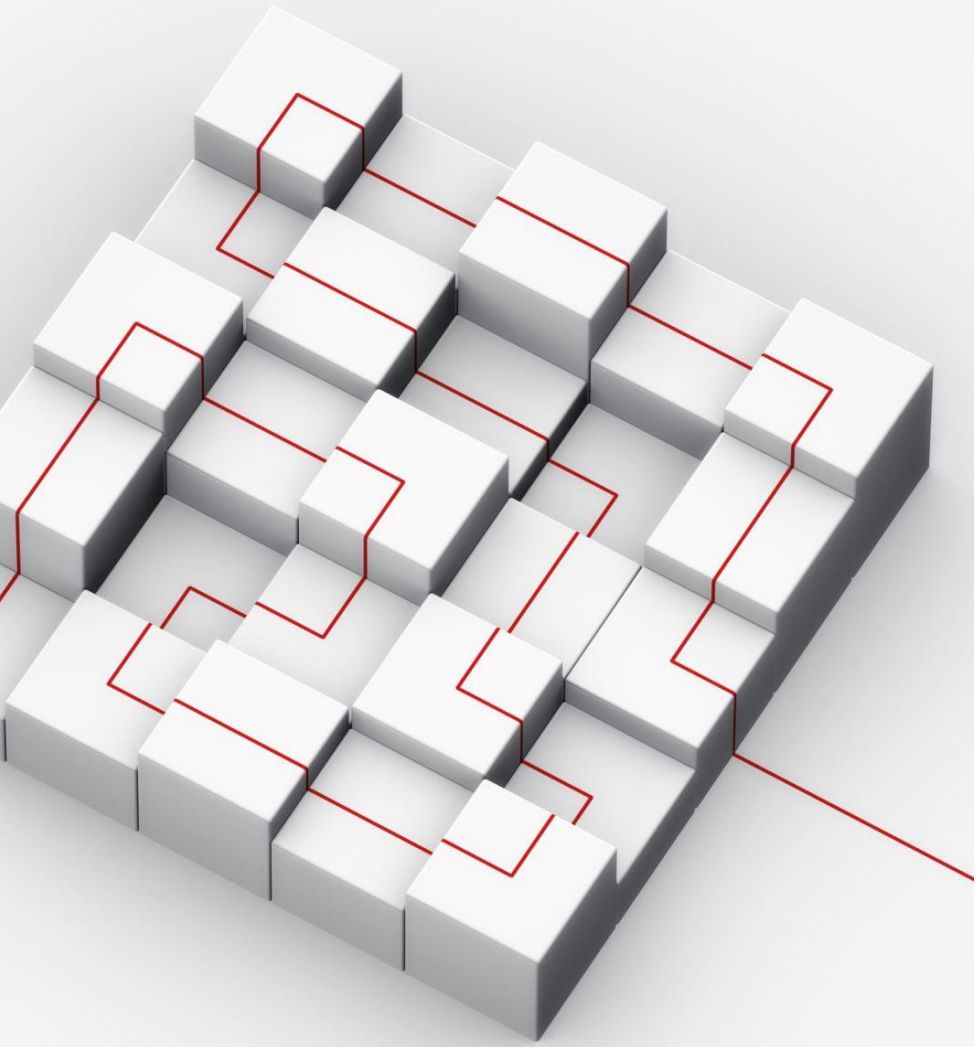
