

<https://www.hackerrank.com/challenges/weather-observation-station-12/problem>
<https://leetcode.com/denizp/>
https://web.kamihq.com/web/viewer.html?state=%7B%22ids%22%3A%5B%221n_HFQunbolFGSuHykem7_yblcJbbzFyU%22%5D%2C%22action%22%3A%22open%22%2C%22userId%22%3A%22107531782040487277606%22%2C%22resourceKeys%22%3A%7B%7D%7D&kami_user_id=12540471

<https://sqliteonline.com/>

MS SQL

SQL(Structured Query Language), veritabanındaki verileri yönettiğimiz yazılım dilleridir. MS SQL, Oracle, MySQL en çok kullanılan SQL dilleridir. Bu yazımda en temel SQL komutlarından başlayıp ileri seviye sorgular yazarak sizlere SQL hakkında en gerekli bilgileri vermeyi hedeflemekteyim.

1. SELECT

Select komutuyla veri tabanındaki veriyi çekebiliriz. Veri tabanındaki hangi tablodan bu verinin çekileceğini ise FROM'dan sonra belirtmeliyiz.

	A	B	C	D	E
1	CUSTOMER				
2	CustomerName	City	District	Birthdate	Gender
3	Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00:00.000	E
4	Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00:00.000	E
5	Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00:00.000	K
6	Özkan DERİLİOĞLU	Kütahya	Altıntaş	1996-06-02 00:00:00.000	E
7	Can DOLAR	Aksaray	Ağaçören	1986-05-20 00:00:00.000	E
8	Haydar DEMİRKAPI	Yozgat	Sorgun	1944-09-12 00:00:00.000	E

Resim 1

Örneğin Resim 1'deki 'CUSTOMER' tablosundaki tüm şehirleri görüntülemek istersek şu komutu yazmalıyız;

>> **SELECT** City **FROM** CUSTOMER

sonuç;

City
Kütahya
Adana
Van
Kütahya
Aksaray
Yozgat

Tablonun tamamini görüntülemek istersek;
>> SELECT * FROM CUSTOMER

2. SELECT INTO

SELECT INTO yapisi ile daha cok tablolarin kopyasini olusturmak amacli kullaniriz.

Eger 'CUSTOMER' tablosunun bir kopyasini cekmek istiyorsak;
>> SELECT * INTO MusteriKopya FROM CUSTOMER

3. WHERE

Where komutu ile bir kosul belirleyerek sadece istedigimiz kosulu saglayan degerleri dondurebiliriz.

Eger CUSTOMER tablosundaki musterilerden, dogum gunu '1944-09-12' olan kisilerin isim ve sehir bilgilerini cekmek istersek;
>> SELECT CustomerName, City
FROM CUSTOMER
WHERE Birthdate = '1944-09-12'

sonuc;

CustomerName	City
Haydar DEMİRKAPI	Yozgat

4. BETWEEN, AND, OR, NOT

Eger CUSTOMER tablosundaki kisilerden dogum gunu '1967-01-01' ile '2000-01-03' arasinda olan kisilerin isim ve dogum tarihi bilgilerini secmek istersek;
>> SELECT *
FROM CUSTOMER
WHERE Birthdate BETWEEN '1967-01-01' AND '2000-01-03'

sonuc;

CustomerName	Birthdate
Azad ÖNÜR	1989-03-23 00:00:00.000
Sude KAMURAN	1967-12-23 00:00:00.000
Özkan DERİLİOĞLU	1996-06-02 00:00:00.000
Can DOLAR	1986-05-20 00:00:00.000

5. DISTINCT

Distinct komutu ile tabloda birden fazla olan veriler birer kez gosterilir. Ornegin CUSTOMER tablosunda 'Kutahya' sehri iki kere geciyorken biz distinct kullanarak secim yaparsak tekrarlayan verileri teke dusurecektir;

```
>> SELECT DISTINCT City
      FROM CUSTOMER
```

sonuc;

City
Kütahya
Adana
Van
Aksaray
Yozgat

6. SELECT TOP

Select Top komutu ile seçmek istedigimiz ilk n veriyi çekebiliriz.

