**SQL**

SQL is a standard language for storing, manipulating and retrieving data in databases.

SQL in: **MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres**, and other database systems.

**SQL is a standard language for accessing and manipulating databases.**

## What is SQL?

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a **standard** of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

## What Can SQL do?

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

## Using SQL in Your Web Site

To build a web site that shows data from a database, you will **need**:

* An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
* To use a server-side scripting language, like PHP or ASP
* To use SQL to get the data you want
* To use HTML / CSS to style the page

## RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Database🡪Table🡪Field🡪DataType

## Customers table



## Keep in Mind That...

* SQL keywords are NOT case sensitive: select is the same as SELECT

We will write all SQL keywords in upper-case.

## Some of The Most Important SQL Commands

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

Example: SELECT \* FROM Customers;

Creating Database

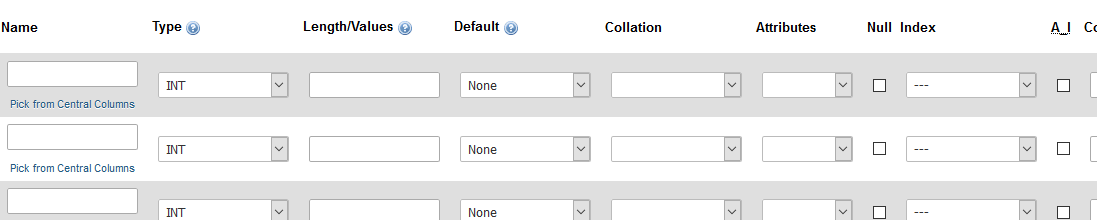
1. localhost/phpmyadmin
2. New
3. Write Database Name (InsDemo) 🡪 Create

## Create Table

1. Write any name (Customer)
2. No. of Columns – enter any value (7) 🡪 Go



1. After that we will see



1. Write the Column name in each box

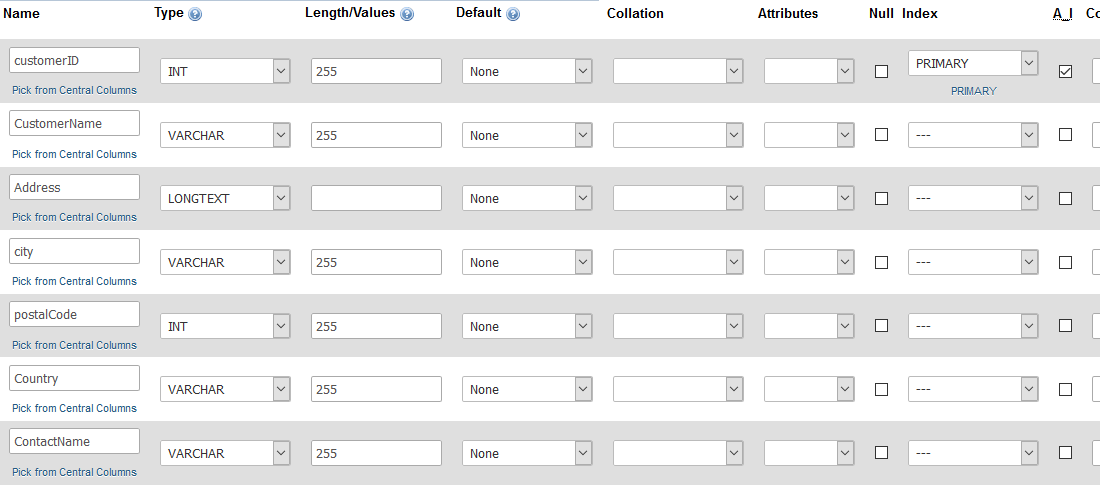
Example –

Name 🡪 CustomerID,

Type🡪 int

Length🡪 255 (Max)

A\_I 🡪 Checked 🡪Go



1. Click on Go button.

SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

### INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways.

The first way specifies both the column names and the values to be inserted:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

