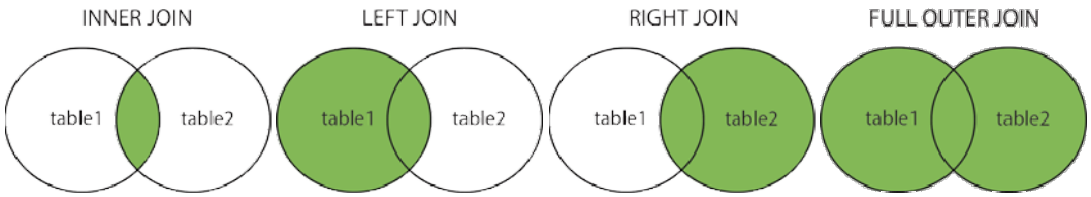
Different Types of SQL JOINs

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table, For example the join operator returns inner rows for customers who placed orders and outer rows for customers who didn't place orders, with *NULLs* in the order attributes
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table




```
-------------------------------------------------------------------------------------
--Part 1: Returns customers and their orders, including customers who place who did not
place any orders (NULL). C.custid, C.companyname, O.orderid, O.orderdate

SELECT C.custid, C.companyname, O.orderid, O.orderdate
FROM Sales.Customers AS C
LEFT JOIN Sales.Orders AS O
ON C.custid = O.custid

--Part 2: Suppose you need to return only customers who did not place any orders or, more
--technically speaking, you need to return only outer rows.

SELECT C.custid, C.companyname, O.orderid, O.orderdate
FROM Sales.Customers AS C
LEFT JOIN Sales.Orders AS O
ON C.custid = O.custid
WHERE O.orderid IS NULL

-------------------------------------------------------------------------------------
--Part 3: Return the count of orders for each customer: COUNT(*) won't work because it
--will add the NULL values also.

SELECT C.custid, COUNT(O.orderid) AS numorders, O.orderdate
FROM Sales.Customers AS C
LEFT JOIN Sales.Orders AS O
ON C.custid = O.custid
GROUP BY C.custid

-------------------------------------------------------------------------------------
--Return US customers, and for each customer return the total number of orders and total
--quantities(qty in Sales.OrdersDetails column):Tables involved: Sales.Customers,
Sales.Orders, and Sales.OrderDetails
-- Note since its each customer and their total number of orders you use DISTINCT
--- **Aggregrates** GROUP BY
```

```sql
SELECT C.custid, COUNT(DISTINCT O.orderid) AS numorders, SUM(OD.qty) AS totalqty
FROM Sales.Customers AS C
JOIN Sales.Orders AS O
ON C.custid = O.custid
JOIN Sales.OrderDetails AS OD
ON OD.orderid = O.orderid
WHERE C.country = 'USA'
GROUP BY C.custid

-----------------------------------------------------------------------------------
--Return customers with orders placed in the first six months of2016, with their orders:
--Tables involved: Sales.Customers and Sales.Orders. custid, companyname,orderid,
orderdate

SELECT C.custid, C.companyname, O.orderid,O.orderdate
FROM Sales.Customers AS C
JOIN Sales.Orders AS O
ON C.custid = O.custid
WHERE O.orderdate  BETWEEN '20160101' AND '20160630'

-----------------------------------------------------------------------------------
--Write a query that returns five columns from AP database.VendorID/VendorName for
Vendors table
--InvoiceNumber /InvoiceDate/: Balance -- InvoiceTotal - PaymentTotal - CreditTotal AS
Balance
--from Invoice table. Return Vendors that placed orders where Balance > 0. Sort by
VendorName

USE AP;

SELECT V.VendorId, V.VendorName, I.InvoiceNumber, InvoiceDate,
I.InvoiceTotal - PaymentTotal - CreditTotal AS Balance
FROM Vendors AS V
JOIN Invoices AS I
ON V.VendorID = I.VendorID
WHERE InvoiceTotal - PaymentTotal - CreditTotal > 0
ORDER BY VendorName ;

-----------------------------------------------------------------------------------
--Write a query from AP database that returns from GLAccounts Table
--AccountNo and AccountDescription and InvoiceLineItems Table. Return one row for each
--Account number that has never been used. Sort by AccountNo
--Account never beeen used(NULL)

SELECT GL.AccountNo, GL.AccountDescription
FROM GLAccounts AS GL
LEFT JOIN InvoiceLineItems AS I
ON GL.AccountNo = I.AccountNo
WHERE I.AccountNo IS NULL
ORDER BY GL.AccountNo;
```

The SQL UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- UNION ALL : With duplicates. UNION: Without duplicates

```
--------------------------------------------------------------------------------
--Write a query that returns distinct locations that are either employee locations
-- or customer location form HR.Employees and Sales.Customers **UNION**
--Columns country, region and city
SELECT country, region, city
FROM HR.Employees
UNION
SELECT country, region, city
FROM Sales.Customers

--------------------------------------------------------------------------------
```

The SQL INTERSECT Operatuor

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.
Just as with the UNION operator, the same rules apply when using the INTERSECT operator. MySQL does not support the INTERSECT operator.

```
--------------------------------------------------------------------------------
--the following code returns distinct locations, country, region, city
--that are both employee locations and customer locations from HR.Employees
--and Sales.customers Tables, in other words where they intersect

SELECT country, region, city
FROM HR.Employees
INTERSECT
SELECT country, region, city
FROM Sales.Customers;
```