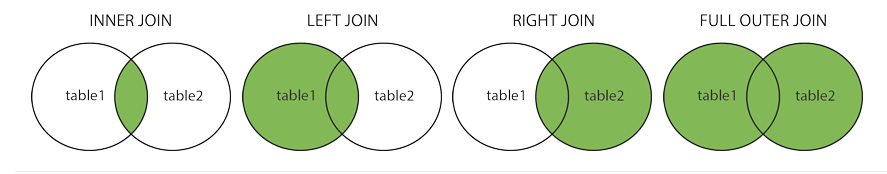
* SELECT DISTINCT Country FROM Customers;
* SELECT COUNT(DISTINCT Country) FROM Customers;
* SELECT Count(\*) AS DistinctCountries  
  FROM (SELECT DISTINCT Country FROM Customers);
* SELECT \* FROM Customers  
  WHERE Country='Germany' AND (City='Berlin' OR City='München');
* SELECT \* FROM Customers  
  WHERE NOT Country='Germany' AND NOT Country='USA';
* SELECT \* FROM Customers  
  ORDER BY Country ASC, CustomerName DESC;
* INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
  VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
* SELECT CustomerName, ContactName, Address  
  FROM Customers  
  WHERE Address IS NULL;
* SELECT CustomerName, ContactName, Address  
  FROM Customers  
  WHERE Address IS NOT NULL;
* UPDATE Customers  
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
  WHERE CustomerID = 1;
* DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
* SELECT TOP 3 \* FROM Customers  
  WHERE Country='Germany';
* SELECT MIN(Price) AS SmallestPrice  
  FROM Products;
* SELECT MAX(Price) AS LargestPrice  
  FROM Products;
* SELECT AVG(Price)  
  FROM Products;
* SELECT SUM(Quantity)  
  FROM OrderDetails;
* SELECT \* FROM Customers  
  WHERE CustomerName LIKE 'a%';
* SELECT \* FROM Customers  
  WHERE CustomerName NOT LIKE 'L\_n\_on';
* SELECT \* FROM Customers  
  WHERE Country IN (SELECT Country FROM Suppliers);
* SELECT \* FROM Customers  
  WHERE Country NOT IN ('Germany', 'France', 'UK');
* SELECT \* FROM Products  
  WHERE Price BETWEEN 10 AND 20;
* SELECT \* FROM Products  
  WHERE Price BETWEEN 10 AND 20  
  AND CategoryID NOT IN (1,2,3);
* SELECT \* FROM Products  
  WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'  
  ORDER BY ProductName;
* SELECT \* FROM Orders  
  WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
* SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address  
  FROM Customers;
* SELECT o.OrderID, o.OrderDate, c.CustomerName  
  FROM Customers AS c, Orders AS o  
  WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;



* SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
  FROM Orders  
  INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
* SELECT Customers.CustomerName, Orders.OrderID  
  FROM Customers  
  LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
  ORDER BY Customers.CustomerName;
* SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
  FROM Orders  
  RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
  ORDER BY Orders.OrderID;
* SELECT Customers.CustomerName, Orders.OrderID  
  FROM Customers  
  FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
  ORDER BY Customers.CustomerName;
* SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
  FROM Customers A, Customers B  
  WHERE A.CustomerID <> B.CustomerID  
  AND A.City = B.City  
  ORDER BY A.City;

SQL statement returns the cities (only distinct values) from both the "Customers" and the "Suppliers" table:

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

SQL statement returns the cities (duplicate values also) from both the "Customers" and the "Suppliers" table:

* SELECT City FROM Customers  
  UNION ALL  
  SELECT City FROM Suppliers  
  ORDER BY City;

SQL statement lists all customers and suppliers:

* SELECT 'Customer' AS Type, ContactName, City, Country  
  FROM Customers  
  UNION  
  SELECT 'Supplier', ContactName, City, Country  
  FROM Suppliers;
* SELECT COUNT(CustomerID), Country  
  FROM Customers  
  GROUP BY Country  
  ORDER BY COUNT(CustomerID) DESC;
* SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders  
  LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
  GROUP BY ShipperName;
* SELECT COUNT(CustomerID), Country  
  FROM Customers  
  GROUP BY Country  
  HAVING COUNT(CustomerID) > 5  
  ORDER BY COUNT(CustomerID) DESC;
* SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
  FROM Orders  
  INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
  WHERE LastName = 'Davolio' OR LastName = 'Fuller'  
  GROUP BY LastName  
  HAVING COUNT(Orders.OrderID) > 25;
* SELECT SupplierName  
  FROM Suppliers  
  WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);
* SELECT ProductName  
  FROM Products  
  WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity > 99);

