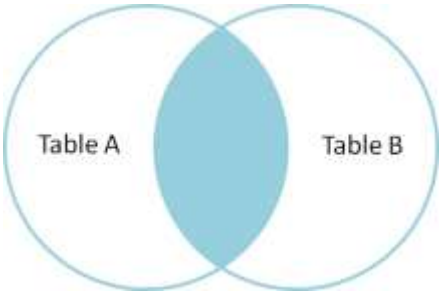
Using Joins

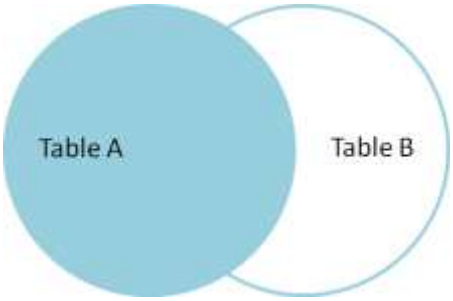
Inner join vs. outer join: Give me everything that's in TableA **AND** TableB (matching values: inner join); or give me everything that's in TableA **OR** TableB (matching and unmatching values: outer join).

Definitions:

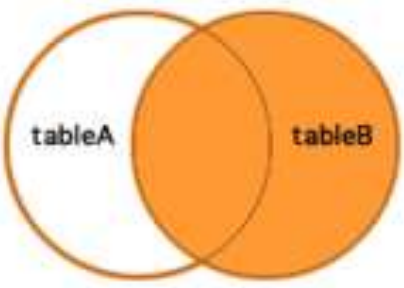
• (INNER) JOIN: Returns rows with matching values in both tables.



• LEFT (OUTER) JOIN: Returns all rows from left table, even if there are no matches in the right table. If there is no match, the right side will contain null.

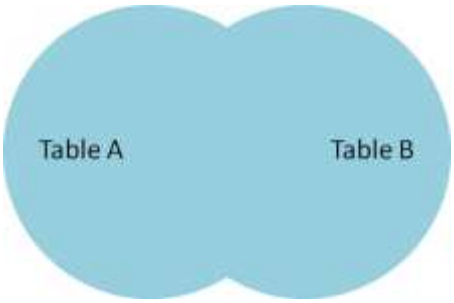


• RIGHT (OUTER) JOIN: Return all rows from the right table, even if there are no matches in the left table. If there is no match, the left side will contain null.



• FULL (OUTER) JOIN: returns all records in Table A **and** Table B, with matching records from both sides where available, any unmatched records will display null.

Note: MySQL does not support full outer joins at this time.



MySQL Full Outer Join (emulation)

```
SELECT * FROM t1
LEFT JOIN t2 ON t1.pk = t2.fk
UNION ALL
SELECT * FROM t1
RIGHT JOIN t2 ON t1.pk = t2.fk
WHERE t1.id IS NULL; // left inclusive join, then right exclusive join using UNION ALL.
```

Note: Second query doesn't introduce "duplicates" (that is, rows already returned by first query), which emulates a FULL OUTER JOIN.

Table Examples

select * from tablea;

```
+---------+------------+
| tbla_id | tbla_value |
+---------+------------+
|       1 | red        |
|       2 | green      |
|       3 | blue       |
|       4 | yellow     |
|       5 | silver     |
+---------+------------+
```

select * from tableb;

```
+---------+---------+------------+
| tblb_id | tbla_id | tblb_value |
+---------+---------+------------+
|       1 |       1 | apple      |
|       2 |    NULL | snow       |
|       3 |       3 | sky        |
|       4 |       4 | canary     |
|       5 |    NULL | diamond    |
+---------+---------+------------+
```

Join Examples

Inner Join: returns only matching records in **both** Table A and Table B.