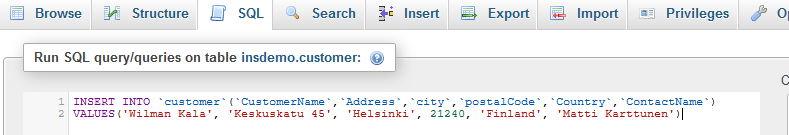
### Example

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

**Note: For Execute this Go to SQL Section in PhpMyAdmin**

Example 2 –

INSERT INTO `customer`(`CustomerName`,`Address`,`city`,`postalCode`,`Country`,`ContactName`)

VALUES('Wilman Kala', 'Keskuskatu 45', 'Helsinki', 21240, 'Finland', 'Matti Karttunen')

Then Click on Go, If success



SQL SELECT Statement

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

Syntax:

SELECT column1, column2, ... FROM table\_name;

SELECT \* FROM table\_name;

Example:



SELECT CustomerName, City FROM Customers;

SELECT \* FROM Customers;

## The SQL WHERE Clause

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

**Note:** The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!

SELECT \* FROM Customers WHERE Country='Mexico';

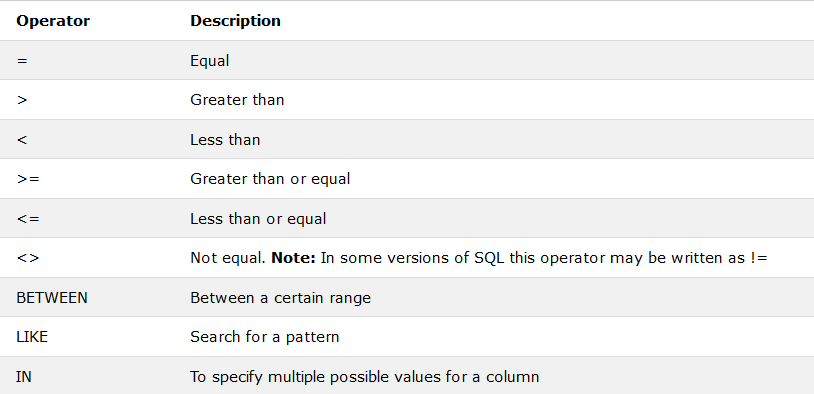
## Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

### Example

SELECT \* FROM Customers WHERE CustomerID=1;



## The SQL SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

### SELECT DISTINCT Syntax

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

SELECT DISTINCT Country FROM Customers;

SELECT Count(\*) AS DistinctCountries  
FROM (SELECT DISTINCT Country FROM Customers);

## The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

### AND Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

### SELECT \* FROM Customers WHERE Country='Germany' AND City='Berlin';

### OR Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

SELECT \* FROM Customers  
WHERE City='Berlin' OR City='München';

## The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending(**ASC**) order by default. To sort the records in descending order, use the **DESC** keyword.

### ORDER BY Syntax

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

SELECT \* FROM Customers ORDER BY Country;

## What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

**Note:** A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

## How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

### IS NULL Syntax

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

### IS NOT NULL Syntax

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

### Example

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL;

## The SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

### UPDATE Syntax

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

### Example

### UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

## The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

### DELETE Syntax

DELETE FROM table\_name WHERE condition;

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

## Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table\_name;

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

### Example

DELETE FROM Customers;

## The SQL SELECT TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact on performance.

**Note:** Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

**SQL Server / MS Access Syntax:**

SELECT TOP *number*|*percent* *column\_name(s)*  
FROM *table\_name*WHERE *condition*;

**MySQL Syntax:**

SELECT *column\_name(s)*  
FROM *table\_name*WHERE *condition*  
LIMIT *number*;

**Oracle Syntax:**

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE ROWNUM <= *number*;

SELECT TOP 3 \* FROM Customers;

SELECT \* FROM Customers  
LIMIT 3;

## SQL TOP PERCENT Example

The following SQL statement selects the first 50% of the records from the "Customers" table:

### Example

SELECT TOP 50 PERCENT \* FROM Customers;

SELECT TOP 3 \* FROM Customers WHERE Country='Germany';

