# SQL Keywords

| Keyword | Description |
|---|---|
| ADD | Adds a column in an existing table |
| ADD CONSTRAINT | Adds a constraint after a table is already created |
| ALTER | Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table |
| ALTER COLUMN | Changes the data type of a column in a table |
| ALTER TABLE | Adds, deletes, or modifies columns in a table |
| ALL | Returns true if all of the subquery values meet the condition |
| AND | Only includes rows where both conditions is true |
| ANY | Returns true if any of the subquery values meet the condition |
| AS | Renames a column or table with an alias |
| ASC | Sorts the result set in ascending order |
| BACKUP DATABASE | Creates a back up of an existing database |
| BETWEEN | Selects values within a given range |
| CASE | Creates different outputs based on conditions |
| CHECK | A constraint that limits the value that can be placed in a column |
| COLUMN | Changes the data type of a column or deletes a column in a table |
| CONSTRAINT | Adds or deletes a constraint |
| CREATE | Creates a database, index, view, table, or procedure |
| CREATE DATABASE | Creates a new SQL database |
| CREATE INDEX | Creates an index on a table (allows duplicate values) |

| | |
|---|---|
| CREATE OR REPLACE VIEW | Updates a view |
| CREATE TABLE | Creates a new table in the database |
| CREATE PROCEDURE | Creates a stored procedure |
| CREATE UNIQUE INDEX | Creates a unique index on a table (no duplicate values) |
| CREATE VIEW | Creates a view based on the result set of a SELECT statement |
| DATABASE | Creates or deletes an SQL database |
| DEFAULT | A constraint that provides a default value for a column |
| DELETE | Deletes rows from a table |
| DESC | Sorts the result set in descending order |
| DISTINCT | Selects only distinct (different) values |
| DROP | Deletes a column, constraint, database, index, table, or view |
| DROP COLUMN | Deletes a column in a table |
| DROP CONSTRAINT | Deletes a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint |
| DROP DATABASE | Deletes an existing SQL database |
| DROP DEFAULT | Deletes a DEFAULT constraint |
| DROP INDEX | Deletes an index in a table |
| DROP TABLE | Deletes an existing table in the database |
| DROP VIEW | Deletes a view |
| EXEC | Executes a stored procedure |
| EXISTS | Tests for the existence of any record in a subquery |
| FOREIGN KEY | A constraint that is a key used to link two tables together |
| FROM | Specifies which table to select or delete data from |

| | |
|---|---|
| FULL OUTER JOIN | Returns all rows when there is a match in either left table or right table |
| GROUP BY | Groups the result set (used with aggregate functions: COUNT, MAX, MIN, SUM, AVG) |
| HAVING | Used instead of WHERE with aggregate functions |
| IN | Allows you to specify multiple values in a WHERE clause |
| INDEX | Creates or deletes an index in a table |
| INNER JOIN | Returns rows that have matching values in both tables |
| INSERT INTO | Inserts new rows in a table |
| INSERT INTO SELECT | Copies data from one table into another table |
| IS NULL | Tests for empty values |
| IS NOT NULL | Tests for non-empty values |
| JOIN | Joins tables |
| LEFT JOIN | Returns all rows from the left table, and the matching rows from the right table |
| LIKE | Searches for a specified pattern in a column |
| LIMIT | Specifies the number of records to return in the result set |
| NOT | Only includes rows where a condition is not true |
| NOT NULL | A constraint that enforces a column to not accept NULL values |
| OR | Includes rows where either condition is true |
| ORDER BY | Sorts the result set in ascending or descending order |
| OUTER JOIN | Returns all rows when there is a match in either left table or right table |
| PRIMARY KEY | A constraint that uniquely identifies each record in a database table |
| PROCEDURE | A stored procedure |

| | |
|---|---|
| RIGHT JOIN | Returns all rows from the right table, and the matching rows from the left table |
| ROWNUM | Specifies the number of records to return in the result set |
| SELECT | Selects data from a database |
| SELECT DISTINCT | Selects only distinct (different) values |
| SELECT INTO | Copies data from one table into a new table |
| SELECT TOP | Specifies the number of records to return in the result set |
| SET | Specifies which columns and values that should be updated in a table |
| TABLE | Creates a table, or adds, deletes, or modifies columns in a table, or deletes a table or data inside a table |
| TOP | Specifies the number of records to return in the result set |
| TRUNCATE TABLE | Deletes the data inside a table, but not the table itself |
| UNION | Combines the result set of two or more SELECT statements (only distinct values) |
| UNION ALL | Combines the result set of two or more SELECT statements (allows duplicate values) |
| UNIQUE | A constraint that ensures that all values in a column are unique |
| UPDATE | Updates existing rows in a table |
| VALUES | Specifies the values of an INSERT INTO statement |
| VIEW | Creates, updates, or deletes a view |
| WHERE | Filters a result set to include only records that fulfill a specified condition |

