



5602201

3. SQL Advanced

CHUTIPORN ANUTARIYA



Warm up!

WHAT DID WE LEARN LAST TIME....



SQL: Basics

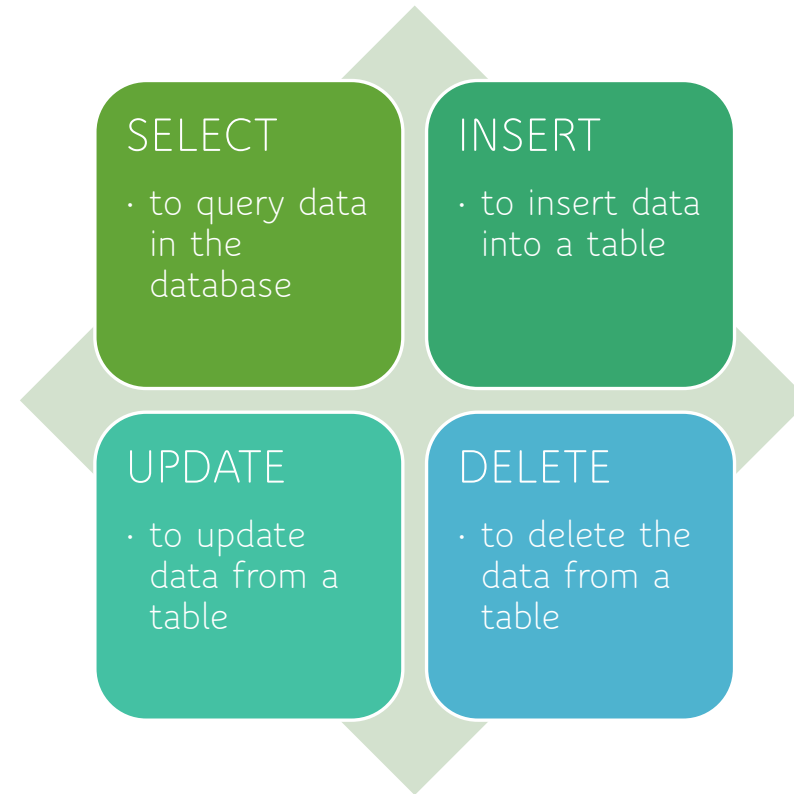
STRUCTURED QUERY LANGUAGE

SQL: Structured Query Language

- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
- SQL standard has two main components:
 - Data Definition Language (DDL): to define the structure of the database and control data access.
 - Data Manipulation Language (DML): for data retrieval and updating.

Data Manipulation Language (DML)

Data Manipulation
Language (DML):
for data retrieval
and updating.



Demo Database and SQL Playground

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all



Your Database:

Tablename	Records
<u>Customers</u>	91
<u>Categories</u>	8
<u>Employees</u>	10
<u>OrderDetails</u>	518
<u>Orders</u>	196
<u>Products</u>	77
<u>Shippers</u>	3
<u>Suppliers</u>	29

SELECT Syntax

```
SELECT column1, column2, ...  
FROM table_name;
```

SQL SELECT Statement

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

- SELECT
- SELECT DISTINCT
- WHERE
- ORDER BY
- AND, OR, NOT
- LIMIT



LET'S

START

MIN(), MAX() Functions

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

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

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```



MIN(), MAX() Functions

Finds the price of the cheapest product.

Finds the price of the most expensive product.

Example

```
SELECT MIN(Price) AS SmallestPrice  
FROM Products;
```

Example

```
SELECT MAX(Price) AS LargestPrice  
FROM Products;
```

COUNT(), AVG() and SUM() Functions

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

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

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

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

LIKE Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE columnN LIKE pattern;
```

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 sign (_) represents one, single character

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

LIKE Examples



LIKE Examples

Select all customers with a
CustomerName starting with "a"

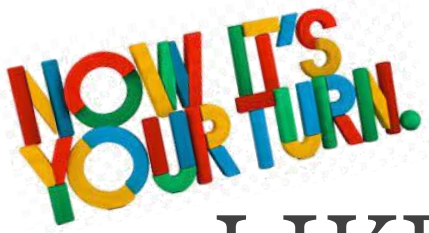
Select all customers with a
CustomerName ending with "a"

Example

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

Example

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

LIKE Operator

1. Select all customers with a CustomerName that have "or" in any position.
2. Select all customers with a CustomerName that have "r" in the second position.
3. Select all customers with a CustomerName that starts with "a" and are at least 3 characters in length.
4. Selects all customers with a ContactName that starts with "a" and ends with "o".
5. Select all customers with a CustomerName that does NOT start with "a".
Hint: Use NOT.

ALIAS

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Alias Column Syntax