## The SQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

### MIN() Syntax

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

### MAX() Syntax

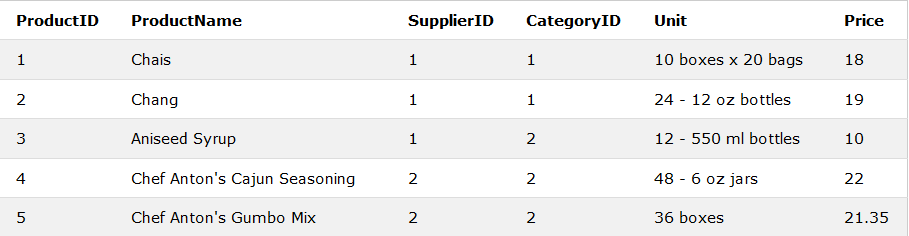
SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

## MIN() Example

The following SQL statement finds the price of the cheapest product:

### Example

SELECT MIN(Price) AS SmallestPrice  
FROM Products;



## MAX() Example

The following SQL statement finds the price of the most expensive product:

### Example

SELECT MAX(Price) AS LargestPrice  
FROM Products;

## The SQL COUNT(), AVG() and SUM() Functions

The COUNT() function returns the number of rows that matches a specified criteria.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

### COUNT() Syntax

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

### AVG() Syntax

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

### SUM() Syntax

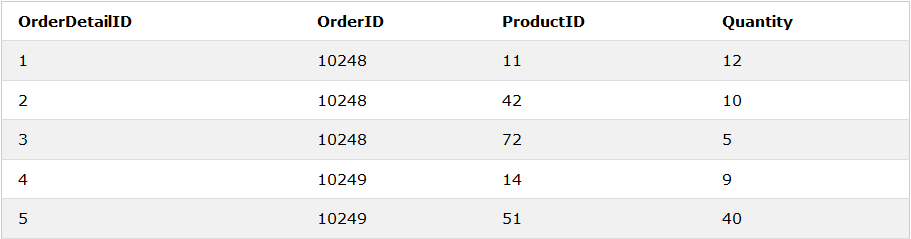
SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

### Example

SELECT COUNT(ProductID) FROM Products; **Note:** NULL values are not counted.

SELECT AVG(Price) FROM Products; **Note:** NULL values are ignored.

SELECT SUM(Quantity) FROM OrderDetails; **Note:** NULL values are ignored.



## The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* % - The percent sign represents zero, one, or multiple characters
* \_ - The underscore represents a single character

**Note:** MS Access uses an asterisk (\*) instead of the percent sign (%), and a question mark (?) instead of the underscore (\_).

The percent sign and the underscore can also be used in combinations!

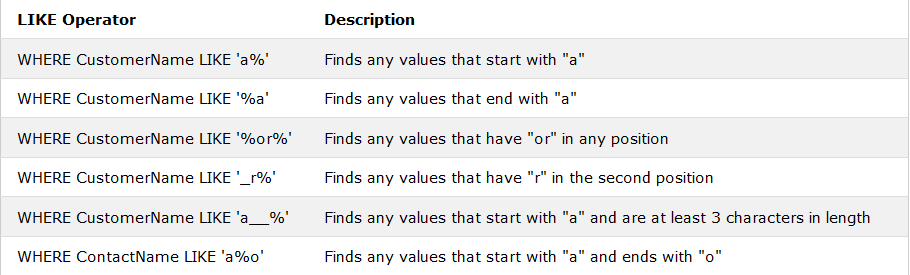
### LIKE Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

