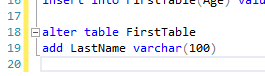
**Structed Query Language**

DDL- data defination language

**Create (**misol create table\_name(column\_name datatype, column\_name2 datatype, …..**)**

**Alter –** table ni arxitekturasini ozgartirishda kerak bo’ladi

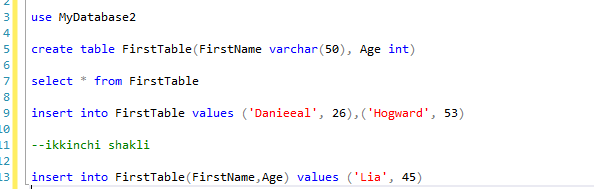
**Drop –** drop butunlay table ni ochirib yuboradi

**Truncate –** table ni ichidagi ma’lumotlarni ochrish uchun ishlatilinadi, Faqat column name lar qoladi Ichidagi malumotlar yoqoladi

**Varchar-** so’zlarni kiritiladigaan column bosa varchar beriladi

**Int –** butun son

Insert into bilan table ga value qo’shsa bo’ladi:

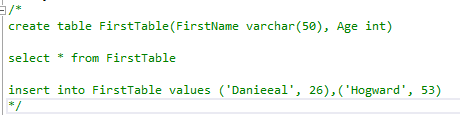
  
--varchar ni yonidagi son bu column name ga kiritiladiga max harflar

Comment qilish va olib tashlash

CTRL+K+C **comment**

CTRL+K+U **uncomment**

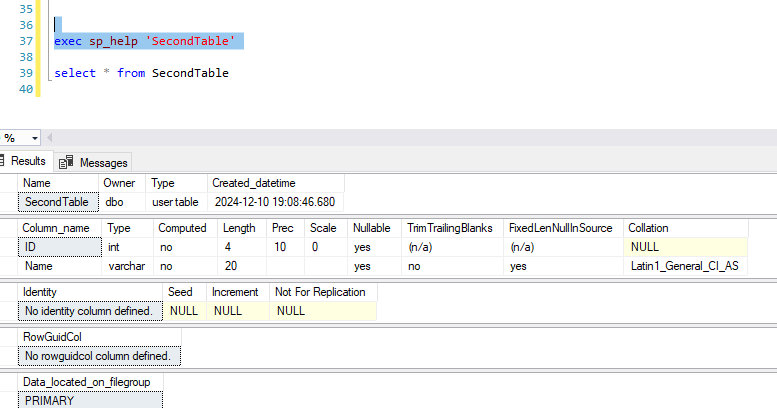
---keyin bir necha rowlarni birdan comment qilish uchun /\* dan foydalanamiz ya’ni qayerda boshlansa /\* tugash qatoriga esa \*/ ni qo’yamiz. Masalan:



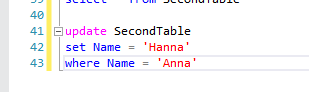
DDL- yani data definition language bilan table ni o’zini ozgartirish uchun ishlatilinadi

**DML yani Data manipulation Language bilan table ni ichidagi ma’lumotlarni boshqaramiz.**

exec sp\_help bu table ni arxitekturasini ko’rsatib beradi. Masalan:



**Update** bilan ishlash: bu bilan malum bir table ichidagi malumotlarni o’zgartira olamiz. Masalan:

Bu yerda data update qilinyapti demak birinchi nimani ozgartirmoqchi bolsak oshani **set** bilan yozib olamiz va **where** bilan qaysi datani ozgartirish kerakligini kiritamiz.

**Delete from** SecondTable = **Truncate** SecondTable

Bu code bilan esa qaysidir aniq bir rowni o’chirib tashlasa bo’ladi:

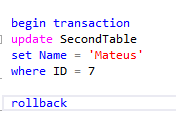
**How to drop column**

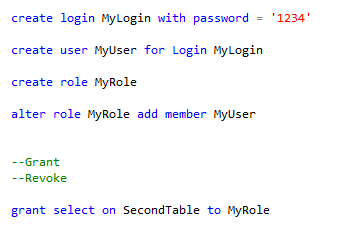
alter bilan qaysi table ligi ko’rsatiladi va drop column qilib column name yoziladi.