Eger CUSTOMER tablosundaki ilk 3 veriyi seçmek istiyorsak;

```
>> SELECT TOP 3*
      FROM CUSTOMER
```

sonuc;

CustomerName	City	District	Birthdate	Gender
Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00:00.000	E
Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00:00.000	E
Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00:00.000	K

7. INSERT INTO

Tabloda var olan bir sutuna yeni bir veri eklemek icin INSERT INTO komutu kullanilir.

Eger CUSTOMER tablosuna ismi 'Deniz Pinar' ve yasadigi sehir 'Istanbul' olan yeni bir kisi eklenmek istenirse;

```
>> INSERT INTO CUSTOMER([CustomerName], [City])
      VALUES ('Deniz Pinar', 'Istanbul')
```

sonuc;

CustomerName	City	District	Birthdate	Gender
Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00:00.000	E
Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00:00.000	E
Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00:00.000	K
Özkan DERİLİOĞLU	Kütahya	Altıntaş	1996-06-02 00:00:00.000	E
Can DOLAR	Aksaray	Ağaçören	1986-05-20 00:00:00.000	E
Haydar DEMİRKAPI	Yozgat	Sorgun	1944-09-12 00:00:00.000	E
Deniz Pinar	Istanbul	null	null	null

8. TRUNCATE TABLE

Tablonun ilk olusturulduđu halini bozmadan sadece icindeki verilerin bosaltilmasini saglar.

Ornegin;

```
>> TRUNCATE TABLE CUSTOMER
```

sonuc;

CustomerName	City	District	Birthdate	Gender

9. UPDATE, SET

Veritabaninda bir duzenleme veya guncelleme yapilmasi istendiginde UPDATE komutu kullanilir.

Eger 'Can Dolar' in yasadigi sehri Istanbul olarak guncellemek istersek;

```
>> UPDATE CUSTOMER
```

```
SET City = 'Istanbul'
```

```
WHERE CustomerName = 'Can Dolar'
```

sonuc;

CustomerName	City	District	Birthdate	Gender
Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00	E
Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00	E
Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00	K
Özkan DERİLİOĞLU	Kütahya	Altıntaş	1996-06-02 00:00	E
Can DOLAR	Istanbul	Ağaçören	1986-05-20 00:00	E
Haydar DEMİRKAPI	Yozgat	Sorgun	1944-09-12 00:00	E

10. DELETE

Veri tabaninda bulunan belli alanlari silmek icin kullanilir. Truncate Table ile arasindaki fark ise ornegin tabloda verilen id otomatik artan bir veriyse veri delete komutuyla silinmesine ragmen aslinda tam silinmez, cunku id kaldigi yerden saymaya devam eder. Truncate table ile silindiginde ise tum veriler sifirlanir.

Eger CUSTOMER tablosunu silmek istiyorsak;

```
>> DELETE FROM COSTMER
```

yazmamiz yeterlidir.

11. ORDER BY

Order By ile veriyi istedigimiz sekilde siralayabiliriz.

```
>> SELECT *
      FROM CUSTOMER
      ORDER BY City;
```

sonuc;

CustomerName	City	District	Birthdate	Gender
Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00:00.000	E
Can DOLAR	Istanbul	Ağaçören	1986-05-20 00:00:00.000	E
Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00:00.000	E
Özkan DERİLİOĞLU	Kütahya	Altıntaş	1996-06-02 00:00:00.000	E
Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00:00.000	K
Haydar DEMİRKAPI	Yozgat	Sorgun	1944-09-12 00:00:00.000	E

Goruldugu uzere City kendi icinde isim sirasina gore siralandi ve daha sonra bu siraya gore Customer tablosu siralandi.

```
>> SELECT *
      FROM CUSTOMER
      ORDER BY City DESC;
```

gibi yazmis olsaydik, azalan sekilde siralanirdi.

sonuc;

CustomerName	City	District	Birthdate	Gender
Haydar DEMİRKAPI	Yozgat	Sorgun	1944-09-12 00:00:00.000	E
Sude KAMURAN	Van	İpekyolu	1967-12-23 00:00:00.000	K
Özkan DERİLİOĞLU	Kütahya	Altıntaş	1996-06-02 00:00:00.000	E
Ayhan ÖZÇİL	Kütahya	Pazarlar	1963-11-08 00:00:00.000	E
Can DOLAR	Istanbul	Ağaçören	1986-05-20 00:00:00.000	E
Azad ÖNÜR	Adana	Karataş	1989-03-23 00:00:00.000	E

Eger artan sekilde siralamak isteseydik DESC yerine ASC yazardik. ORDER BY default olarak ASC calistirir.