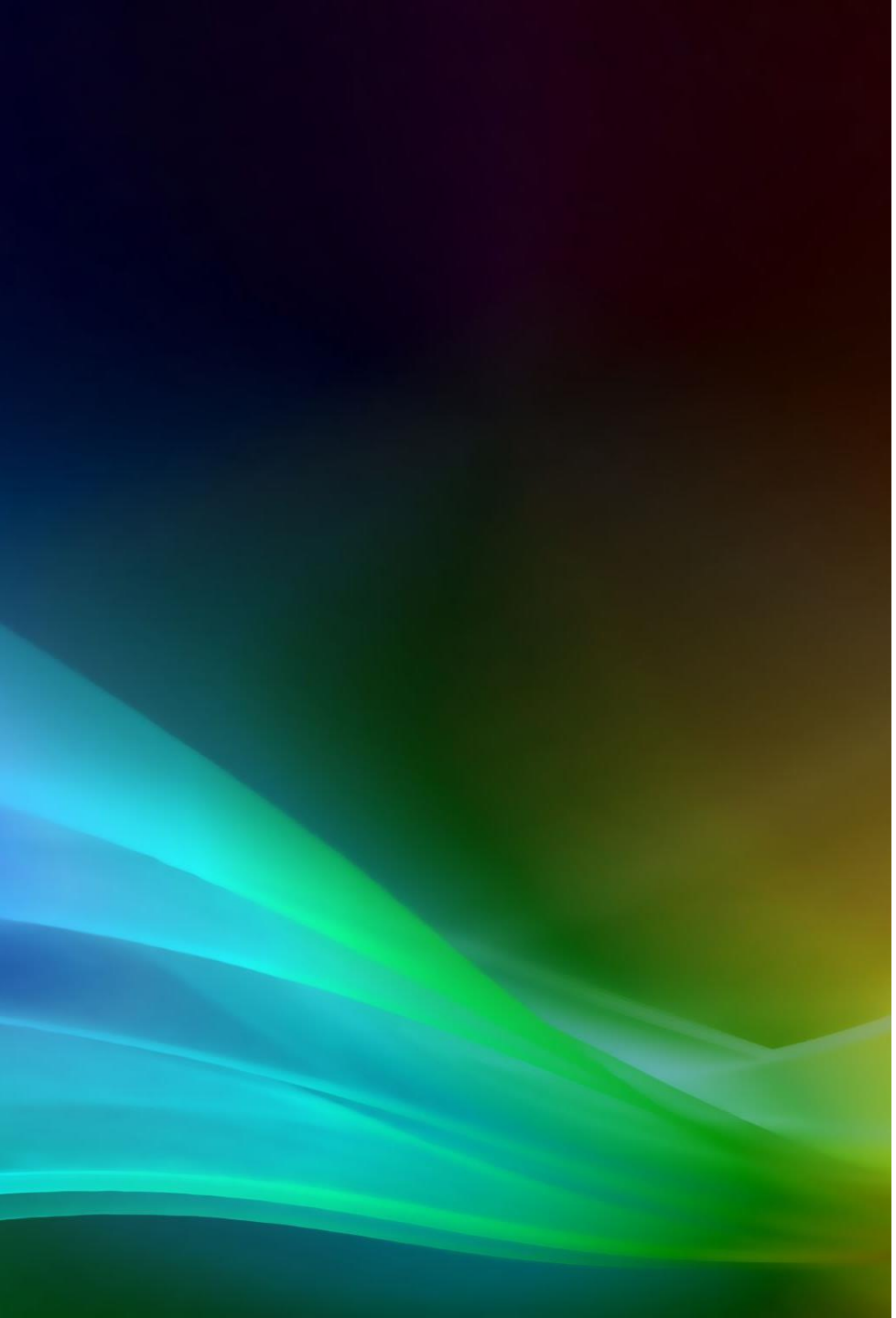