**DQL – Data Query Language**

--Select

**TCL Transaction Control Language**

**Begin transaction** bu bilan yozgan codimizni check qilishimiz mumkin yani biror code qatori oldiddan shuni yozib kyn run qilsak nma bolishini korib oolishimiz mumkin agar togri bolsa yoki code ni run qilish ni davom ettirmoqchi bo’lsak **commit**  deb kiritamiz yoki o’z holiga qaytarmoqchi bo’lsak **rollback** deb yozamiz. Masalan:



DCL Data Control Language – bu bilan table larga login password qoyish va shunga oxshash operatsiyalar bajariladi.

**DATA TYPES**

url : https://www.w3schools.com/sql/sql\_datatypes.asp

select \* from INFORMATION\_SCHEMA.TABLES - bu database dagi barcha table larni chiqarib beradi.

***Constraints***

Primary key

Foreign key

Table dagi Constraintni nomini izlash uchun exec sp\_help 'table\_name' yoziladi.

Constraint ni ochirib tashlash:

 Alter table *table\_name*

Drop constraint *constraint\_name*

**Wildcards(bular like bilan ishlaydi)**

**%** - bu qaysidir harfdan kyn bir yoki bir nechta harf kelgan taqdirda uni qidirish uchun ishlatilinadi. Masalan: *movies* tabledan ‘King kong’ filmini qidirishim kerak shu holatda man ‘King%’ bersam, har qanday nomi King deb boshlangan kinolarni topib beradi.

**\_** - bu esa bitta harf ya’ni characterni bildiradi. Masalan ‘King Kong’ ni qidirmoqchi bo’lsam, ‘King Kon\_’ desam chiqadi. '\_r%' bu misolda 2-harfi r bolgan so’z qidirilyapti

**[] –** bu bilan ma’lum bir harfdan harfgacha bo’lgan harflarni kiritsak bo’ladi. Masalan:

select \* from soccer.player\_mast

where playing\_club not like '[A-L]%' bu yerda playing\_clubida bosh harfi A dan L gacha bo’lmagan club lar chiqadi.

**distinct –** duplicatlar ni yoqotin bitta qilib chiqaradi value larni

select top 10 percent \* from employee.employees

order by salary

**Bu yerda** top 10% ga kiradiganlarni chiqarish uchun query yozilgan

select \* from employee.employees

order by salary desc

offset 3 rows fetch next 5 rows only

Offsetni SQL da ishlatilinishi, bu yerda 3 ta row tashlab 5 tasini olishi ko’zda tutilgan. ***OFFSET*** va ***FETCH***

Ikki column ni qo’shib bitta column qilish uchun :

select emp\_name + ' ' + job\_name as fullname from employee.employees

yoziladi.

**Lesson 6**

**Union all –** bu ikkita jadvalni qo’shish uchun ishlatilinadi.(duplicatlarni olib tashlamaydi)

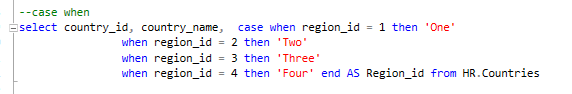
**Union –** bunda esa by default duplicatlarni olib tashlab, asc order qilib chiqarib beradi.

**Except** – esa birinchi ko’rsatilgan jadvalda bor lkn ikkinchisida yo’qlarini chiqarib beradi ya’ni difference qiladigan value larni chiqaradi.

**Intersect** – Bu esa except ni teskarisi ya’ni ikkala jadvalda borlarini chiqarib beradi. Ikkalasida bir xil bolgan data ni chiqaradi.