12. HAVING

GROUP BY komutunu kullandigimizda genelde bir kosul belirtmek istedigimizde HAVING ifadesini kullaniriz. HAVING guplandirma islemi yapildiktan sonra kullanilirken WHERE daha guplandirma yapilmadan calistirilmalidir. Ornegin;

```
>>SELECT BirthDate, Gender
      FROM Customer
      GROUP BY BirthDate, Gender
      HAVING c4 = '1948-07-23 00:00:00.000'
```

sonuc;

Birthdate	Gender
K	1948-07-23 00:00:00.000

Goruldugu uzere HAVING ifadesini GROUP BY'dan sonra yazdik. Peki HAVING yerine WHERE kullanmak isteseydik nasil yazmaliydik;

```
>>SELECT BirthDate, Gender
FROM Customer
WHERE c4 = '1948-07-23 00:00:00.000'
GROUP BY BirthDate, Gender
```

13.SUBSTRING

Eger bir string icerisinden belli karakterleri cekmek istersek substring komutunu kullaniriz. Bu fonksiyon 3 parametre alır, once string sonra da istenen ciktinin ilk ve son indexleri yazilir.

```
>> SELECT SUBSTRING('deniz pinar', 1, 3)
sonuc;
'den'
```

14.CHARINDEX

Eger bir string icerisindeki karakterin indexini dondurmek istersek charindex komutunu kullanabiliriz. Bu fonksiyon da yine 3 parametre alır, once indexini dondurmek istedigimiz karakteri, sonra bu karakteri arayacagimiz stringi ve son olarak bu string icerisinde hangi indexten baslayarak arama yapacagimizi yaziyoruz.

```
>> SELECT CHARINDEX('p', 'deniz pinar', 1)
sonuc;
7
```

15.CONCAT

Stringleri birlestirmek icin kullanilir.

```
>> SELECT CONCAT('deniz', 'pinar')
sonuc;
'denizpinar'
```

16.SUM, MIN, MAX, AVG, COUNT

Bu fonksiyonlari asagida ornekler yaparak gosterecegim. Elimizde bir siniftaki ogrenciler ve notlarinin oldugu grades tablosu olsun.

Last name	First name	SSN	Test1	Test2	Test3	Test4	Final	Grade
Alfalfa	Aloysius	123-45-6789	40.0	90.0	100.0	83.0	49.0	D-
Alfred	University	123-12-1234	41.0	97.0	96.0	97.0	48.0	D+
Gerty	Gamma	567-89-0123	41.0	80.0	60.0	40.0	44.0	C
Android	Electric	087-65-4321	42.0	23.0	36.0	45.0	47.0	B-
Bumpkin	Fred	456-78-9012	43.0	78.0	88.0	77.0	45.0	A-
Rubble	Betty	234-56-7890	44.0	90.0	80.0	90.0	46.0	C-
Noshov	Cecil	345-67-8901	45.0	11.0	-1.0	4.0	43.0	F
Buff	Bif	632-79-9939	46.0	20.0	30.0	40.0	50.0	B+

grades tablosu

Eger bu tablodaki tum ogrencileri aldiklari final notlarına göre siralamak istersek;

```
>> SELECT *  
      FROM grades  
      ORDER BY Final
```

sonuc;

Last Name	First Name	SSN	Test1	Test2	Test3	Test4	Final	Grade
Noshow	Cecil	345-67-8901	45.0	11.0	-1.0	4.0	43.0	F
Gerty	Gramma	567-89-0123	41.0	80.0	60.0	40.0	44.0	C
Bumpkin	Fred	456-78-9012	43.0	78.0	88.0	77.0	45.0	A-
Rubble	Betty	234-56-7890	44.0	90.0	80.0	90.0	46.0	C-
Android	Electric	087-65-4321	42.0	23.0	36.0	45.0	47.0	B-
Alfred	University	123-12-1234	41.0	97.0	96.0	97.0	48.0	D+
Alfalfa	Aloysius	123-45-6789	40.0	90.0	100.0	83.0	49.0	D-
Buff	Bif	632-79-9939	46.0	20.0	30.0	40.0	50.0	B+

