So far we've used the keyword JOIN which really means INNER JOIN

```
USE v_art;
```

So if we had

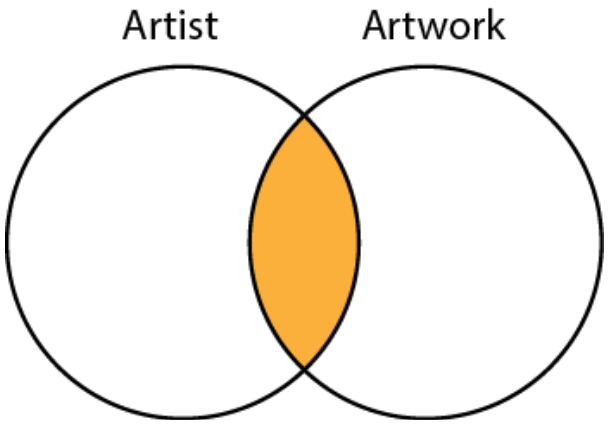
```
SELECT fname, lname, title
FROM artist
    JOIN artwork
    ON artist.artist_id = artwork.artist_id;
```

Then we really are saying do an inner join.

```
SELECT fname, lname, title
FROM artist
    INNER JOIN artwork
    ON artist.artist_id = artwork.artist_id;
```

Only the matching rows of each table show up. Or in other words only the primary keys of the artist table match up with rows that reference that same foreign key in the artwork table show up.

The Venn diagram represents how only the related data would show up. It shows only the inner portion filled in because only the rows of the two tables with matching values will show up in the results. Only the artist with a primary key value that shows up at a foreign key value will show.



But remember, we had an artist that was in the artist table, Michaelangelo, but he didn't have any of his artwork listed in our artwork table. Michaelangelo would be over here in the white part of the left table. So in other words, the primary key associate with Michaelangelo was not used as a foreign key anywhere in the artwork table. If we wanted him to show up in our result

set, this would require an OUTER JOIN. Outer joins show all the rows from one table and then only the matching rows of the other table. If we add the keyword LEFT before the join this says, take the table on the left (or the first table) and show me everything in that table whether it has a relationship with the other table or not and the table on the right or second table will only show the related data between those two tables.

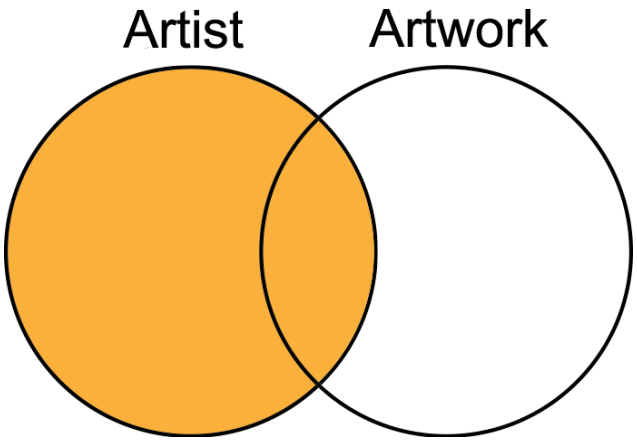
```
SELECT fname, lname, title
FROM artist
        LEFT JOIN artwork
        ON artist.artist_id = artwork.artist_id;
```

We can see the Michaelangelo now shows up in the title column there is a null value because he doesn't have any artwork in that table. Michaelangelo's primary key is not used as a foreign key in the artwork table.