```
SELECT column_name AS alias_name  
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

Alias for Columns Examples

Example

```
SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;
```

Example

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;
```

```
SELECT CustomerName, CONCAT(Address, ', ', PostalCode, ', ', City, ', ', Country) AS Address  
FROM Customers;
```

JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the "Orders" table.

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, look at a selection from the "Customers" table.

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

JOIN Example

Orders

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Customers

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

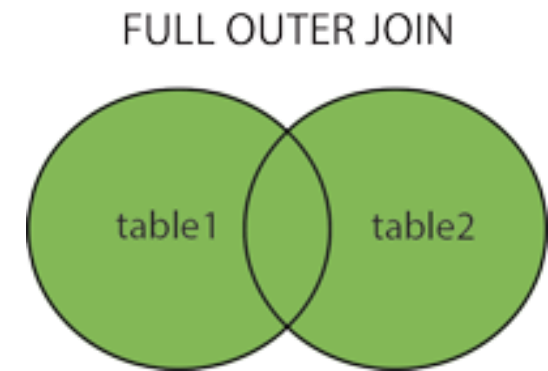
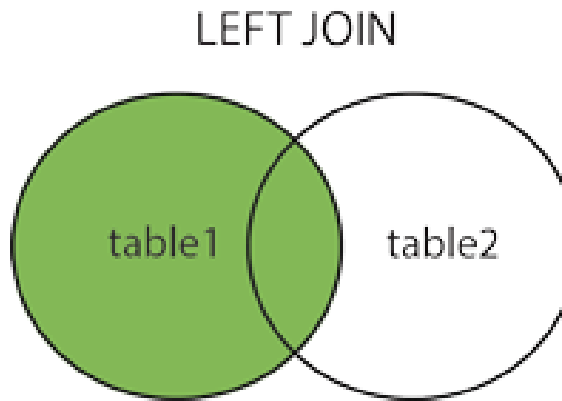
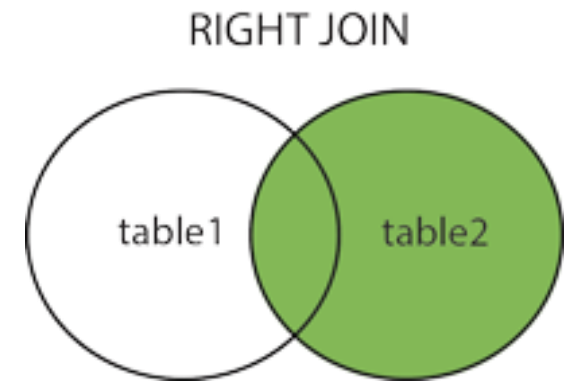
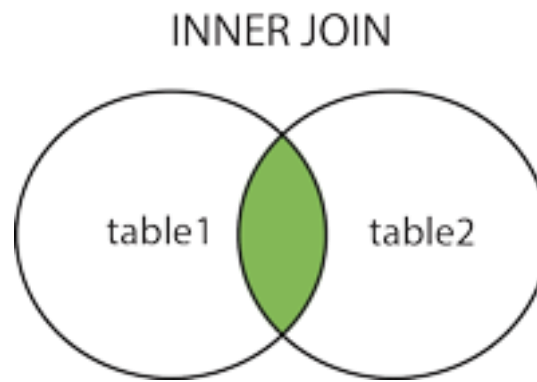
JOIN Example

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Different Types of SQL JOINS

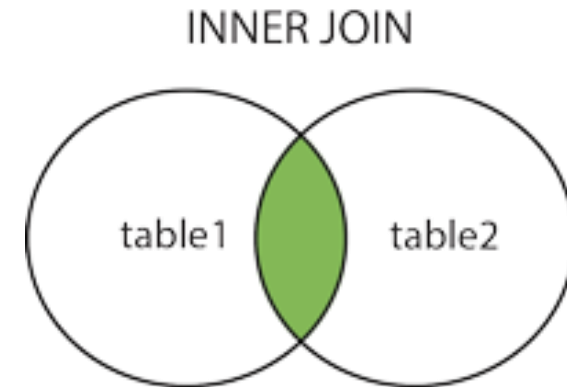


- **(INNER) JOIN** : Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN** : Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN** : Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN** : Returns all records when there is a match in either left or right table

INNER JOIN

The **INNER JOIN** keyword selects records that have matching values in both tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```





INNER JOIN Example

Select all orders with customer information:

Example

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Select all orders with customer and shipper information:

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

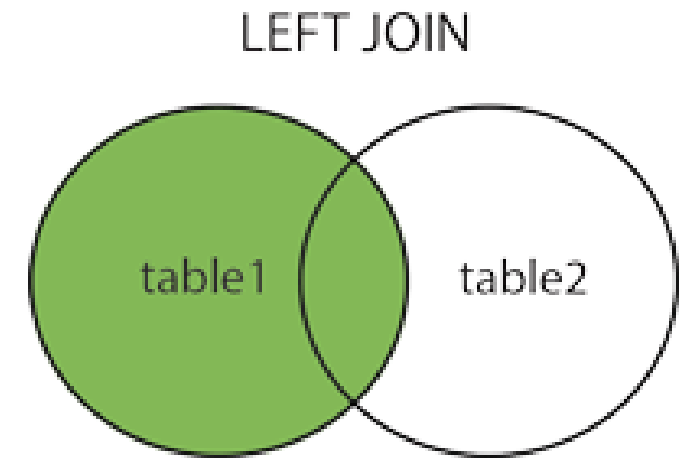
Note: The **INNER JOIN** keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the "Orders" table that do not have matches in "Customers", these orders will not be shown!

LEFT JOIN

The **LEFT JOIN** keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```



Note: In some databases LEFT JOIN is called LEFT OUTER JOIN.



LEFT JOIN Example

Select all
customers, and
any orders they
might have:

Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Note: The **LEFT JOIN** keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

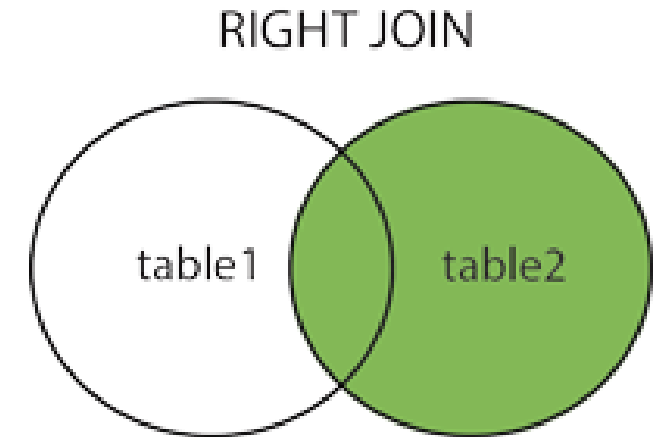
RIGHT JOIN

The **RIGHT JOIN** keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.





RIGHT JOIN Example

Return all
employees, and
any orders they
might have placed:

Example

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;
```

