SQL Основи

* **SELECT \* FROM ...**

**SELECT** се използва за избор на данни от база данни. Резултата който се връща от селекцията се записва в таблицата с резултати или така нареченият result-set.  
 **Пример : SELECT \* FROM Customers;  
 SELECT -** обявяваме че ще селектираме от дадена таблица.  
 \* - звездата означава че ще селектираме всички колони от таблицата ( **CustomerID , CustomerName , ContactName, Adress, City, PostCode, Country**)  
 **FROM** - след тази дума оказваме от коя таблица ще селектираме от дясно на думата. В примера таблицата е **Customer**



**Резултата :**

***CustomerID CustomerName ContactName Address City PostalCode Country***

***1 Alfreds Futterkiste Maria Anders Obere Str. 57 Berlin 12209 Germany***

***2 Ana Trujillo Emparedados y helados Ana Trujillo Avda. de la Constitución 2222 México D.F. 05021 Mexico ...*  
 Ако не желаем в резултата да се показват всички колони от селектираната таблица трябва ( \* ) да изброим точно резултата от кои колони ни е нужен.  
  
 Пример :   
 SELECT CustomerID, CustomerName, ContactName FROM Customers;  
  
 Резултат :   
 *CustomerID CustomerName ContactName***

***1 Alfreds Futterkiste Maria Anders***

***2 Ana Trujillo Emparedados y helados Ana Trujillo***

[***https://www.w3schools.com/sql/trysql.asp?filename=trysql\_select\_all***](https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all)

* ***SELECT DISTINCT***

Използва се да се върнат само уникалните записи в таблицата, тъйкато в много случаи в таблицата има записи които се повтарят като имена, стойности и т.н.

**Пример : SELECT DISTINCT Country FROM Customers;  
  
 Резултата:** Ще се върнат всички държави от които са клиентите в таблицата **Customers ,** тъй като може да имаме 200 клиента които са от дадена държава. Това ни позволява да разберем от кои дъжави са нашите клиенти.

* ***WHERE клауза...***

**WHERE** думата се използва за да филтрираме резултата по някъкво условие и да вземем данните които отговарят на него.  
 **Пример: SELECT \* FROM Customer  
 WHERE Country='Mexico';** Казваме че ще селектирам от таблицата **Customers** където колоната **Country** на даден резултат има стойнос **Mexico**. Така ще извадим всички **Customers** където отговарят на това условие.



Също така в **WHERE** клаузата може да използваме повече от едно условия. Това става с други специални думи като **AND , OR , NOT.**

**Пример : SELECT \* FROM Customers**

**WHERE City='Berlin' OR City='München';**

Резултат :

CustomerID CustomerName ContactName Address City PostalCode Country

1 Alfreds Futterkiste Maria Anders Obere Str. 57 **Berlin** 12209 Germany

25 Frankenversand Peter Franken Berliner Platz 43 **München** 80805 Germany

Така взимаме записите от таблицата **Customers** където **City** стойноста е равна на **Berlin** или **München.**[**https://www.w3schools.com/sql/sql\_and\_or.asp**](https://www.w3schools.com/sql/sql_and_or.asp)

* **ORDER BY...**

**ORDER BY ... ASC|DESC** се използва за сортиране на резултата от заявката във възходящ ( **ASC** ) или нисходящ ред ( **DESC** ).

Пример : **SELECT \* FROM Customers**

**ORDER BY Country ASC, CustomerName DESC;**



**Резултат :**   
 Заявката връща резултат от таблицата Customers сортира във възходящ ред по колоната **Country** и сортиран по низходящ ред по колоната **CustomerName**

* **SQL COUNT(), AVG() и SUM() Агрегиращи функции..**

Функцията COUNT () връща броя редове, съответстващи на определени критерии.

Функцията AVG () връща средната стойност на цифрова колона.

Функцията SUM () връща общата сума на цифровата колона.

**Пример: SELECT COUNT(ProductID)**

**FROM Products;**

Заявката връща резултат от броя на записите в **Product** таблицата, като брои колко **ProductID** има записани.



**Пример : SELECT AVG(Price)**

**FROM Products;**

Заявката връща резултат на средната стойност на цената за всички продукти от таблицата **Products**.



**Пример : SELECT SUM(Quantity)**

**FROM OrderDetails;**

Заявката връща резултат на сумата на количеството от таблицата **OrderDetails**.

* **JOINS (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN)**

JOIN клаузата се използва за комбиниране на редове от една или повече таблици, базирани на връзка между тях. Има няколко вида JOINS използват се спрямо това каква информация искам да изкараме от двете таблици като наприме пълна информация която отговаря в двете таблици или информация която е зададе само в едната таблица, а в другата се попълва с празни записи.