**If lar SQL da**

**Case when … then …. End**

* *Case when condition then true result else false result* -- shu structura bo’yicha ishlaydi lekin har doim ham else shartmas. Ammo **end** ni har doim qo’yish kerak.
* 

**IIF**

Bu esa xuddi case when ni ikkinchi xolati faqat excel ga oxshab, () va , lar bilan bo’ladi syntax lar. Misol:

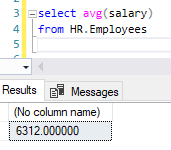
select country\_id, country\_name, IIF(region\_id = 1, 'One',

IIF(region\_id=2,'Two',

IIF(region\_id=3,'Three',

'Four'))) as region\_id from HR.Countries

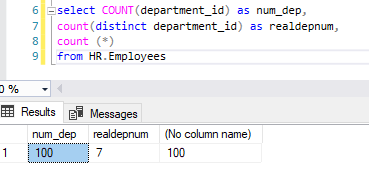
**aggregate functions**

Avg(column\_name)

select cast(avg(salary) as decimal(8,2)) as average\_salary

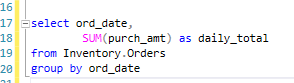
from HR.Employees

bu yerda esa type decimal da va column ga name berib chiroyli chiqarish(cast bilan)



**Count**

Count sanaydi. Misolda distinctsiz ishlatilganida column dagi barcha rowlarni sanab chiqqan, agar distinct qo’shsak duplicatlarni olib tashlab, aniq nechtaligini sanaydi, **count(\*)** ham barcha rowlarni sanab chiqadi, **null** larni ham lekin **count(department\_id)** esa **null** ni olmaydi.

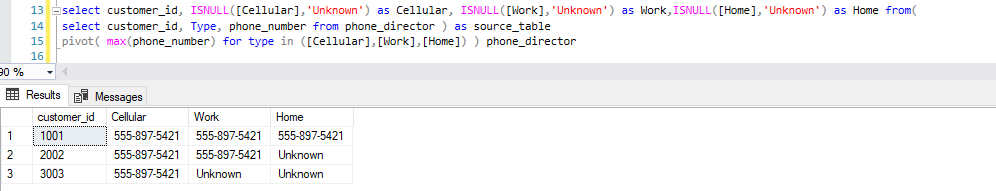


Sum ga misol:

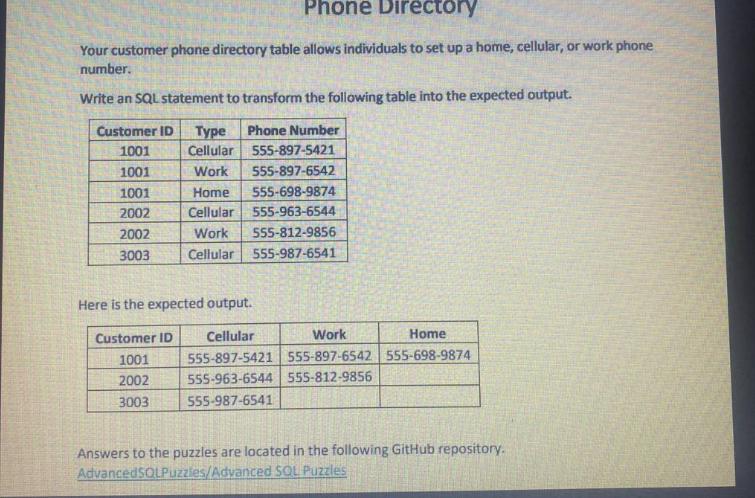
Sum, max ya’ni agregation lar bn group by ni ishlatish kk chunki bular bittadan olrtiq bo’lgan valuelarni hisoblash uchun ishlatilinadi. Misolda ham group\_date bilan har birk un ajratilib o’sha kunlarning Sumi hisoblangan.



Count bilan column dagi unique valuelarni sanash uchun distinct ishlatilinadi.

**Isnull()** bu function cell dagi value ni null emasgini tekshiradi va o’rniga boshqa narsa qo’ysa bo’ladi. 

Bu yerda pivot qanday yasash ham keltirilgan, shuni yodda tutish kerakki pivot da har doim agregat keltirib otish kerak shu bilan birgalikda column da keladigan value larni [] qavslari bn yozish kerak.



Shu puzzle ni yechimi teppada keltirildi.Demak

**Pivot()** yozib olib birinchi Orinda ortada yani row qatorda ham column qatorda ham kelmaydigan, ya’ni value larni **agregat** bilan yozib olib, **for** ni source\_tabledagi qaysi columndagi datalarni pivotdagi column ga o’tkazmoqchi bo’lsak oshani yozamz, va in bilan column larni kiritamz.