Eger grades tablosundaki ogrenci sayisini bulmak istiyorsak;

```
>> SELECT COUNT("First Name")  
      FROM grades
```

sonuc;

8

Eger Test2 sinavinda alinan en dusuk notu dondurmek isteseydik;

```
>> SELECT MIN(Test2)  
      FROM grades
```

sonuc;

11.0

Alinan en yuksek notu dondurmek isteseydik MIN yerine MAX yazmamiz yeterliydi.

Eger butun ogrencilerin Test1 sinavindan aldigi notların ortalamasını dondurmek istiyorsak;

```
>>SELECT AVG(test1)  
      FROM grades
```

sonuc;

42.75

Eger adi 'Bif Buff' olan ogrencinin Test1'den aldigi not ile Test2'den aldigi notun toplamini dondurmek istiyorsak;

```
>> SELECT SUM(test1) + SUM(test2)
      FROM grades
      WHERE firstname = 'Bif'
```

sonuc;
66

17.JOIN

Tablolari birlestirmek istedigimizde join fonksiyonlarini kullaniriz. Asagida grades ile AVG tablolari uzerinden join islemlerini gostermeye calisacagim.

! SSN	TestAVG	GeneralAVG
123-45-6789	78,25	63,625
123-12-1234	82,75	65,375
567-89-0123	55,25	49,625
087-65-4321	36,5	41,75
456-78-9012	71,5	58,25
234-56-7890	76	61
345-67-8901	15,25	29,125
632-79-9939	34	42
234-56-2890	45	62
345-67-3901	65	47
632-79-9439	24	65

AVG TABLOSU

!	LastN...	FirstName	SSN	Test1	Test2	Test3	Test4	Final	Grade
	Alfalfa	Aloysius	123-45-67...	40	90	100	83	49	D-
	Alfred	University	123-12-12...	41	97	96	97	48	D+
	Gerty	Gramma	567-89-01...	41	80	60	40	44	C
	Android	Electric	087-65-43...	42	23	36	45	47	B-
	Bumpkin	Fred	456-78-90...	43	78	88	77	45	A-
	Rubble	Betty	234-56-78...	44	90	80	90	46	C-
	Noshow	Cecil	345-67-89...	45	11	-1	4	43	F
	Buff	Bif	632-79-99...	46	20	30	40	50	B+

GRADES TABLOSU

a. INNER JOIN

İki tablonun kesişimi olan değerleri çekmek istediğimizde INNER JOIN kullanırız. Örneğin ben grades tablosundaki SSN'lere göre AVG tablosundaki kesilen değerleri çekmek istersem;

```
>> SELECT *
      FROM GRADES
      INNER JOIN AVG ON GRADES.SSN = AVG.SSN
```

sonuc;

!	Las...	FirstN...	SSN	Test1	Test2	Test3	Test4	Final	Grade	TestA...	GeneralAVG
	Alfalfa	Aloysius	BLOB	40	90	100	83	49	D-	78,25	63,625
	Alfred	Univer...	BLOB	41	97	96	97	48	D+	82,75	65,375
	Gerty	Gramma	BLOB	41	80	60	40	44	C	55,25	49,625
	Android	Electric	BLOB	42	23	36	45	47	B-	36,5	41,75
	Bumpkin	Fred	BLOB	43	78	88	77	45	A-	71,5	58,25
	Rubble	Betty	BLOB	44	90	80	90	46	C-	76	61
	Noshow	Cecil	BLOB	45	11	-1	4	43	F	15,25	29,125
	Buff	Bif	BLOB	46	20	30	40	50	B+	34	42