**INNER JOIN :** Използва се за взимане на резултати които имат съвпадаши стойности и в двете таблици

Пример :

**SELECT Orders.OrderID, Customers.CustomerName**

**FROM Orders**

**INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;**

В примера искаме да изкараме всички поръчки за даден клиент, за да направим сливането на двете таблици където са записани поръчките на клиентите трябва да направим връзка между таблиците **Customers** и **Orders** като връзката между тях се прави чрез **CustomerID**. Това е така защото имаме foreign key **CustomerID** в таблицата **Orders** и така знаем коя поръчка за кой клиент се отнася.   
 Примерно имаме запис в таблицата **Customer** който е с **CustomerID - 1**. Когато имаме поръчка **Order** тя в себе си съдържа клиент примерно с **CustomerID - 1.**



**Резултат:**Виждаме че клиент с име Wilman Kala има поръка с номер 10248. Така може да изтеглим обстойна информация за поръчката с този номер.

**LEFT JOIN :**

**LEFT JOIN ни позволява да вземем всички резултати от таблицата която е от ляво и да се свържем с таблицата която е от дясно. Ако някой от резултатите в дясно не съществува за таблицата от ляво, той се попълва с NULL.**

**Пример :**

**SELECT Customers.CustomerName, Orders.OrderID**

**FROM Customers**

**LEFT JOIN Orders**

**ON Customers.CustomerID=Orders.CustomerID**

**ORDER BY Customers.CustomerName;**

С този пример искаме да вземем всички клиенти и техните поръчки. Ако даден клиент няма поръчка , резултата на поръчката ще бъде запълнен с NULL.



Виждаме че клиент с име **Alfreds Futterkiste** няма поръчка и неговият резултат се попълва с **NULL.**

**RIGHT JOIN**

**При SQL RIGHT JOIN е същото като LEFT , но при него лявата таблица се попълва с NULL ако няма намерен резултат.**

**Пример :**

**SELECT Orders.OrderID, Employees.LastName, Employees.FirstName**

**FROM Orders**

**RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID**

**ORDER BY Orders.OrderID;**



Виждаме че за **Employee** с имена **West** Adam няма поръчка и записа за **OrderID** остава празен.

**FULL OUTER JOIN...**

**FULL OUTER JOIN** връща всички записи, когато е налице съвпадение в записите за таблици вляво (table1) или right (table2).

Използването на **FULL OUTER JOIN** може да върне много голям брой данни и трябва да се внимава как се използва.

**Пример:**

**SELECT Customers.CustomerName, Orders.OrderID**

**FROM Customers**

**FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID**

**ORDER BY Customers.CustomerName;**

**В примера взимаме всички поръчки независимо дали имат клиент за тях и всички клиенти независимо дали има поръчка за тях!**



**Резултат :**

Виждаме че за клиент с име **Alfreds Futterkiste** нямаме поръчка но той се добавя към крайният резултат. Малко по надолу виждаме че имаме поръчки с ноемера **10382 и 10351**, но за тях няма клиенти. Така резултатите се взимат от двете таблици независимо дали имат резултат.

* **INSERT INTO:**

**INSERT INTO** стейтмънта се използва за добавяне на нов запис в таблица.  
 **Пример :**

**INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)**

**VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');**

Казваме че ще добавяме нов ред в таблицата Customers и изброяваме колоните на таблицата в които ще слагаме стойности в случая :

**(CustomerName, ContactName, Address, City, PostalCode, Country)**

И след това декларираме какви са стойностите които искаме да запишем в дадена колона, като се следва редъ в който сме изброили по горе.

**VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');**

**'Cardinal'** отговаря за колона **CustomerName ,** '**Tom B. Erichsen'** за **ContactName** и т.н.

**Резултат**:  
 Ако заявката мине успешно и се запише новият резултат в таблицата се връща следното съобщение :   
 **You have made changes to the database. Rows affected: 1**

* **UPDATE TABLE...**

Ъпдейтването на запис в таблица става чрез WHERE клауза , за да може да знаем кой точно запис от таблицата искаме да променим.

**Пример :**

**UPDATE Customers**

**SET ContactName='Alfred Schmidt', City='Frankfurt'**

**WHERE CustomerID=1**;  
  
 Тук променяме клиент който е с **CustomerID=**1 като слагаме новите стойности за **ContactName и City чрез  
 SET ContactName='Alfred Schmidt', City='Frankfurt'.**

**Резултат:**

Отново ако заявката е преминала успешно ще видим следното съобщение:

**You have made changes to the database. Rows affected: 1**

* **DELETE**

Изтриване на запис от таблица става отново посредством **WHERE :**

**Пример :**

**DELETE FROM Customers**

**WHERE CustomerName='Alfreds Futterkiste';  
  
Тук искаме да изтрием запис от таблицата Customers където CustomerName='Alfreds Futterkiste'**

**Резултат:**

Отново ако заявката е преминала успешно ще видим следното съобщение:

**You have made changes to the database. Rows affected: 1**

**ПОГЛЕДНИ И ОСТАНАЛИТЕ НЕЩА КАТО:**

**UNION, GROUP BY, HAVING, ALIASES, BETWEEN, LIKE.**

**Може да прочетеш обстойно за тях с примери от :** [**https://www.w3schools.com/sql/**](https://www.w3schools.com/sql/)