The following SQL statement creates a backup copy of Customers:

* SELECT \* INTO CustomersBackup2017  
  FROM Customers;

The following SQL statement uses the IN clause to copy the table into a new table in another database:

* SELECT \* INTO CustomersBackup2017 IN 'Backup.mdb'  
  FROM Customers;

The following SQL statement copies only a few columns into a new table:

* SELECT CustomerName, ContactName INTO CustomersBackup2017  
  FROM Customers;

The following SQL statement copies only the German customers into a new table:

* SELECT \* INTO CustomersGermany  
  FROM Customers  
  WHERE Country = 'Germany';

The following SQL statement copies data from more than one table into a new table:

* SELECT Customers.CustomerName, Orders.OrderID  
  INTO CustomersOrderBackup2017  
  FROM Customers  
  LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

The following SQL statement copies "Suppliers" into "Customers" (the columns that are not filled with data, will contain NULL):

* INSERT INTO Customers (CustomerName, City, Country)  
  SELECT SupplierName, City, Country FROM Suppliers;

The following SQL statement copies "Suppliers" into "Customers" (fill all columns):

* INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
  SELECT SupplierName, ContactName, Address, City, PostalCode, Country FROM Suppliers;

The following SQL statement copies only the German suppliers into "Customers":

* INSERT INTO Customers (CustomerName, City, Country)  
  SELECT SupplierName, City, Country FROM Suppliers  
  WHERE Country='Germany';

The following SQL goes through conditions and returns a value when the first condition is met:

* SELECT OrderID, Quantity,  
  CASE  
      WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
      WHEN Quantity = 30 THEN 'The quantity is 30'  
      ELSE 'The quantity is under 30'  
  END AS QuantityText  
  FROM OrderDetails;

The following SQL will order the customers by City. However, if City is NULL, then order by Country:

* SELECT CustomerName, City, Country  
  FROM Customers  
  ORDER BY  
  (CASE  
      WHEN City IS NULL THEN Country  
      ELSE City  
  END);

Suppose that the "UnitsOnOrder" column is optional, and may contain NULL values.

* SELECT ProductName, UnitPrice \* (UnitsInStock + UnitsOnOrder)  
  FROM Products;

In the example above, if any of the "UnitsOnOrder" values are NULL, the result will be NULL.

**Solutions**

* **MySQL**

The MySQL [IFNULL()](https://www.w3schools.com/sql/func_mysql_ifnull.asp) function lets you return an alternative value if an expression is NULL:

* SELECT ProductName, UnitPrice \* (UnitsInStock + IFNULL(UnitsOnOrder, 0))  
  FROM Products;

or we can use the [COALESCE()](https://www.w3schools.com/sql/func_mysql_coalesce.asp) function, like this:

* SELECT ProductName, UnitPrice \* (UnitsInStock + COALESCE(UnitsOnOrder, 0))  
  FROM Products;
* **SQL Server**

The SQL Server [ISNULL()](https://www.w3schools.com/sql/func_sqlserver_isnull.asp) function lets you return an alternative value when an expression is NULL:

* SELECT ProductName, UnitPrice \* (UnitsInStock + ISNULL(UnitsOnOrder, 0))  
  FROM Products;

The following SQL statement creates a stored procedure that selects Customers from a particular City with a particular PostalCode from the "Customers" table:

* Example
* CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode nvarchar(10)  
  AS  
  SELECT \* FROM Customers WHERE City = @City AND PostalCode = @PostalCode  
  GO;
* Execute the stored procedure above as follows:
* Example
* EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';

--Select all:  
SELECT \* FROM Customers;

/\*SELECT \* FROM Customers;  
SELECT \* FROM Products;  
SELECT \* FROM Orders;  
SELECT \* FROM Categories;\*/  
SELECT \* FROM Suppliers;