# 5602201

# 4. Creating Databases and Tables

CHUTIPORN ANUTARIYA



# Recap

WHAT DID WE LEARN LAST TIME....

# SQL Advanced: JOIN

# JOIN

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

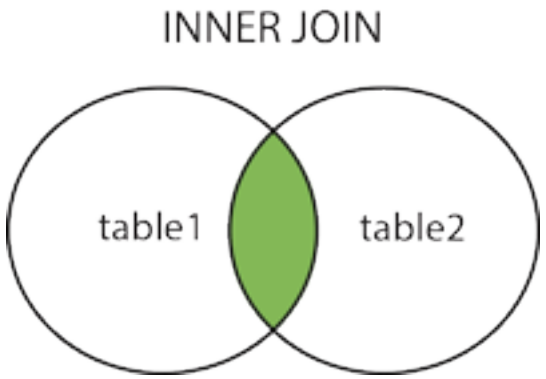
Let's look at a selection from the "Orders" table.

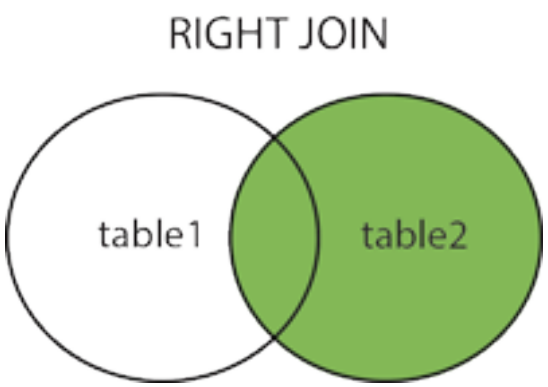
| OrderID | CustomerID | OrderDate |
|---|---|---|
| 10308 | 2 | 1996-09-18 |
| 10309 | 37 | 1996-09-19 |
| 10310 | 77 | 1996-09-20 |

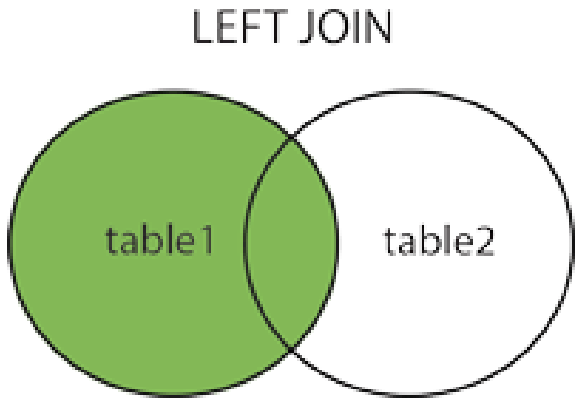
Then, look at a selection from the "Customers" table.

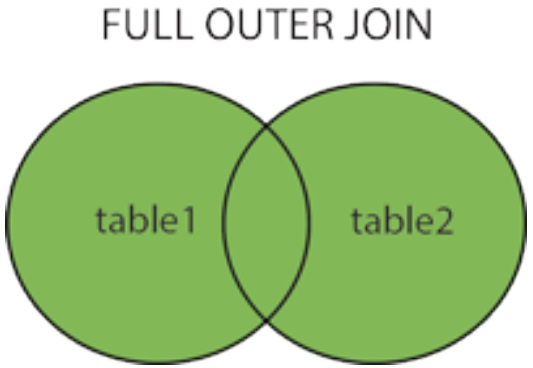
| CustomerID | CustomerName | ContactName | Country |
|---|---|---|---|
| 1 | Alfreds Futterkiste | Maria Anders | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mexico |

# Different Types of SQL JOINs



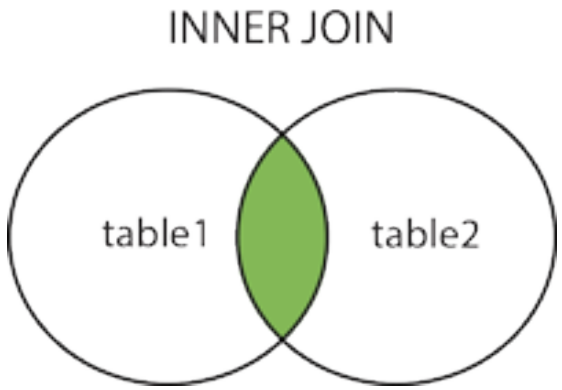




- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (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:

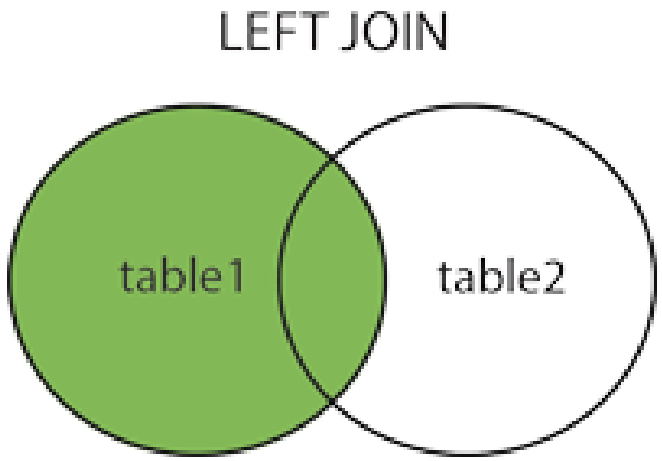
Example

```
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:

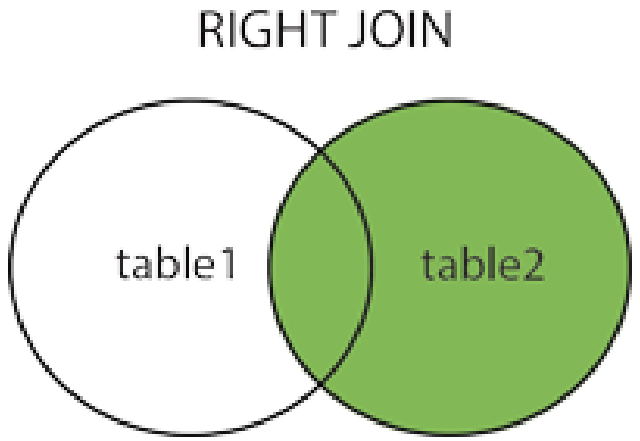
Example