The keyword LEFT or RIGHT, we will see RIGHT in a moment, both refer to OUTER joins. The term outer is implied. You can use the keyword if you want but it is not required, just as the keyword inner was not required. This would run the exact same way.

```
SELECT fname, lname, title
FROM artist
        LEFT OUTER JOIN artwork
        ON artist.artist_id = artwork.artist_id;
```

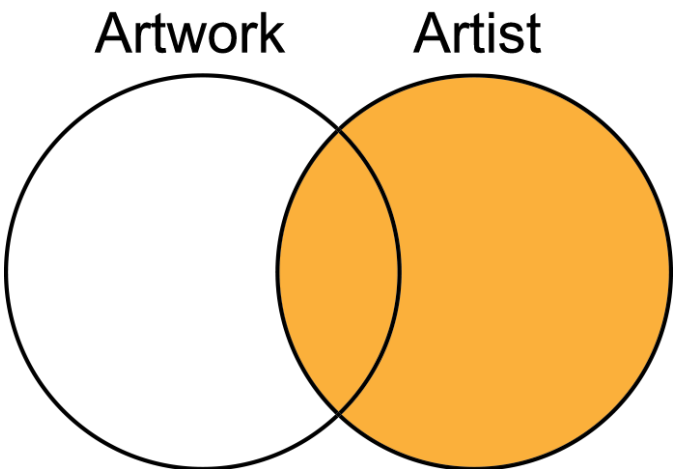
Let's take a look at how this looks on a Venn diagram. Every row of data is showing from the artist table but only the related rows from the artwork table. In our data, we don't have any artwork that is not associate with an artist, but if we did it would not show up.



We could reverse the order of the tables and use the RIGHT keyword and it would return the same results. Now we are saying the table on the right is where we want every row to be returned whether it has a relationship to the other table or not.

```
SELECT fname, lname, title
FROM artwork
        RIGHT OUTER JOIN artist
        ON artist.artist_id = artwork.artist_id;
```

So our Venn diagram is the same but just reversed as well.

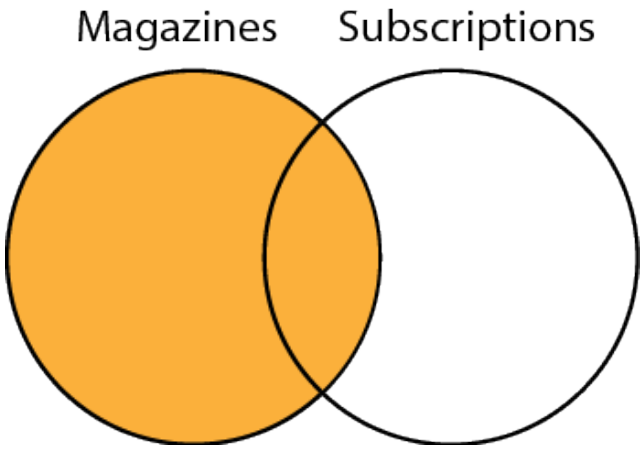


Let's create a query from the magazine database. This time let's use some table aliases as well.

```
SELECT magazineName, magazinePrice, subscriptionStartDate
FROM magazine m
        JOIN subscription s
   ON id = magazine_id;
```

We want the name of the magazine and the price to show up in our result set and those values both will come from the magazine table. We also want any subscription start dates associated with those magazines to show up. That date will come from the subscription table. So we will need to join those two tables together. Now the matching rows from each table will show up

But remember we have a magazine called MySQL Magic that doesn't have any subscribers yet. It does not show up in the results because there is no foreign key in the subscription table that matches the primary key of MySQL Magic. But maybe we want it to show up in the result set. We want all magazines to show up whether they have a subscription date or not.




```
SELECT magazineName, magazinePrice, subscriptionStartDate
FROM magazine m
        LEFT JOIN subscription s
   ON id = magazine_id;
```

The result set on this one will now show the magazine 'MySQL Magic' and its price but because there is not subscription date associated with it the subscription start date column will have a null value in it. There is no subscription start date because no one has subscribed to that magazine yet.