# SQL Server String Functions

| Function | Description |
|---|---|
| ASCII | Returns the ASCII value for the specific character |
| CHAR | Returns the character based on the ASCII code |
| CHARINDEX | Returns the position of a substring in a string |
| CONCAT | Adds two or more strings together |
| Concat with + | Adds two or more strings together |
| CONCAT_WS | Adds two or more strings together with a separator |
| DATALENGTH | Returns the number of bytes used to represent an expression |
| DIFFERENCE | Compares two SOUNDEX values, and returns an integer value |
| FORMAT | Formats a value with the specified format |
| LEFT | Extracts a number of characters from a string (starting from left) |
| LEN | Returns the length of a string |
| LOWER | Converts a string to lower-case |
| LTRIM | Removes leading spaces from a string |
| NCHAR | Returns the Unicode character based on the number code |
| PATINDEX | Returns the position of a pattern in a string |
| QUOTENAME | Returns a Unicode string with delimiters added to make the string a valid SQL Server delimited identifier |
| REPLACE | Replaces all occurrences of a substring within a string, with a new substring |
| REPLICATE | Repeats a string a specified number of times |
| REVERSE | Reverses a string and returns the result |

| | |
|---|---|
| RIGHT | Extracts a number of characters from a string (starting from right) |
| RTRIM | Removes trailing spaces from a string |
| SOUNDEX | Returns a four-character code to evaluate the similarity of two strings |
| SPACE | Returns a string of the specified number of space characters |
| STR | Returns a number as string |
| STUFF | Deletes a part of a string and then inserts another part into the string, starting at a specified position |
| SUBSTRING | Extracts some characters from a string |
| TRANSLATE | Returns the string from the first argument after the characters specified in the second argument are translated into the characters specified in the third argument. |
| TRIM | Removes leading and trailing spaces (or other specified characters) from a string |
| UNICODE | Returns the Unicode value for the first character of the input expression |
| UPPER | Converts a string to upper-case |

## SQL Server Math/Numeric Functions

| Function | Description |
|---|---|
| ABS | Returns the absolute value of a number |
| ACOS | Returns the arc cosine of a number |
| ASIN | Returns the arc sine of a number |
| ATAN | Returns the arc tangent of a number |
| ATN2 | Returns the arc tangent of two numbers |

| | |
|---|---|
| AVG | Returns the average value of an expression |
| CEILING | Returns the smallest integer value that is >= a number |
| COUNT | Returns the number of records returned by a select query |
| COS | Returns the cosine of a number |
| COT | Returns the cotangent of a number |
| DEGREES | Converts a value in radians to degrees |
| EXP | Returns e raised to the power of a specified number |
| FLOOR | Returns the largest integer value that is <= to a number |
| LOG | Returns the natural logarithm of a number, or the logarithm of a number to a specified base |
| LOG10 | Returns the natural logarithm of a number to base 10 |
| MAX | Returns the maximum value in a set of values |
| MIN | Returns the minimum value in a set of values |
| PI | Returns the value of PI |
| POWER | Returns the value of a number raised to the power of another number |
| RADIANS | Converts a degree value into radians |
| RAND | Returns a random number |
| ROUND | Rounds a number to a specified number of decimal places |
| SIGN | Returns the sign of a number |
| SIN | Returns the sine of a number |
| SQRT | Returns the square root of a number |
| SQUARE | Returns the square of a number |

| | |
|---|---|
| SUM | Calculates the sum of a set of values |
| TAN | Returns the tangent of a number |

## SQL Server Date Functions

| Function | Description |
|---|---|
| CURRENT_TIMESTAMP | Returns the current date and time |
| DATEADD | Adds a time/date interval to a date and then returns the date |
| DATEDIFF | Returns the difference between two dates |
| DATEFROMPARTS | Returns a date from the specified parts (year, month, and day values) |
| DATENAME | Returns a specified part of a date (as string) |
| DATEPART | Returns a specified part of a date (as integer) |
| DAY | Returns the day of the month for a specified date |
| GETDATE | Returns the current database system date and time |
| GETUTCDATE | Returns the current database system UTC date and time |
| ISDATE | Checks an expression and returns 1 if it is a valid date, otherwise 0 |
| MONTH | Returns the month part for a specified date (a number from 1 to 12) |
| SYSDATETIME | Returns the date and time of the SQL Server |
| YEAR | Returns the year part for a specified date |