```
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;
```

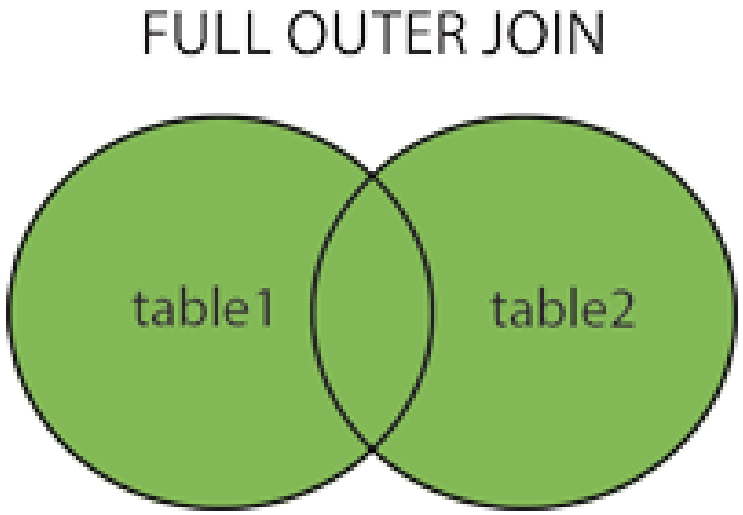
**Note:** The `RIGHT JOIN` keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders).

# FULL OUTER JOIN

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

FULL OUTER JOIN Syntax

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




**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

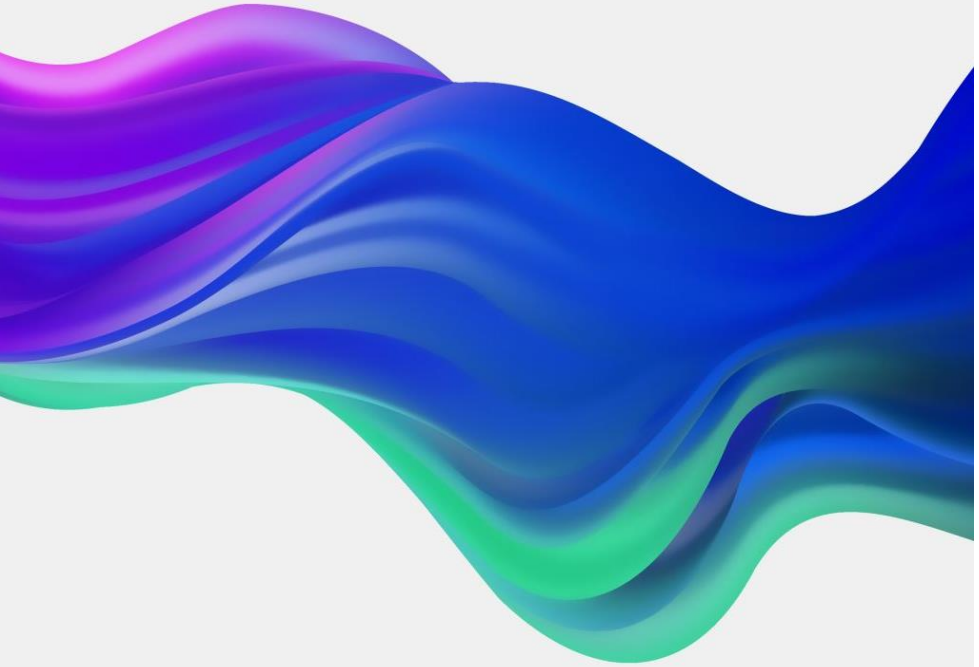

Match customers that are from the same city