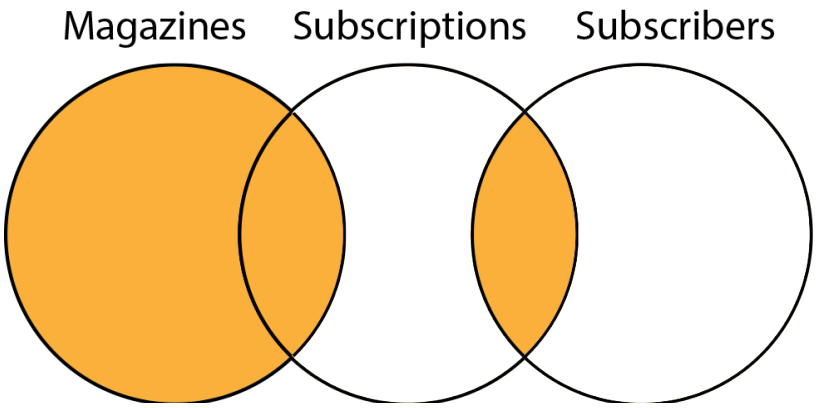
Again remember with the keywords LEFT and RIGHT, OUTER is implied. It's the same if you say LEFT OUTER JOIN.

```
SELECT magazineName, magazinePrice, subscriptionStartDate
FROM magazine m
        LEFT OUTER JOIN subscription s
   ON id = magazine_id;
```

Now what if we also want to show not only the subscription start date but the last name of who subscribes to that magazine? We'd add the first and last name of the subscriber to the result set but since those are in the subscriber table we need to join that one to the query as well.

```
SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
FROM magazine m
        LEFT JOIN subscription s
        ON m.id = magazine_id
        JOIN subscriber sr
        ON sr.id = subscriber_id;
```

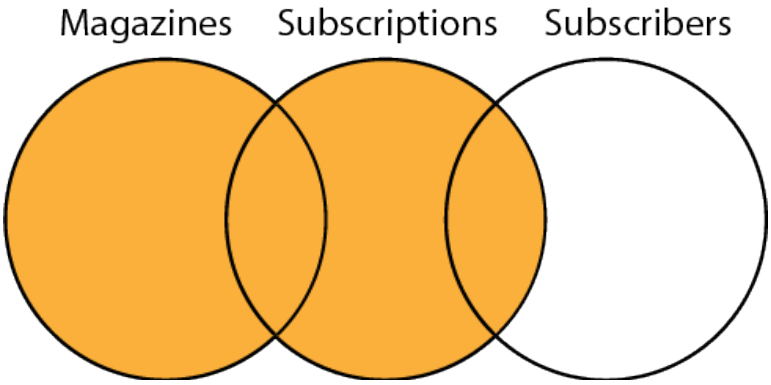
It works, but did you notice we lost MySQL Magic from the result set even though the first part of the query was the same LEFT JOIN? That's because when we said JOIN subscriber that was an inner join for that particular table. So it was not going to be able to get all the results from the first join included with it.



We'd have to also have it be an outer join in order for MySQL Magic to still show up.

```
SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
FROM magazine m
        LEFT JOIN subscription s
        ON m.id = magazine_id
        LEFT JOIN subscriber sr
        ON sr.id = subscriber_id;
```

Now MySQL Magic shows up. And the Venn would look like this.

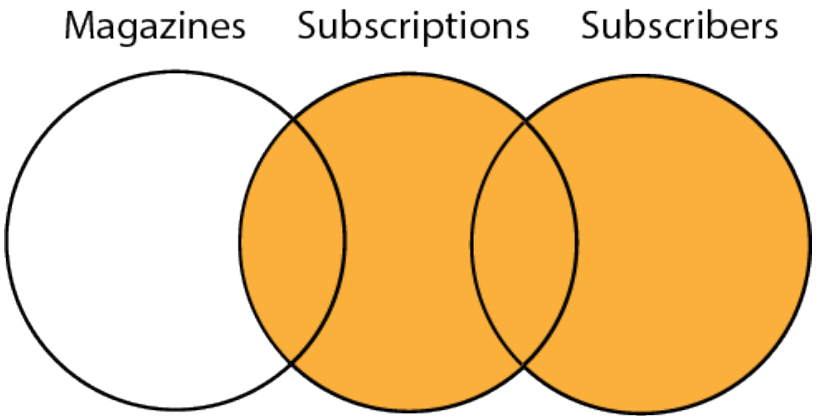


OK one last type of OUTER join. That's the Full Outer Join. It is not used often but I do want to show one just so you can see it.