## SQL Server Advanced Functions

| Function | Description |
|---|---|
| CAST | Converts a value (of any type) into a specified datatype |
| COALESCE | Returns the first non-null value in a list |
| CONVERT | Converts a value (of any type) into a specified datatype |
| CURRENT_USER | Returns the name of the current user in the SQL Server database |
| IIF | Returns a value if a condition is TRUE, or another value if a condition is FALSE |
| ISNULL | Return a specified value if the expression is NULL, otherwise return the expression |
| ISNUMERIC | Tests whether an expression is numeric |
| NULLIF | Returns NULL if two expressions are equal |
| SESSION_USER | Returns the name of the current user in the SQL Server database |
| SESSIONPROPERTY | Returns the session settings for a specified option |
| SYSTEM_USER | Returns the login name for the current user |
| USER_NAME | Returns the database user name based on the specified id |

SELECT * FROM Customers;

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

```
SELECT CustomerName,City FROM Customers;
SELECT DISTINCT Country FROM Customers;
SELECT COUNT(DISTINCT Country) FROM Customers;
```

yukaridaki komut firefoxta calismaz.

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

bu calisir iste!

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

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

SELECT * FROM Customers
WHERE NOT Country='Germany';

SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='München');

SELECT * FROM Customers
ORDER BY Country;
```

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

ya da ASC de kullanilabilirdi!

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName!!!

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

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

eger yeni bir sutun eklenecekse boyle yapilir ama eger var olan sutunlara bir ekleme yapilacaksa asagidaki gibi yapilir!!!

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

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

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

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

```
SELECT TOP 3 * FROM Customers;
```

```
SELECT TOP 50 PERCENT * FROM Customers;
```

```
SELECT TOP 3 * FROM Customers
```

```
WHERE Country='Germany';

SELECT MIN(Price) AS SmallestPrice
FROM Products;

SELECT MAX(Price) AS LargestPrice
FROM Products;

SELECT COUNT(ProductID)
FROM Products;

SELECT AVG(Price)
FROM Products;

SELECT SUM(Quantity)
FROM OrderDetails;
```

| 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" |

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

| Symbol | Description | Example |
|---|---|---|
| * | 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 a range of characters | 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 |

| Symbol | Description | Example |
|---|---|---|
| % | 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 a range of characters | c[a-b]t finds cat and cbt |

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```
bununla b,s veya p ile baslayan herhangi sehirler seciliyor.

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';
```
bununla da b,s veya p ile baslamayanlar listeleniyor!!!

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

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

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);
```

```
SELECT * FROM Products
WHERE ProductName BETWEEN "Carnarvon Tigers" AND "Chef
Anton's Cajun Seasoning"
ORDER BY ProductName;
```

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;
```
Bu iki tarih arasindaki tum degerleri alir!!!

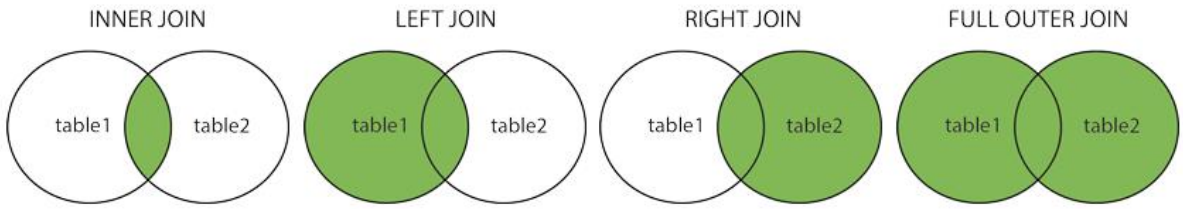
```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

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

eger bosluk birakip yazmak istiyorsan bu sekilde kullanmalisin ya da ‘’ ile!!!

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ',
' + Country AS Address
FROM Customers;
```

arti ile concat yapar ve ayni yere yan yana yazar!!!