```
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;
```


# JOIN Three Tables

1 Write an SQL statement selects all orders with customer and shipper information:

| OrderID | CustomerName | ShipperName |
|---|---|---|
| 10248 | Wilman Kala | Federal Shipping |
| 10249 | Tradição Hipermercados | Speedy Express |
| 10250 | Hanari Carnes | United Package |
| 10251 | Victuailles en stock | Speedy Express |
| 10252 | Suprêmes délices | United Package |
| 10253 | Hanari Carnes | United Package |
| 10254 | Chop-suey Chinese | United Package |
| 10255 | Richter Supermarkt | Federal Shipping |
| 10256 | Wellington Importadora | United Package |
| 10257 | HILARIÓN-Abastos | Federal Shipping |
| 10258 | Ernst Handel | Speedy Express |
| 10259 | Centro comercial Moctezuma | Federal Shipping |



# GROUP BY

# GROUP BY

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

## GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

# GROUP BY Examples


2

Lists the number of customers in each country:

```
SELECT Country,
_____________ as NumberOfCustomers
FROM Customers
GROUP BY Country;
```

| Country | NumberOfCustomers |
|---|---|
| Argentina | 3 |
| Austria | 2 |
| Belgium | 2 |
| Brazil | 9 |
| Canada | 3 |
| Denmark | 2 |
| Finland | 2 |
| France | 11 |
| Germany | 11 |
| Ireland | 1 |

# GROUP BY Examples


3

Lists the number of customers in each country, sorted high to low:

```
SELECT Country,
_____________ as NumberOfCustomers
FROM Customers
GROUP BY Country
_____________;
```

| Country | NumberOfCustomers |
|---|---|
| USA | 13 |
| Germany | 11 |
| France | 11 |
| Brazil | 9 |
| UK | 7 |

# GROUP BY with JOIN Examples

Lists the number of orders sent by each shipper:


| ShipperName | NumberOfOrders |
|---|---|
| Federal Shipping | 68 |
| Speedy Express | 54 |
| United Package | 74 |

# GROUP BY with JOIN Examples

Lists the number of orders sent by each shipper:

| ShipperName | NumberOfOrders |
|---|---|
| Federal Shipping | 68 |
| Speedy Express | 54 |
| United Package | 74 |