With our magazine database we also have subscribers who have not subscribed to any magazines yet. For example if we run this RIGHT outer join we can see that Walter Wong is showing up even though he has not subscribed to a magazine yet.

```
SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
FROM magazine m
    RIGHT JOIN subscription s
    ON m.id = magazine_id
    RIGHT JOIN subscriber sr
    ON sr.id = subscriber_id;
```

The Venn diagram would look like this with all the subscribers showing up but only the related subscriptions.



What if we want to show all the magazines even if they don't have a subscription and all the subscribers even if they don't have a subscription all in one query? We'd have to use a FULL OUTER JOIN. Because MySQL doesn't support the actuall keywords or syntax of FULL OUTER JOIN (SQL does by the way but not MySQL) we use a work around with the keyword UNION to simulate the full outer join.

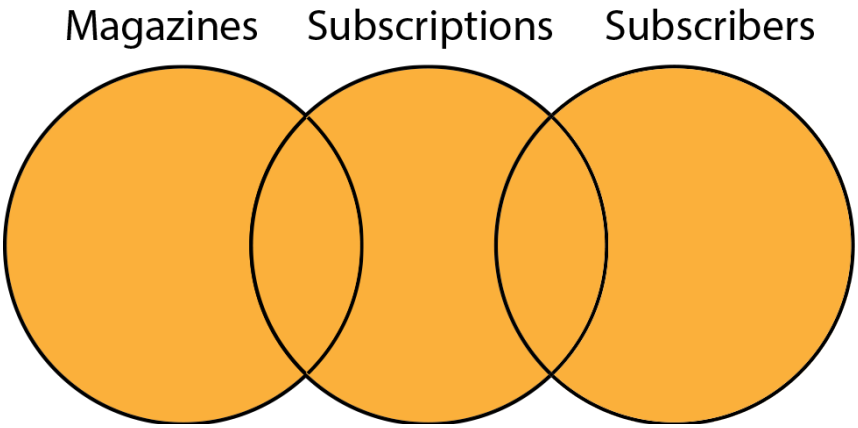
So I am basically just going to but those last two queries together with the keyword UNION between them. This will give us what we want.

```
    SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
    FROM magazine m
        LEFT JOIN subscription s
        ON m.id = magazine_id
        LEFT JOIN subscriber sr
        ON sr.id = subscriber_id
UNION
    SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
```

```
FROM magazine m
        RIGHT JOIN subscription s
        ON m.id = magazine_id
        RIGHT JOIN subscriber sr
        ON sr.id = subscriber_id;
```

The first SELECT clause gives us all the magazines even with no subscribers and the second SELECT gives us all the subscribers even if haven't subscribed to any magazine yet.

So the venn diagram would be like overlaying the last two diagrams on top of each other.



This might be something you'd use with an exception report to see out of the ordinary data. You could filter it with null conditions to see what magazines you might want to promote or advertise and which customers might need a discount code sent to them to motivate them to sign up for a magazine.

```
        SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
        FROM magazine m
                LEFT JOIN subscription s
                ON m.id = magazine_id
                LEFT JOIN subscriber sr
                ON sr.id = subscriber_id
        WHERE subscriptionStartDate IS NULL
UNION
        SELECT magazineName, magazinePrice, subscriptionStartDate, subscriberLastName
        FROM magazine m
                RIGHT JOIN subscription s
                ON m.id = magazine_id
                RIGHT JOIN subscriber sr
                ON sr.id = subscriber_id
        WHERE subscriptionStartDate IS NULL;
```

Now if we had hundreds and hundreds of customers and magazines, we can quickly see what might need some attention to get our business profits up.

So again outer joins are not as common as inner joins and full outer joins are even less common, but this was a quick look at them so you are familiar with them.