Yukarıda SSN'leri ortak olan tüm değerler iki tablodan çekildi ve birleştirildi. Şimdi bir de AVG tablosuna SSN'leri ortak olan GRADES tablosunu ekleyelim;

```
>> SELECT *
      FROM AVG
      INNER JOIN GRADES ON AVG.SSN = GRADES.SSN
```

sonuc;

SSN	TestA...	Gener...	LastN...	FirstN...	Test1	Test2	Test3	Test4	Final	Grade
BLOB	78,25	63,625	Alfalfa	Aloysius	40	90	100	83	49	D-
BLOB	82,75	65,375	Alfred	Univer...	41	97	96	97	48	D+
BLOB	55,25	49,625	Gerty	Gramma	41	80	60	40	44	C
BLOB	36,5	41,75	Android	Electric	42	23	36	45	47	B-
BLOB	71,5	58,25	Bumpkin	Fred	43	78	88	77	45	A-
BLOB	76	61	Rubble	Betty	44	90	80	90	46	C-
BLOB	15,25	29,125	Noshow	Cecil	45	11	-1	4	43	F
BLOB	34	42	Buff	Bif	46	20	30	40	50	B+

Goruldugu uzere iki tabloda ortak olan SSN degerlerini kullanarak iki tabloyu birbirine birlestirdik. Iste bu ortak olan degerleri join etme terimi bize INNER JOIN'i isaret etmektedir. Kod kismina yazdigimiz INNER ifadesini kullanmadan da yine ayni islemleri yapabiliriz.

b. LEFT JOIN

Ornegin Grades tablosuna AVG tablosunu LEFT JOIN ile eklemek ve Test1 degerlerini dondurmek istersek, bize Grades'deki tum degerleri verirken AVG'deki sadece kesisen(Grades'de karsiligi olan) degerler eklenir.

```
>>SELECT AVG.TestAVG, GRADES.LastName
FROM AVG
LEFT JOIN GRADES ON GRADES.SSN = AVG.SSN
```

sonuc;

testavg	lastname
78,25	Alfalfa
82,75	Alfred
55,25	Gerty
36,5	Android
71,5	Bumpkin
76	Rubble
15,25	Noshow
34	Buff
45	NULL
65	NULL
24	NULL

Goruldugu uzere AVG'deki tum degerleri aldi fakat Grades'deki karsiligi olanlari dondurdu. Karsiligi olmayanlar ise NULL seklinde geldi. Burada sol kume AVG, sag kume ise GRADES olmustur. Boylece soldaki tum degerler alinmisken, sagdaki kume icin sadece karsiligi olan degerler secilmistir.

c. **RIGHT JOIN**

Yukarida left join icin yaptigimiz islemin aynisini kumleleri yer degistirerek yaparsak o zaman da RIGHT JOIN yapmis oluruz. Yukaridaki ornegin ciktilisiyla ayni ciktiliyi dondurecek kodu bu sefer RIGHT JOIN ile yazalim;

```
>> SELECT AVG.TestAVG, GRADES.LastName  
      FROM GRADES  
      RIGHT JOIN AVG ON GRADES.SSN = AVG.SSN
```

Bu sefer sag kume AVG iken, sol kume GRADES oldu. Yine AVG'deki tum elemanlar geldi fakat GRADES'te karsiligi olan soyadlar cekildi. Karsiligi olmayan alanlar ise otomatikmen NULL ile dolduruldu.

d. **FULL JOIN**

Eger iki kumedeki tum degerlerin gelmesini istiyorsak yani aslinda iki kumenin birlesimini dondurmek istiyorsak FULL JOIN metodunu kullaniriz. Ornegin yukaridaki ornek ile ayni ciktiliyi dondurmek icin asagidaki iki sorgu da uygundur;

```
>> SELECT AVG.TestAVG, GRADES.LastName  
      FROM GRADES  
      FULL JOIN AVG ON GRADES.SSN = AVG.SSN;
```

```
>> SELECT AVG.TestAVG, GRADES.LastName  
      FROM AVG  
      FULL JOIN GRADES ON GRADES.SSN = AVG.SSN;
```

Bu iki sorgu arasindaki tek fark sag ve sol kumlerin yerlerinin degismesidir. Fakat sonuc olarak iki sorgu da ayni sonucu vermektedir. FULL JOIN iki kume arasinda birlesim islemi yaptigi icin sag ve sol kumenin hangi kumeler oldugu onemini kaybeder ve sonuc ayni olur.