**Tip:** You can also combine any number of conditions using AND or OR operators.

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |
| --- |
|  |





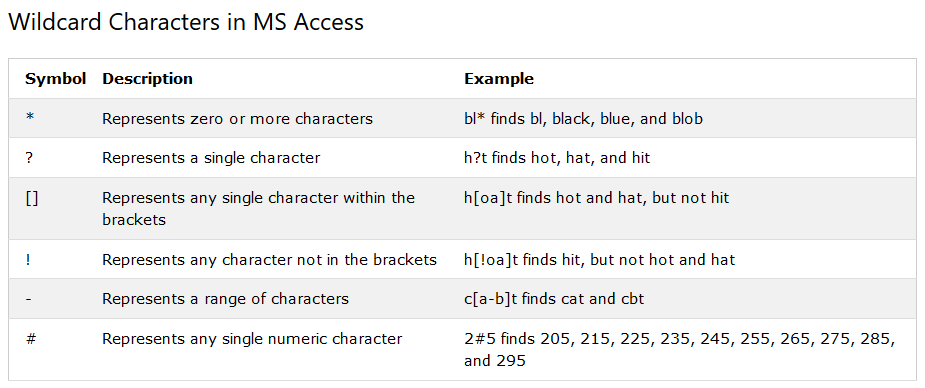
### Example

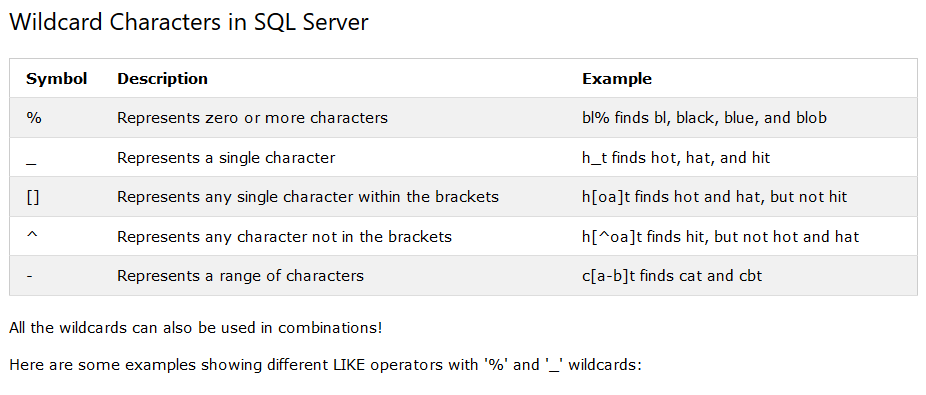
SELECT \* FROM Customers WHERE CustomerName LIKE 'a%';

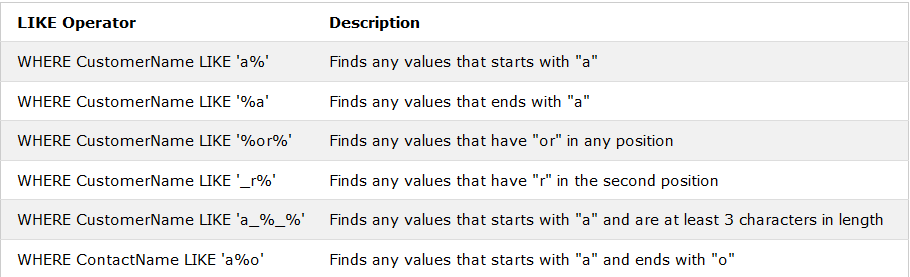
## SQL Wildcard Characters

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the SQL LIKE operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.







## Using the % Wildcard

The following SQL statement selects all customers with a City starting with "ber":

### Example

SELECT \* FROM Customers  
WHERE City LIKE 'ber%';

The following SQL statement selects all customers with a City containing the pattern "es":

### Example

SELECT \* FROM Customers  
WHERE City LIKE '%es%';

## Using the \_ Wildcard

The following SQL statement selects all customers with a City starting with any character, followed by "ondon":

### Example

SELECT \* FROM Customers  
WHERE City LIKE '\_ondon';

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

### Example

SELECT \* FROM Customers  
WHERE City LIKE 'L\_n\_on';

## Using the [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

### Example

SELECT \* FROM Customers  
WHERE City LIKE '[bsp]%';

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

### Example

SELECT \* FROM Customers  
WHERE City LIKE '[a-c]%';

## Using the [!charlist] Wildcard

The two following SQL statements select all customers with a City NOT starting with "b", "s", or "p":

### Example

SELECT \* FROM Customers  
WHERE City LIKE '[!bsp]%';

Or:

### Example

SELECT \* FROM Customers  
WHERE City NOT LIKE '[bsp]%';

## The SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

### IN Syntax

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name* IN (*SELECT STATEMENT*);

## IN Operator Examples

The following SQL statement selects all customers that are located in "Germany", "France" and "UK":

### Example

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

The following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":

### Example

SELECT \* FROM Customers  
WHERE Country NOT IN ('Germany', 'France', 'UK');