**Working orders**

**Execution Order:**

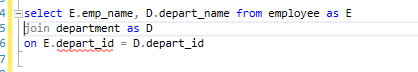
1. **FROM**:
   * The database engine starts by identifying the tables or views from which data will be retrieved.
2. **WHERE**:
   * Filters rows based on the specified condition(s). Only rows meeting these criteria proceed to the next step.(Where esa table ni o’ziga filter beradi)
3. **GROUP BY**:
   * Groups the filtered rows based on the specified column(s). Aggregation functions like SUM, COUNT, or AVG are then applied to each group.
4. **HAVING**:
   * Filters the grouped rows based on the result of the aggregation. Only groups meeting these conditions proceed.(Having bu group by qilingan resultlarga filter berish uchun ishlatilinadi , va u aggregation lar bilan ishlatilinadi)
5. **SELECT**:
   * Chooses the columns or calculations to include in the result set.
6. **ORDER BY**:
   * Sorts the final result set based on one or more columns or expressions.

**Fact table and dimension table**

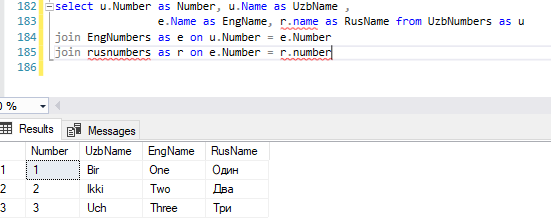
**Fact table** - bu measurable ya’ni hisob-kitoblar va analizlar olib boriladigan table bo’lib doim o’zgarib turadi va foreign keylar joylashadi asosan.

**Dimension table** – esa descriptive ya’ni tavfsizlovchi data lar joylanadi,analiz uchun contex ya’ni ma’lumot beruvchi datalar bu yerda primary keylar joylashadi asosan

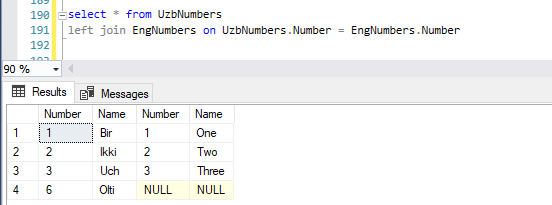
**Joins**

Misol:

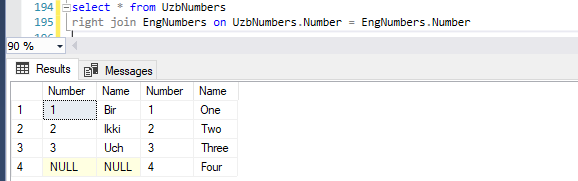
--Inner join esa ikkala table dagi bir biriga mos bo’lgan rowlarni qo’shadi. Misol:



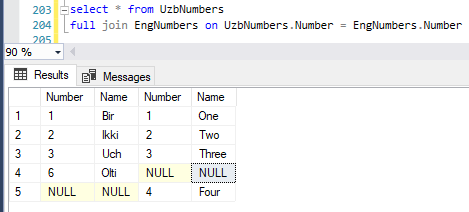
--Left join bu chap tarafdagi tabledagi malumotlarni o’zida saqlaydi, va kelayotgan tabledagi boshqa ma’lumotni kirgizmaydi.

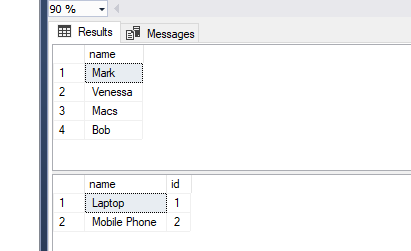


–right join esa shuni teskarisi ya’ni kelayotgan tabledagi barcha ma’lumotlarni chiqarib, qabul qilayotgandagini mosini oladi.



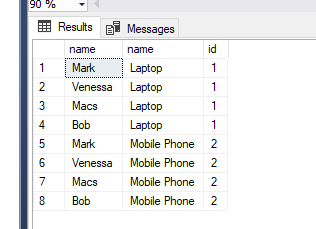
-–full join

esa bor hamma narsani qo’shib tashlaydi.