Equi-join (repeats column names): doesn't require same column names

```
SELECT *
FROM tablea, tableb
WHERE tablea.tbla_id = tableb.tbla_id;
```

```
+---------+------------+---------+---------+------------+
| tbla_id | tbla_value | tblb_id | tbla_id | tblb_value |
+---------+------------+---------+---------+------------+
|       1 | red        |       1 |       1 | apple      |
|       3 | blue       |       3 |       3 | sky        |
|       4 | yellow     |       4 |       4 | canary     |
+---------+------------+---------+---------+------------+
```

(INNER) Join On (repeats column names): doesn't require same column names

```
SELECT *
FROM tablea
JOIN tableb ON tablea.tbla_id = tableb.tbla_id;
```

```
+---------+------------+---------+---------+------------+
| tbla_id | tbla_value | tblb_id | tbla_id | tblb_value |
+---------+------------+---------+---------+------------+
|       1 | red        |       1 |       1 | apple      |
|       3 | blue       |       3 |       3 | sky        |
|       4 | yellow     |       4 |       4 | canary     |
+---------+------------+---------+---------+------------+
```

(INNER) Join Using (doesn't repeat column names): does require same column names

```
SELECT *
FROM tablea JOIN tableb USING (tbla_id);
```

```
+---------+------------+---------+------------+
| tbla_id | tbla_value | tblb_id | tblb_value |
+---------+------------+---------+------------+
|       1 | red        |       1 | apple      |
|       3 | blue       |       3 | sky        |
|       4 | yellow     |       4 | canary     |
+---------+------------+---------+------------+
```

Natural-join (doesn't repeat column names): does require same column names
While a natural join is easy to use, with simple syntax, be careful using **NATURAL JOIN**: A natural join is based on **all** columns in the two tables that have the same name. It selects rows from the two tables that have equal values in the relevant columns.

This is why it is *important* that your tables have unique attribute names!

```
SELECT *
FROM tablea NATURAL JOIN tableb;
```

```
+---------+------------+---------+------------+
| tbla_id | tbla_value | tblb_id | tblb_value |
+---------+------------+---------+------------+
|       1 | red        |       1 | apple      |
|       3 | blue       |       3 | sky        |
|       4 | yellow     |       4 | canary     |
+---------+------------+---------+------------+
```

Outer Joins: returns matching and unmatched records in Table A and Table B.

LEFT (OUTER) JOIN: Returns all rows from left table, even if there are no matches in the right table. If there is no match, the right side will contain null.

```
SELECT *
FROM tablea
LEFT OUTER JOIN tableb ON tablea.tbla_id = tableb.tbla_id;
```

```
+---------+------------+---------+---------+------------+
| tbla_id | tbla_value | tblb_id | tbla_id | tblb_value |
+---------+------------+---------+---------+------------+
|       1 | red        |       1 |       1 | apple      |
|       2 | green      |    NULL |    NULL | NULL       |
|       3 | blue       |       3 |       3 | sky        |
|       4 | yellow     |       4 |       4 | canary     |
|       5 | silver     |    NULL |    NULL | NULL       |
+---------+------------+---------+---------+------------+
```

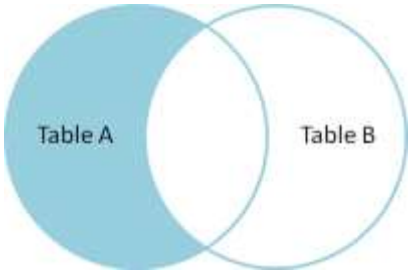
RIGHT (OUTER) JOIN: Return all rows from the right table, even if there are no matches in the left table. If there is no match, the left side will contain null.

```
SELECT *
FROM tablea
RIGHT OUTER JOIN tableb ON tablea.tbla_id = tableb.tbla_id;
```

```
+---------+------------+---------+---------+------------+
| tbla_id | tbla_value | tblb_id | tbla_id | tblb_value |
+---------+------------+---------+---------+------------+
|       1 | red        |       1 |       1 | apple      |
|    NULL | NULL       |       2 |    NULL | snow       |
|       3 | blue       |       3 |       3 | sky        |
|       4 | yellow     |       4 |       4 | canary     |
|    NULL | NULL       |       5 |    NULL | diamond    |
+---------+------------+---------+---------+------------+
```

To return the set of records **only** in Table A, **but not** in Table B, perform the same left outer join, then exclude the records not required from the right side via a where clause.

```
SELECT *
FROM tablea
LEFT OUTER JOIN tableb ON tablea.tbla_id = tableb.tbla_id
WHERE tableb.tbla_id IS null;
```

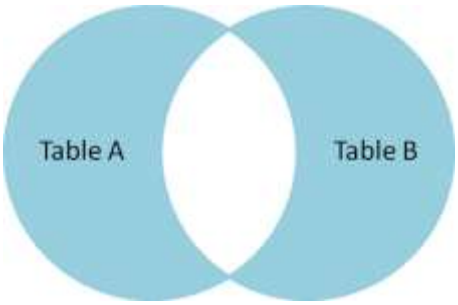


Note: Similarly, can return the set of records **only** in Table B, **but not** in Table A.