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

```
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);
```

##JOINLER ILE ILGILI NOTLAR##
Inner join ile atiyorum iki tabloyu birlestirecegiz ama sadece kesisimlerini aliyoruz aslinda. Join ile inner join ayni. Left join’i soyle dusun; iki tane kume var ve soldaki kumenin tum elemanlarini al ve sagdakiyle kesisimlerini yaz, ama soldaki her turlu olacak, kesisimleri yoksa null olarak gozukecek. Full join de aslinda left veya right join gibidir, hangi kumeyi once yazdiysan ona gore degisir.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
```

```
FULL OUTER JOIN Orders ON
Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

yukarida sari ile isaretledigim yer sol kume, digeri sag kume!!!

| CustomerName | OrderID |
|---|---|
| Alfreds Futterkiste | *Null* |
| Ana Trujillo Emparedados y helados | 10308 |
| Antonio Moreno Taquería | *Null* |

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

union customerlerdeki sehirlerle supplierlardaki sehirleri birlestirir, tekrar eden ifadeleri bir kere yazar sadece. eger tekrar tekrar yazmasini istiyorsan UNION ALL diye yazmalisin!!!

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE
Products.SupplierID = Suppliers.supplierID AND Price < 20);
```

```
SELECT ProductName
```

```
FROM Products
WHERE ProductID = ANY
(SELECT ProductID
FROM OrderDetails
WHERE Quantity = 10);

SELECT * INTO CustomersBackup2017
FROM Customers;

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 +
ISNULL(UnitsOnOrder, 0))
FROM Products;

CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
EXEC SelectAllCustomers;

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
SELECT * FROM Customers WHERE City = @City
GO;
EXEC SelectAllCustomers @City = 'London';

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30),
@PostalCode nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND PostalCode =
@PostalCode
GO;
```

```
EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1
1DP';
```

-- YAZINCA SONRA GELENLER COMMENT OLARAK ALGILANIYOR!!!
Multiline comments yazacaksan ‘/*’ ve ‘*/’ kullanmalisin!!!

<> not equal to manasina gelir.

```
CREATE DATABASE testDB;
```

```
DROP DATABASE databasename;
```

```
BACKUP DATABASE testDB
TO DISK = 'D:\backups\testDB.bak';
```

```
BACKUP DATABASE databasename
TO DISK = ‘D:\backups\testDB.bak’
WITH DIFFERENTIAL;
```
sadece yapilan guncellemeleri alir!!!

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

```
CREATE TABLE TestTable AS
SELECT customername, contactname
FROM customers;
```
baska bir tablo kullanarak tablo olusturma!!!

```
DROP TABLE Shippers;
```

```
TRUNCATE TABLE table_name;
```
tabloyu komple silmeden sadece icindeki datayi siler!!!

```
ALTER TABLE Persons
ADD DateOfBirth date;
```
bu su anlama geliyor; ben persons tablosunda degisiklik yapmak istiyorum, dateofbirth diye bir sutun eklemek istiyorum ve bu sutun date turunde bir bilgi saklasin diyorum!!!

```
CREATE TABLE Persons (
        ID int NOT NULL,
         LastName varchar(255) NOT NULL,
        FirstName varchar(255) NOT NULL,
        Age int
);
```
not null diyere id nin null olamayacagini soyledik!!!

```
CREATE TABLE Persons (
        ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```
unique ile id nin farkli degerler almasi gerektigini soyledik!!!

```
CREATE TABLE Persons (
        ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```
primary key null olamaz!!!

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

```
CREATE TABLE Persons (
        ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

city ye default olarak sandnes degerini atiyor!!! (default)

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

lastname ve first name e idxpname indexini atadik!!!

```
ALTER TABLE table_name
DROP INDEX index_name;
```

indexi silmek icin bunu yap!!!

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

yukarida identity(1,1) diyerek bu degerin 1 ile baslayip otomatik bir sekilde birer birer artacagini belirttik. Yani tabloya yeni bir satir eklenince bu deger otomatik geliyor!!!

**SQL Server** comes with the following data types for storing a date or a date/time value in the database:

- `DATE` - format YYYY-MM-DD
- `DATETIME` - format: YYYY-MM-DD HH:MI:SS
- `SMALLDATETIME` - format: YYYY-MM-DD HH:MI:SS
- `TIMESTAMP` - format: a unique number

tarih ile saati hep ayri yazmaya calis, cunku veriyi cekmesi kolay olur boyle!!!

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
SELECT * FROM [Brazil Customers];
```

yukarida bir view olusturduk ve bu view i cagirdik!!!

```
CREATE OR REPLACE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName, City
FROM Customers
WHERE Country = 'Brazil';
```

olusturulan view i guncelledik!!!

```
DROP VIEW [Brazil Customers];
```

view i sildik!!!

https://www.youtube.com/watch?v=rKwoBdlfo5g