 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      552 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      715 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |     2218 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |     3518 |
|    2234 | Charles Mingus - Pithecanthropus Erectus    | MUS     |     NULL |
|    2337 | John Coltrane - Blue Train                  | MUS     |     NULL |
|    2412 | David Newman - Davey Blue                   | MUS     |      525 |
|    2412 | David Newman - Davey Blue                   | MUS     |     2225 |
|    2487 | Stanley Turrentine - Don't Mess With Mr. T  | MUS     |     NULL |
|    2746 | Charles Mingus - Blues & Politics           | MUS     |      525 |
|    2746 | Charles Mingus - Blues & Politics           | MUS     |     2218 |
|    2747 | Charles Mingus - Blues & Roots              | MUS     |      520 |
|    2933 | David Newman - I Remember Brother Ray       | MUS     |     NULL |
|    2947 | Ornette Coleman - Sound Grammer             | MUS     |      525 |
|    2947 | Ornette Coleman - Sound Grammer             | MUS     |     2225 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      413 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      552 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      715 |
|    2984 | John Coltrane - Lush Life                   | MUS     |     3518 |
|    2987 | Stanley Turrentine - Ballads                | MUS     |     NULL |
+---------+---------------------------------------------+---------+----------+
22 rows in set (0.00 sec)
```

Demo 13: This query gives us rows for the same products- why are we missing values in the first column which shows the product id? Every product has a product Id!

```
select OD.prod_id, prod_desc, catg_id, order_id
from product.products PR
LEFT JOIN orderEntry.orderDetails   OD  on PR.prod_id = OD.prod_id
where catg_id in ('MUS')
order by OD.prod_id, order_id;
+---------+---------------------------------------------+---------+----------+
| prod_id | prod_desc                                   | catg_id | order_id |
+---------+---------------------------------------------+---------+----------+
|    NULL | John Coltrane - Blue Train                  | MUS     |     NULL |
|    NULL | David Newman - I Remember Brother Ray       | MUS     |     NULL |
|    NULL | Stanley Turrentine - Ballads                | MUS     |     NULL |
|    NULL | Stanley Turrentine - Don't Mess With Mr. T  | MUS     |     NULL |
|    NULL | Charles Mingus - Pithecanthropus Erectus    | MUS     |     NULL |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      413 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      525 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      552 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |      715 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |     2218 |
|    2014 | Bix Beiderbecke - Tiger Rag                 | MUS     |     3518 |
|    2412 | David Newman - Davey Blue                   | MUS     |      525 |
|    2412 | David Newman - Davey Blue                   | MUS     |     2225 |
|    2746 | Charles Mingus - Blues & Politics           | MUS     |      525 |
|    2746 | Charles Mingus - Blues & Politics           | MUS     |     2218 |
|    2747 | Charles Mingus - Blues & Roots              | MUS     |      520 |
|    2947 | Ornette Coleman - Sound Grammer             | MUS     |      525 |
|    2947 | Ornette Coleman - Sound Grammer             | MUS     |     2225 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      413 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      552 |
|    2984 | John Coltrane - Lush Life                   | MUS     |      715 |
|    2984 | John Coltrane - Lush Life                   | MUS     |     3518 |
+---------+---------------------------------------------+---------+----------+
22 rows in set (0.00 sec)
```

What I did is switch the column alias for the first column and for the sort key to use the order detail s table. If I am looking for the product id in the order details table, the products which are not ordered do not have a value for that column and display as nulls.