--Cross join bu ikkita table dagi ma’lumotlarni bir biriga ko’paytirish uchun ishlatilinadi.

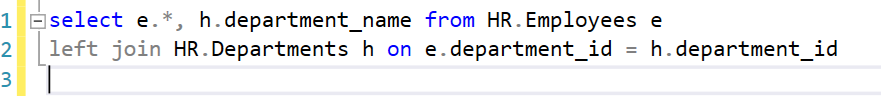
Bizda shunaqa ikkita table bor edi.

Agar cross joinni ishlatsak:



Bunaqa natijaga ega bo’lamiz.

--Self join



Bu misol e.\* uchun berildi. Bu yerda e table dagi barcha ma’lumot chiqsin, h table dagi faqat department name chiqsin deyilgan. Hamma columnlar ni bitta bitta yozib chiqmaslik uchun bu.

****

Cell dagi keraksiz space ni o’chirish – **trim()**

--cross apply = The CROSS APPLY operator is semantically similar to INNER JOIN operator. It retrieves those records from the table valued function and the table being joined, where it finds matching rows between the two.

--outer apply = On the other hand, OUTER APPLY retrieves all the records from both the table valued function and the table, irrespective of the match.

Apply lar bilan functionlarni ishlatsa bo’ladi. Joinlar bilan esa yo’q.

**Data and time functions**

Getdate() – database dagi date ni olib chiqaradigan function. Return *datetime* type

Example: select GETDATE()

GETUTCDATE() – hozirgi universal time zone dagi date ni chiqarib beradi

CURRENT\_TIMESTAMP – bu hozirgi vaqtni chiqarish uchun ishlatilinadi.

ISDATE

ISDATE(expression) – qavs ichida berilgan narsani tekshiradi va datetime bo’lsa 1 qaytaradi, bo’lmasa 0

SELECT ISDATE('Hello world!'); >>> result: 0

SELECT ISDATE('2017'); >>> result: 1

DATEADD

DATEADD(*time\_which\_we\_select\_to\_pus\_or\_minus(interval)* , *the\_number\_to\_add\_or\_remove , ‘DATE’* )

Example: dateadd(month, 2 , ’2024-06-08’) >>> result ‘2024-08-08’

2 oy qoshildi

* year, yyyy, yy = Year
* quarter, qq, q = Quarter
* month, mm, m = month
* dayofyear, dy, y = Day of the year
* day, dd, d = Day
* week, ww, wk = Week
* weekday, dw, w = Weekday
* hour, hh = hour
* minute, mi, n = Minute
* second, ss, s = Second
* millisecond, ms = Millisecond

intervals that can be written to the first part;

DATEDIFF

DATEDIFF – bu function ikkita date dan farqini hisoblash uchun ishlatilinadi.

Example: DATEDIFF(interval, firstdate, seconddate ) (*seconddate dan first dateni ayiradi; secondate - firstdate* )

Datediff(month, ‘2024-08-01’, ‘2024-04-24’) >>> result -4

DATEFROMPARTS

SELECT DATEFROMPARTS(2018, 10, 31) AS DateFromParts;

DATEFROMPARTS(year, month, day) -- Bu function 3 talik raqamni date ga aylantirib beradi

**Year** – har doim 4 talik raqam bo’lishi kerak;

**Month** – 1 dan 12 gacha bo’lgan raqamlar bo’lishi kerak;

**Day** – 1 dan 31 gacha bo’lishi kerak.

DATENAME

DATENAME(interval, date) bu function intervalda qaysi date ni ko’rsatsak o’shanisini chiqarib beradi. (resultda nvarchar qaytaradi.) bu esa faqat stringda qaytaradi. Month name va week name kerak bo’lsa.

SELECT DATENAME(year, '2017/08/25') result >>> 2017

SELECT DATENAME(month, '2017/08/25') result >>> august

DATEPART

DATEPART(interval, date)- bu function ham dataname ga o’xshab ishlaydi lekin, int qaytaradi resultda va operation lar bilan ishlatsa bo’ladi.

Intervals can be used:

* year, yyyy, yy = Year
* quarter, qq, q = Quarter
* month, mm, m = month
* dayofyear, dy, y = Day of the year
* day, dd, d = Day of the month
* week, ww, wk = Week
* weekday, dw, w = Weekday
* hour, hh = hour
* minute, mi, n = Minute
* second, ss, s = Second
* millisecond, ms = Millisecond
* microsecond, mcs = Microsecond
* nanosecond, ns = Nanosecond
* tzoffset, tz = Timezone offset
* iso\_week, isowk, isoww = ISO week

SELECT DATEPART(yy, '2017/08/25') >>> result : 2017 (type int)

DAY

DAY(date) – bu function berilgan to’liq sanadan kunni olib chiqib beradi. Return int

SELECT DAY('2017/08/13 09:08') >>> result: 13

MONTH

MONTH(date) – bu function esa date dan month ni olib beradi. Return int.

select MONTH('2024-11-25') result >> 11

YEAR

YEAR(date) – this function return year of the given date. Return int

select year('2022-06-28') Result >>> 2022

SYSDATETIME()

SYSDATETIME() – this function return system datatime or time where sql server is located.

SELECT SYSDATETIME()

**Mathematical functions**

**Floor()**

**Pi()**

**Power()**

Text functions

ASCII()

ASCII(character) - bu ascii jadvaldagi ma’lum bir characterni codini yuboradi.

select ASCII('a') as da result >>> 97

CHAR()

CHAR(code) – bu ascii jadvaldagi code ni kirgizganimizda characterni qaytaradi.

select CHAR(97) result >>> a

UNICODE

UNICODE(character\_expression) – berilgan stringni unicode ga o’tkazadi.

SELECT UNICODE('Atlanta'); result >>>> 65

NCHAR()

NCHAR(number\_code) – bu uni code ni characterga aylantirib beradi faqat unicode dan.

CHARINDEX()

CHARINDEX(substring, string, start) – bu function string dagi berilgan characterni(substring) positionini qidirish uchun ishlatilinadi. Substring – nimani qidirayotgan bo’lsak oshani kiritamiz, string bu butun string, start search qayerdan boshlanishini belgilash. DEFAULT is 1

select CHARINDEX('t', 'Customer') result >>> 4

CONCAT() –

CONCAT(string1, string2, ...., string\_n) – Concat ikkita yoki undan ortiq stringlar ni qo’shish uchun ishlatilinadi.(concat null larni ignor qiladi.)

select CONCAT('Sardor','Ismatov','Age','22') result >>> SardorIsmatovAge22

CONCAT\_WS(‘,’ , ‘A’, ‘B’, ‘C’)

CONCAT\_WS(separator, string1, string2, ...., string\_n)- concat\_withSeparator

Bu function ham stringlarni qo’shadi, faqatgina separator berib, ajratib return qiladi.

select CONCAT\_WS('-','Sardor','Ismatov','Age','22') >>> result: Sardor-Ismatov-Age-22

DATALENGTH()

DATALENGTH(expression) – bu function expressionda nechta character borligini sanab beradi, hamma narsani sanaydi: space larni ham, , - . larni ham

SELECT DATALENGTH('2017-08') result >>>> 7

SELECT DATALENGTH('   W3Schools.com   ') >>>> result 19

DIFFERENCE

DIFFERENCE(expression, expression) bu ikkita string soundex value siga qarab farqini topadi va 0 dan 4 gacha bo’lgan int da baholaydi. 0 weak similarity and 4 strong similarity.

 The SOUNDEX() converts the string to a four-character code based on how the string sounds when spoken.

Soundex bu stringlarni 4 characterli codega ozgartiadi uni ingliz tiliga aytilishi bo’yicha sort qiladi.

select SOUNDEX('Sardor'), SOUNDEX('Sarvar') result >>> S636 S616 (Soundex nimaligiga example)

select DIFFERENCE('Sardor', 'Sarvar') result >>> 3

FORMAT

FORMAT(value, format, culture) – Format bilan sozlarni formatini ozgartirsa bo’ladi.

select FORMAT(3241332, '###-##-##') result >>> 324-13-32

LEFT

LEFT(string, number\_of\_chars) – left function xuddi exceldagi leftdak ishlaydi, birinchi nmadan olishni orsatiladi keyin nechta character olinishi.

select LEFT('SardorIsmatovAge22', 6) result >>> Sardor

RIGHT

RIGHT(string, number\_of\_chars) – o’ng tomondan ko’rsatilgan number\_of\_chars ga qarab oladi.

select RIGHT('SardorIsmatovAge22', 6) result >>> vAge22

SUBSTRING()

SUBSTRING(string, start, length) – bu xuddi mid ga o’xshab ishlaydi. Birinchi string beriladi, boshlanish positioni, uzunligi.

select substring('IsmatovSardorUmidjonO''gli', 8, 6) >>>> Sardor

LEN

LEN(string) – bu function string ni length ini o’lchab beradi. Len stringni oxirida kelgan space larni o’lchamaydi, lekin boshida kelganlarini o’lchaydi. If *string* is NULL, it returns NULL

select LEN(' Sardor') result >>>> 8

LOWER

LOWER(text) – string ni kichik harflar qilib beradi.

select lower('SaRdOr') >>>> result : sardor

UPPER

UPPER(text) - string ni katta harflar qilib beradi

select upper('SaRdOr') >>>> result : SARDOR

TRIM()

TRIM([characters FROM ]string) – trim function bo’sh space larni olib tashlaydi.

select TRIM(' Sardor ') >>>> result: Sardor

SELECT TRIM('#! ' FROM '    #SQL Tutorial!    ')  shuningdek specific character ni ham remove qilsa bo’larkan, faqat SQL server daqilib bo’,askan trim bilan.

LTRIM()

LTRIM(string) – stringni chap tarafidagi, ya’ni oldidagi space larni o’chiradi.

RTIM()

RTRIM(string) – string ni o’ng tarafidagi, ya’ni oxiridagi spacelarni o’chiradi

REPLICATE()

REPLICATE(string, integer) – bu function string ni bir necha marta qayta qayta takrorlashga yordam beradi.

select REPLICATE('Sardor', 4) result: >>> SardorSardorSardorSardor

PATINDEX()

PATINDEX(%pattern%, string) – bu function wildcard lar bilan ishlatilinib, ma’lum bir string ichidagi patternning positionini topish uchun ishlatilinadi. If not found returns 0

Other wildcards can be used in pattern, such as:

* % - Match any string of any length (including 0 length)
* \_ - Match one single character
* [] - Match any characters in the brackets, e.g. [xyz]
* [^] - Match any character not in the brackets, e.g. [^xyz]

*SELECT PATINDEX('%s.com%', 'W3Schools.com')* >>> result 9

*SELECT PATINDEX('%lll%', 'W3Schools.com') >>> result 0*

LEN() – length ni hisoblaydigan command. Agarda bosh joy tashlansa boshidakini hisoblidi oxirini hisoblamidi.

Datalength()- bu esa hamma spacelarni hisoblaydi.

REPLACE()

REPLACE(string, old\_string, new\_string) – bu function bilan, stringni ichidagi ma’lum bir characterlar ni o’zgartira olamiz. Old\_string o’zgartiriladigan string. New\_string yangi kelayotgan strin.

select REPLACE('Sardor Ismatov', 'a' , 'i') result >>>> Sirdor Ismitov

REVERSE()

REVERSE(string) – string ni teskarisiga yozib beradi.

select REVERSE('SardorSardorSardorSardor') result >>>> rodraSrodraSrodraSrodraS

QUOTENAME()

QUOTENAME(string, quote\_char) – bu berilgan stringni default holatida [] ni ichiga olib chiqarib beradi, yoki quote\_char ni o’zimiz ko’rsatsak ham bo’ladi.

SELECT QUOTENAME('abcdef') result >>>> [abcdef]

quote\_char - Optional. A one-character string to use as the delimiter. Can be a single quotation mark ( ' ), a left or right bracket ( [] ), a double quotation mark ( " ), a left or right parenthesis ( () ), a greater than or less than sign ( >< ), a left or right brace ( {} ) or a backtick ( ` ). If quote\_char is not specified, brackets are used.

SELECT QUOTENAME('abcdef', '()')

SPACE()

SPACE(number) – bu function kiritilgan raqam uzunligidagi space larni chiqarib beradi.

select SPACE(10) result: >>>>

STR()

STR(number, length, decimals) - The STR() function returns a number as a string.

STUFF()

STUFF(string, start, length, new\_string) – stuff ko’rsatilgan start poiontdan, ko’rsatilgan uzunlikdagi stringni o’chirib, o’rniga ko’rsatilgan yangi stringni qo’yadi. Bu yerda start – delete qilish positioni, length – nechta chars ni delete qilish uzunligi, new\_string kelayotgan string.

select STUFF('IsmatovSardorUmidjonO''gli', 8,6,'Samir') result: >>> IsmatovSamirUmidjonO'gli

TRANSLATE

TRANSLATE(string, characters, translations) – berilgan stringdagi malum characterlarni malum bir characterlarga o’tkazadi. The TRANSLATE() function 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.

SELECT TRANSLATE('3\*[2+1]/{8-4}', '[]{}', '()()'); >>>> Results in 3\*(2+1)/(8-4)

STRINGG\_AGG

Bu function butun column ni qo’shadi.

View – bu codeni saxranit qilish uchun ishlatilinadi.

Create view as ….. code yoziladi ---- yaratish uchun

Alter view as ….. ---- code ni o’zgartirish uchun

**Temp tables**

Ikki xil shaklda bo’ladi. Local(#) and global(##).

Temporary table bu faqat osha query ichida ishlaydigan table lar:

# va ## bilan create qilingan table lar shularga kiradi.

# bilan create qilinsa faqatgina osha sessiyada turadi kyngi sessiyaga otilsa auto ochib ketadi.

## bilan create qilingan bo’lsa esa global sessiyalarda ishlaydi faqat ishlatmiy qoyilsa ochib ketadi.

**View**

View da code saqlab qoyiladi. Va select \* from view\_name bilan retrive qilsa bo’ladi.

Create qilish: create view *(view\_name)* as *…………code…………*

Ozgartirish code ni: Ater view *view\_name* as …code…

**Trigger**

Triggerlar biror bir holat yuzaga kelganda unga code auto apply qilish uchun ishlatilinadi.

Trigger qanday create qilinadi?

Create trigger *triggername* on *Table\_name*

After insert

As

Begin

Update table\_name

Set update = ‘nimaga ozgartirayotganimiz’

Where column\_name is null or ‘’

End

Shu misoldagi kabi create qilinadi.

**Table variables**

Ozi agar bitta variable yasayotganda:

Declare @number as int = 10

Select @number

qilib yasardik.

Table ni ham shunday variable shaklida yasasa bo’ladi.

Declare @mytable as table(firstname varchar(20), lastname varchar(20))

Insert into @mytable values (‘Rocky’, ‘Balboa’), (‘Geremy’, ‘Doku’)

Select \* from @mytable

**Window functions**

Window fuctions rowlarga agregat funksiyalar berish va rank berish uchun ishlatilinadi, group by qilmasdan agregat berish imkoniyatini yaratadi, bizga.

--Aggregate window functions

Sum(), min(), max()

--Rank window functions

Row\_number() - rowlari raqamlab chiqadi.

Rank() - ham ularga ran bergandek belgilab chiqadi va bir xil qiymatliklarni bir xil son bilan rank qiladi. Va bir xil raqam bolb qolgan bosayam osha raqamni tashlab ketadi

Dense\_rank() – esa xuddi chempionatda orin bergandek ishlaydi.

Partition by beramiz over ni ichiga shunda u group by qilib ishlaydi, Masalan department boyicha sum\_salary hisoblamoqchi bolsak, partition by dep qilsa departament boyicha salary ni hisoblb beradi.

Ntile() – bu esa rowlarni ichida berilgan son boyicha bolib beradi.

--Value Window Functions

-lead - ozidan pasdaki valueni tortib beradi boshqa rowga berilgan offset bo’yicha

-log - esa ozidan tepadaki value ni tortib beradi.

lead(salary, 1) over (order by (select null)) lead,

lag(salary, 1) over (order by (select null)) lag from employee.employees