databases

20 February 2024

15:46

SQL: w3school

* DB->Table->record->field
* Keyword are not case sensitive
* Semicolon to mark the end of statement or separate two statements
* Important keywords
  + 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

* Create
  + CREATE TABLE Persons (

    PersonID int,

    LastName varchar(255),

    FirstName varchar(255),

    Address varchar(255),

    City varchar(255)

);

* INSERT INTO *table\_name* (*column1*,*column2*,*column3*, ...)

VALUES (*value1*,*value2*,*value3*, ...);

* Can omit column name if we are providing value for all column in the same order
* CREATE DATABASE testDB;
* Read
  + SELECT CustomerName, City FROM Customers;
  + SELECT \* FROM Customers;
  + 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='Mexico';

* Comparison operator
  + =        Equal
  + >        Greater than
  + <        Less than
  + >=        Greater than or equal
  + <=        Less than or equal
  + <>        Not equal. Note: In some versions of SQL this operator may be written as !=
  + BETWEEN        Between a certain range
  + LIKE        Search for a pattern
  + IN        To specify multiple possible values for a column
* SELECT \* FROM Products

ORDER BY Price, date ASC|DESC;

* SELECT \* FROM Customers

ORDER BY Country ASC, CustomerName DESC;

* Logical operator
  + AND
  + OR
  + NOT
* SELECT \* FROM Customers

WHERE NOT Country = 'Spain';

* SELECT TOP 3 \* FROM Customers;
* SELECT *column\_name(s) // mysql*

FROM *table\_name*

WHERE *condition*

LIMIT *number*;

* SELECT TOP *number*|*percent* *column\_name(s) //ms sql*

FROM *table\_name*

WHERE *condition*;

* SELECT MIN(Price)

FROM Products;

* SELECT MIN(*column\_name*)

FROM *table\_name*

WHERE *condition*;

* SELECT MAX(*column\_name*)

FROM *table\_name*

WHERE *condition*;

* SELECT COUNT(*column\_name*)

FROM *table\_name*

WHERE *condition*;

* SELECT SUM(*column\_name*)

FROM *table\_name*

WHERE *condition*;

* SELECT SUM(Price \* Quantity)

FROM OrderDetails

LEFT JOIN Products ON OrderDetails.ProductID = Products.ProductID;

* SELECT AVG(Price) AS [average price]

FROM Products;

* SELECT \* FROM Customers

WHERE city LIKE 'L\_nd\_\_';

* SELECT \* FROM Customers

WHERE Country IN ('Germany', 'France', 'UK');

* SELECT \* FROM Products

WHERE Price BETWEEN 10 AND 20

* 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;

* Wildcard

|  |  |
| --- | --- |
| % | Represents zero or more characters |
| \_ | Represents a single character |
| [] | Represents any single character within the brackets \* |
| ^ | Represents any character not in the brackets \* |
| - | Represents any single character within the specified range \* |
| {} | Represents any escaped character \*\* |

|  |  |  |
| --- | --- | --- |
| \* | Represents zero or more characters | bl\* finds bl, black, blue, and blob |
| ? | Represents a single character | h?t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ! | Represents any character not in the brackets | h[!oa]t finds hit, but not hot and hat |
| - | Represents any single character within the specified range | c[a-b]t finds cat and cbt |
| # | Represents any single numeric character | 2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295 |

* 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









* SELECT ProductID, ProductName, CategoryName

FROM Products

INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;

* SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City

FROM Customers A, Customers B

WHERE A.CustomerID <> B.CustomerID //self join

AND A.City = B.City

ORDER BY A.City;

* SELECT City, Country FROM Customers

WHERE Country='Germany'

UNION

SELECT City, Country FROM Suppliers

WHERE Country='Germany'

ORDER BY City;

* SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country;

* SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country

HAVING COUNT(CustomerID) > 5; // having got added as where cant be used aggr func

* 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 = 10);

* SELECT Customers.CustomerName, Orders.OrderID

INTO CustomersOrderBackup2017

FROM Customers

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

INSERT INTO Customers (CustomerName, City, Country)

SELECT SupplierName, City, Country FROM Suppliers;

* 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;

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

FROM Products;

* Update
  + UPDATE *table\_name*

SET *column1*=*value1*,*column2*=*value2*, ...

WHERE *condition*;

* ALTER TABLE Customers

ADD Email varchar(255);

* Delete
  + DELETE FROM *table\_name*WHERE *condition*;
  + DROP TABLE Customers;
  + DROP DATABASE testDB;
* Stored procedure
  + CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)

AS

SELECT \* FROM Customers WHERE City = @City

GO;

* EXEC SelectAllCustomers @City = 'London';
* Comments : -- or /\* …\*/
* Operators: all math , bitwise and comparison operators work
* Backup
  + BACKUP DATABASE testDB

TO DISK = 'D:\backups\testDB.bak'

WITH DIFFERENTIAL;

* Constraint
  + [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
  + [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
  + [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
  + [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables
  + [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition
  + [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified
  + [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly
  + CREATE TABLE Orders (

    OrderID int NOT NULL,

    OrderNumber int NOT NULL,

    PersonID int,

    PRIMARY KEY (OrderID),

    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

);

* CREATE VIEW [Brazil Customers] AS

SELECT CustomerName, ContactName

FROM Customers

WHERE Country = 'Brazil';

* SQL injection: if user gives shady input which contains added sql cmd in place of a var