Note: The **RIGHT JOIN** keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders).

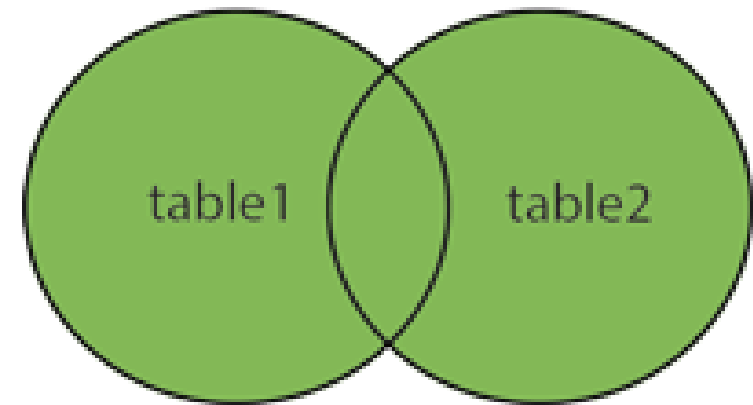
FULL OUTER JOIN Syntax

FULL OUTER JOIN

The **FULL OUTER JOIN** keyword returns all records when there is a match in left (table1) or right (table2) table records.

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

FULL OUTER JOIN



Note: **FULL OUTER JOIN** can potentially return very large result-sets!



FULL OUTER JOIN Example

Select all customers, and all orders:

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

Note: The **FULL OUTER JOIN** keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

SELF JOIN

A self join is a regular join, but the table is joined with itself.

Self Join Syntax

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

T1 and *T2* are different table aliases for the same table.

SELF JOIN Example



Match customers that are from the same city

```
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;
```


Questions? |