```
SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName;
```

# HAVING

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

# HAVING Examples


4

Lists the number of customers in each country. Only include countries with more than 5 customers:

| Country | NumberOfCustomers |
|---|---|
| Brazil | 9 |
| France | 11 |
| Germany | 11 |
| UK | 7 |
| USA | 13 |

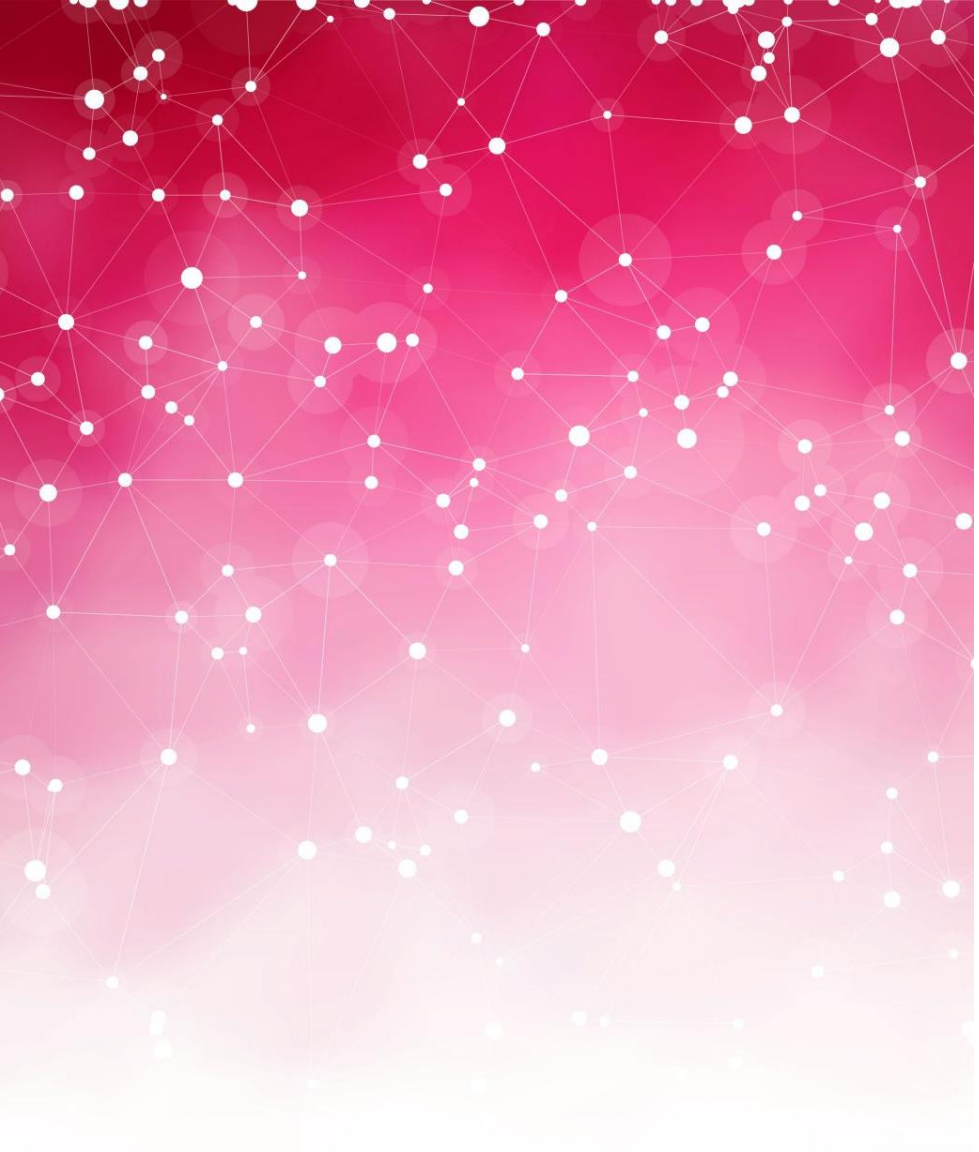
```
SELECT Country,
_____________ as NumberOfCustomers
FROM Customers
GROUP BY Country
HAVING _____________;
```

# HAVING Examples


**5** Lists the number of customers in each country, sorted high to low. Only include countries with more than 5 customers:

| Country | NumberOfCustomers |
|---|---|
| USA | 13 |
| Germany | 11 |
| France | 11 |
| Brazil | 9 |
| UK | 7 |

```
SELECT Country,
_____________ as NumberOfCustomers
FROM Customers
GROUP BY Country
HAVING _____________
____________________;
```



# SQL INSERT, UPDATE, DELETE

# INSERT INTO

1. Specify both the column names and the values to be inserted.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. 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.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

# INSERT INTO Example-1


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

| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
|---|---|---|---|---|---|---|
| 89 | White Clover Markets | Karl Jablonski | 305 - 14th Ave. S. Suite 3B | Seattle | 98128 | USA |
| 90 | Wilman Kala | Matti Karttunen | Keskuskatu 45 | Helsinki | 21240 | Finland |
| 91 | Wolski | Zbyszek | ul. Filtrowa 68 | Walla | 01-012 | Poland |
| 92 | Cardinal | Tom B. Erichsen | Skagen 21 | Stavanger | 4006 | Norway |

# INSERT INTO Example-2


```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
|---|---|---|---|---|---|---|
| 89 | White Clover Markets | Karl Jablonski | 305 - 14th Ave. S. Suite 3B | Seattle | 98128 | USA |
| 90 | Wilman Kala | Matti Karttunen | Keskuskatu 45 | Helsinki | 21240 | Finland |
| 91 | Wolski | Zbyszek | ul. Filtrowa 68 | Walla | 01-012 | Poland |
| 92 | Cardinal | null | null | Stavanger | null | Norway |

# UPDATE

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

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!

## UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

# UPDATE Example


## Example

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

## Example

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

## Example

```
UPDATE Customers
SET ContactName='Juan';
```

# DELETE

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

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 Syntax

```
DELETE FROM table_name WHERE condition;
```

# DELETE Example


Example

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

Example

```
DELETE FROM Customers;
```


Questions?
HSM
ELON