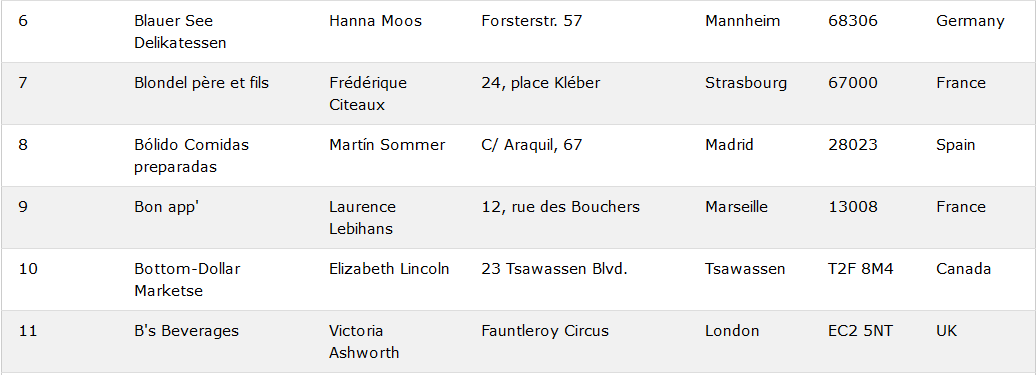
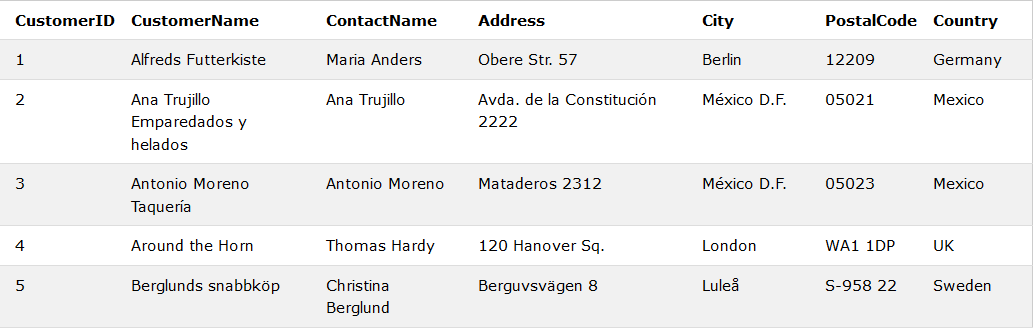
The following SQL statement selects all customers that are from the same countries as the suppliers:

### Example

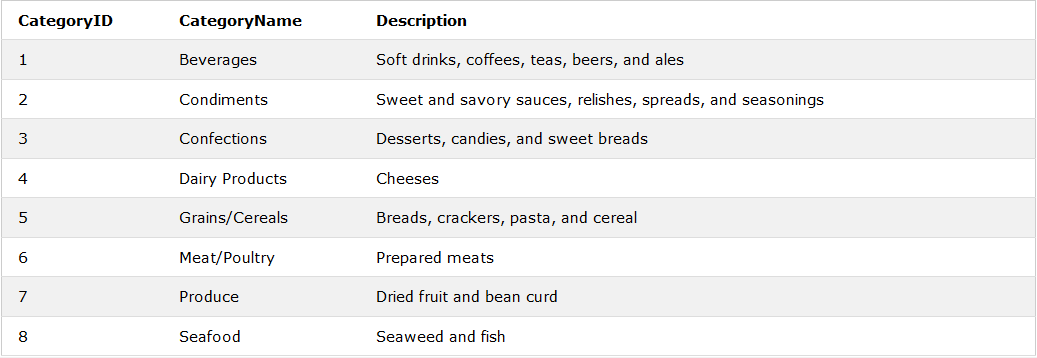
SELECT \* FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);

Table (Have to create first)

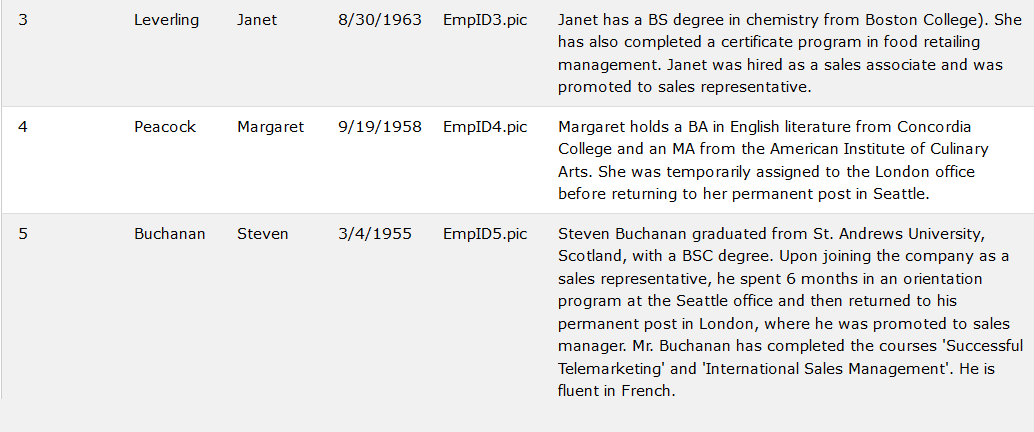
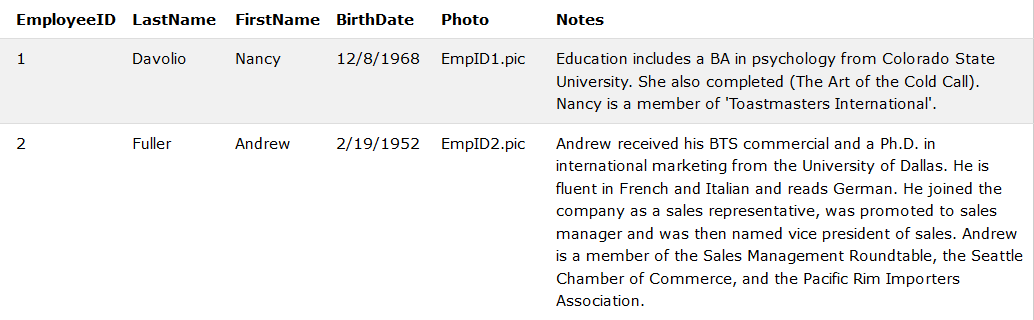
## Customers

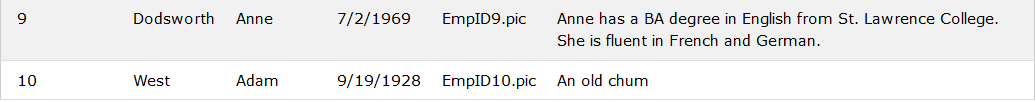
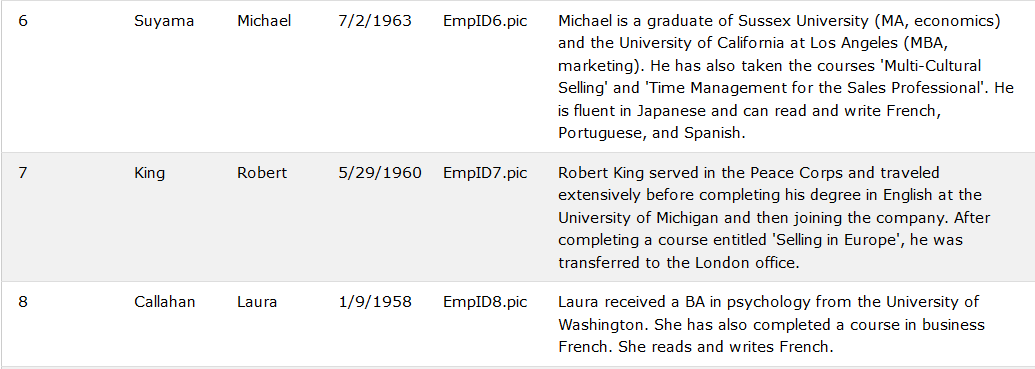


## Categories

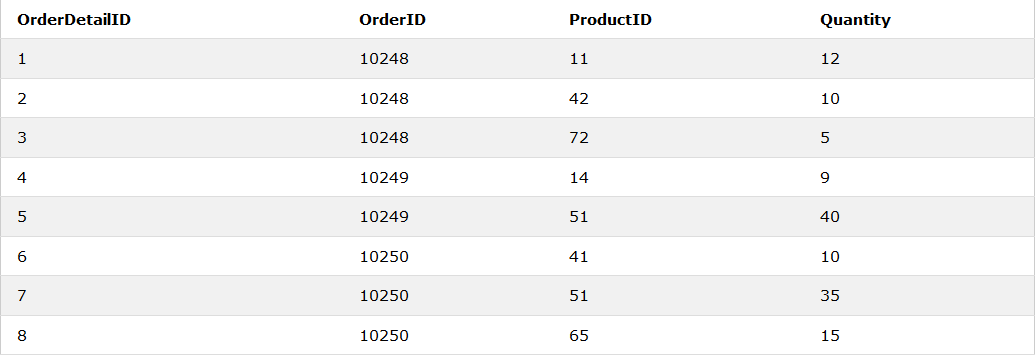


## Employees

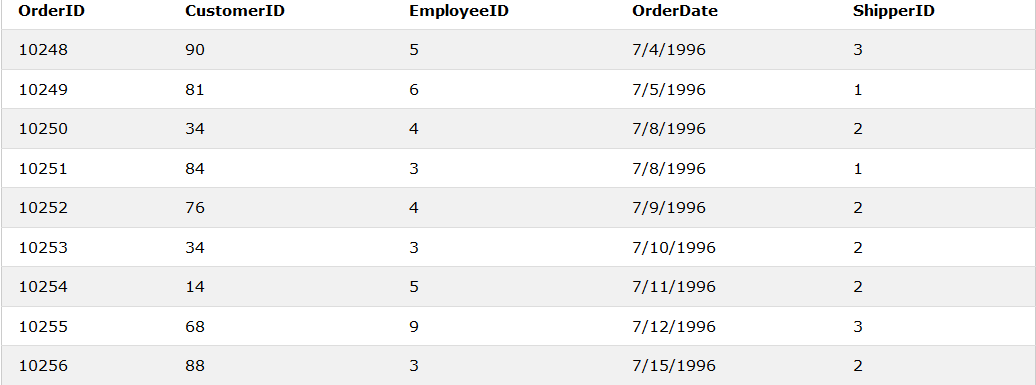




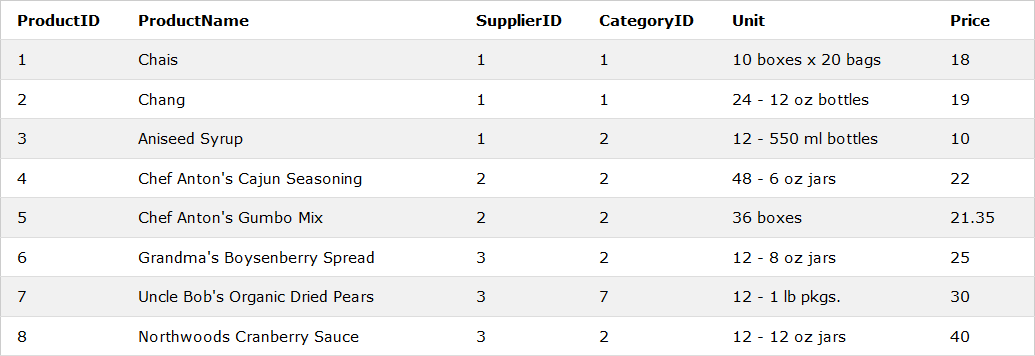
## OrderDetails



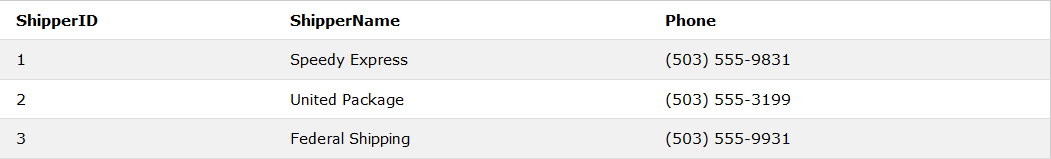
## Orders



## Products



## Shippers



## Suppliers