To return the set of records unique to Table A and Table B, we perform the same full outer join, then exclude the records we don't want from both sides via a where clause.

Note: MySQL does not support full outer joins at this time. (**Will create error!**)

```
SELECT *
FROM tablea
FULL OUTER JOIN tableb ON tablea.tbla_id = tableb.tbla_id
WHERE tablea.tbla_id IS null OR tableb.tbla_id IS null;
```



Log in to *YOUR* database using MySQL Workbench, AND through SSH.

First: drop *ALL* existing tables! Then...

source db/gmc.sql
--or...
\. db/gmc.sql

Reports:

1. Display your grants:
2. Display all of the tables in your database:
3. Display the structure for each table:
4. Display the data for each table:
5. **EQUI-JOIN** (old-style join), returns rows with matching values in **where** clause, does ***not*** require common named attributes (repeats join column names): Return all rows from dealership and vehicle with matching values, INCLUDING duplicate columns.
6. **JOIN ON**, returns rows with matching values in **from** clause, does ***not*** require common named attributes (repeats join column names): (same as above) return all rows from dealership and vehicle with matching values, INCLUDING duplicate columns.
7. **NATURAL JOIN** syntax, tables ***MUST*** share same column name/values (does ***not*** repeat join column names): (same as above) return all rows from dealership and vehicle with matching values, NO duplicate columns.
8. **JOIN USING** syntax, same as **NATURAL JOIN** (tables same column name/values): (same as above) return all rows from dealership and vehicle with matching values, NO duplicate columns.

NOTE:** NATURAL JOIN and JOIN USING ***must include common attribute names.

9. **NATURAL JOIN** syntax (tables MUST share same column name/values): return all rows from customer and purchased vehicles (that is, NO nulls), NO duplicate columns.
10. **LEFT OUTER JOIN:** returns rows matching values on join condition, and *all* rows in left table (assignment) with unmatched values (nulls) in right table, use **JOIN ON**. Return all rows from customer, as well as purchased and unpurchased vehicles, (that is, matched and unmatched values in the vehicle table).

11. Same as above: use **JOIN USING**, same return, eliminates duplicate **asn_id** column.

12. **RIGHT OUTER JOIN:** returns rows matching values on join condition, and *all* rows in right table (vehicle) with unmatched values (nulls) in left table (assignment), use **JOIN ON**. Return all rows from vehicle, as well as matching or unmatching customers, (that is, matched and unmatched values in the assignment table).

13. Same as above: use **JOIN USING**, same return, eliminates duplicate **asn_id**

Log in to mycompany database using MySQL Workbench, AND through SSH.

1. Using MySQLWorkbench, AND 2. Using SSH Client (a. Windows, b. Mac):
a) create a **new connection** with the same connection parameter values as your initial connection, except…
b) change your username and password to **mycompany**.
c) Beginning with report #5, format all dollar amounts, rounded to two decimal places.

Reports:

1. Display your grants:

2. Display all of the tables in the database:

3. Display the structure for each table:

4. Display the data for each table:

5. List all orders showing order number, amount, customer name ("company"), and customer's credit limit (using equi-join and join on):

6. List each salesperson the city and region where they work (using equi-join and join on):

7. List the offices' cities, targets, and names and titles of the salespersons' managers (using equi-join and join on):

8. List offices' cities, with targets over $600,000, and their manager info. (name, title) (using equi-join and join on):

9. List all orders, showing amounts and product descriptions (using equi-join and join on):

10. List orders over $25,000, including the name of the salesperson who took the order, and the name of the customer ("company") who placed it (using equi-join and join on):

11. List orders over $25,000, including the name of the customer ("company") who placed the order, the customer's salesperson, and the office where the salesperson works (using equi-join and join on):

12. List all salespeople and the cities in which they work, including salespeople with no cities: