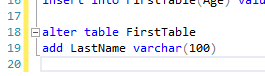
**Structed Query Language**

DDL- data defination language

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

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

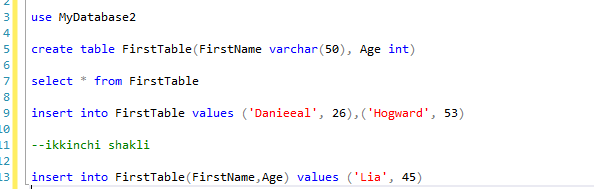
**Drop –** drop butunlay table ni ochirib yuboradi

**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:

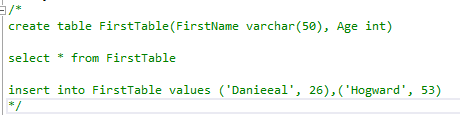
  
--varchar ni yonidagi son bu column name ga kiritiladiga max harflar

Comment qilish va olib tashlash

CTRL+K+C **comment**

CTRL+K+U **uncomment**

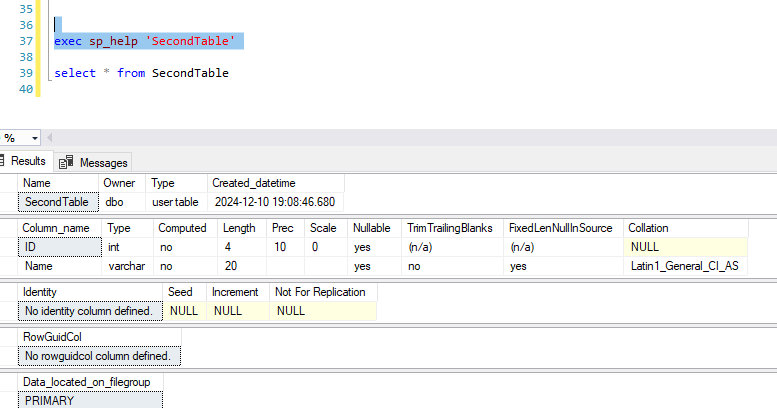
---keyin bir necha rowlarni birdan comment qilish uchun /\* dan foydalanamiz ya’ni qayerda boshlansa /\* tugash qatoriga esa \*/ ni qo’yamiz. Masalan:



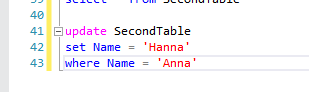
DDL- yani data definition language bilan table ni o’zini ozgartirish uchun ishlatilinadi

**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:

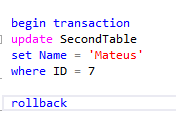
**How to drop column**

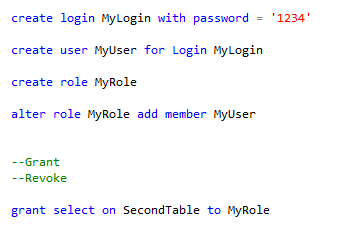
alter bilan qaysi table ligi ko’rsatiladi va drop column qilib column name yoziladi.

**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)

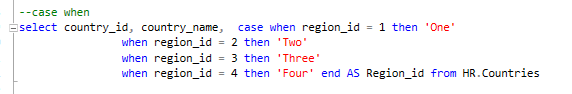
**Union –** bunda esa by default duplicatlarni olib tashlab, asc order qilib chiqarib beradi.

**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:

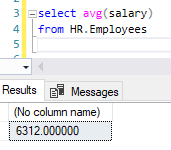
select country\_id, country\_name, IIF(region\_id = 1, 'One',

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

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

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

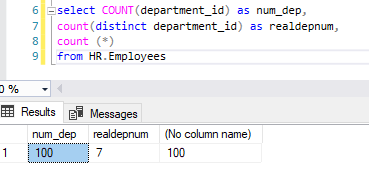
**aggregate functions**

Avg(column\_name)

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

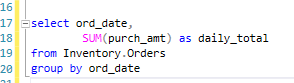
from HR.Employees

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.

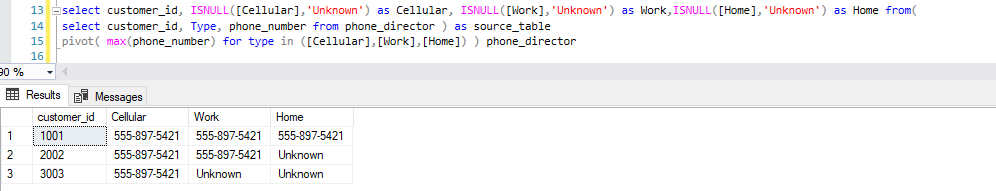


Sum ga misol:

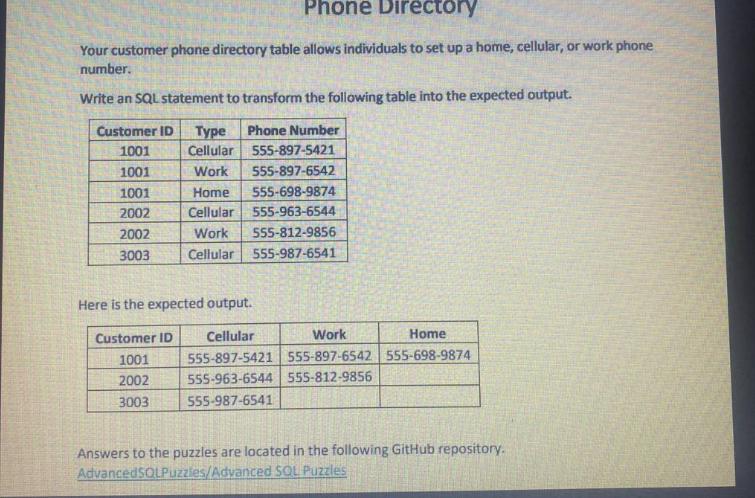
Sum, max ya’ni agregation lar bn group by ni ishlatish kk chunki bular bittadan olrtiq bo’lgan valuelarni hisoblash uchun ishlatilinadi. Misolda ham group\_date bilan har birk un ajratilib o’sha kunlarning Sumi hisoblangan.



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:**

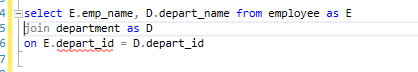
1. **FROM**:
   * The database engine starts by identifying the tables or views from which data will be retrieved.
2. **WHERE**:
   * Filters rows based on the specified condition(s). Only rows meeting these criteria proceed to the next step.(Where esa table ni o’ziga filter beradi)
3. **GROUP BY**:
   * 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**

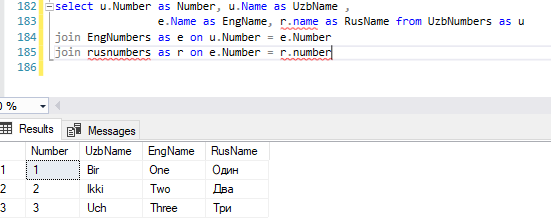
**Fact table** - bu measurable ya’ni hisob-kitoblar va analizlar olib boriladigan table bo’lib doim o’zgarib turadi va foreign keylar joylashadi asosan.

**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

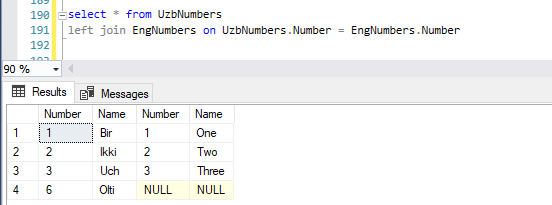
**Joins**

Misol:

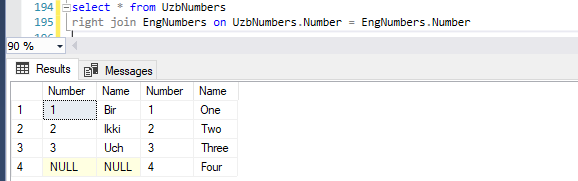
--Inner join esa ikkala table dagi bir biriga mos bo’lgan rowlarni qo’shadi. Misol:



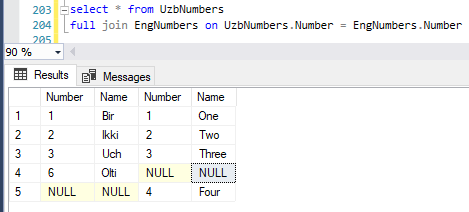
--Left join bu chap tarafdagi tabledagi malumotlarni o’zida saqlaydi, va kelayotgan tabledagi boshqa ma’lumotni kirgizmaydi.

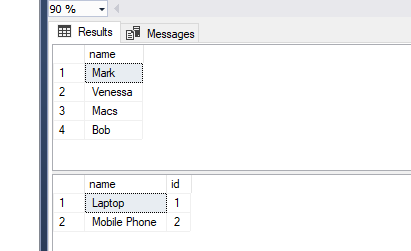


–right join esa shuni teskarisi ya’ni kelayotgan tabledagi barcha ma’lumotlarni chiqarib, qabul qilayotgandagini mosini oladi.



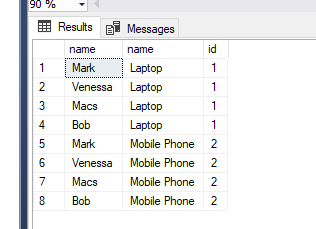
-–full join

esa bor hamma narsani qo’shib tashlaydi.

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

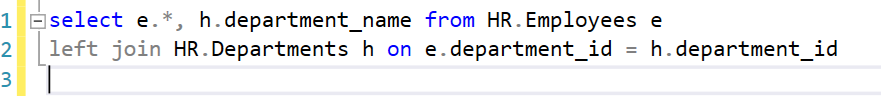
Bizda shunaqa ikkita table bor edi.

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 = --cross join

--outer apply = --left join

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

**Data and time functions**

Getdate()

**Mathematical functions**

**Floor()**

**Pi()**

**Power()**