18. **TRIM**

String üzerinde bazı silme ve düzenlemeler yapmamızı sağlar. Ornegin verilen stringin hem sagindaki hem de solundaki bosluklari silinmesini istiyorsak;

```
>> SELECT TRIM('      deniz pinar      ')
```

sonuc;

deniz

pinar

Goruldugu uzere yukaridaki stringin sag ve solundaki bosluklari temizledik ancak ortadaki bosluga dokunmadik. Ayni zamanda TRIM ile istedigimiz herhangi bir sey stringden silinmesini saglayabiliriz. Ornegin asagida verilen stringden 'd' harfini cikarmak icin TRIM fonksiyonun nasil kullanilmasi gerektigi aciklanmistir;

```
>> SELECT TRIM('d' FROM 'deniz pinar')
```

sonuc;

eniz
pinar

Bir ornek daha, eger 'pinar' ifadesinin silinmesini istersek;

```
>> SELECT TRIM('pinar' FROM 'deniz pinar')
```

sonuc;

deniz

19. LTRIM

Bu fonksiyonla dizinin sol tarafındaki bosluklari yok edebiliriz.

```
>> SELECT LTRIM('   deniz pinar   ')
```

sonuc;

deniz pinar

20. RTRIM

Bu fonksiyonla dizinin sag tarafındaki bosluklari yok edebiliriz.

```
>> SELECT RTRIM('   deniz pinar   ')
```

sonuc;

deniz
pinar

21. LOWER

Bu fonksiyon dizinin tum harflerini kucuk harfe cevirmeyi saglar.

```
>> SELECT LOWER('DENİZ PINAR')
```

sonuc;

deniz pinar

22. UPPER

Bu fonksiyon dizinin tüm elemanlarını büyük harfe dönüştürür.

```
>> SELECT UPPER('deniz pinar')
```

sonuc;

DENİZ PINAR

23. REVERSE

Bu fonksiyon ile diziyi tersten yazdırabiliriz.

```
>> SELECT REVERSE('deniz')
```

sonuc;

zine d

24. REPLICATE

Verilen ifadenin istenilen sayıda tekrarlanmasını sağlar.

```
>> SELECT REPLICATE('la',3)
```

sonuc;

lalal a

25. CONCAT

Concat ile birden fazla diziye birleştirebiliriz.

```
>> SELECT CONCAT('DENİZ', 'PINAR', 'SQL', 'OGRENIYOR')
```

sonuc;

DENİZ PINAR SQL OGRENIYOR

26. CONCAT_WS

Peki eğer verdiğimiz her parametreyi birleştirirken araya istediğimiz bir karakter veya dizi koymak istersek? İşte burada CONCAT_WS fonksiyonu imdadımıza yetiyor.

```
>> SELECT CONCAT_WS('**', 'deniz', 'pinar')
```

sonuc;

deniz**pin ar

27. FORMAT

Format ile veriyi sayı veya tarih gibi değişik türlere dönüştürebiliriz. 3 parametre alır fakat 2 parametreyle de kullanılabilir. İlk parametre tarih değeridir. İkinci parametre formatlama şeklini belirttiğimiz parametredir. Son olarak üçüncü parametre olarak da culture parametresini giriyoruz.

Örneğin GETDATE() ile bugünün tarihini çekelim ve istediğimiz tarih formatında görüntüleyelim;

```
>> SELECT FORMAT(GETDATE(), 'd', 'en-us')
```

sonuc;

4/18/202 1

Şimdi FORMAT fonksiyonunu iki parametre ile kullanalım;

```
>> SELECT FORMAT(GETDATE(), 'd/M/y')
```

sonuc;

18/4/2 1

Yukarıda kullandığımız bazı kısaltmaların ne anlama geldiğine bakalım;

d = day

M = month

m = minute

y = year

Ben burada FORMAT fonksiyonu ile ilgili özet bilgi vermeye çalıştım ama aslında FORMAT fonksiyonunun birçok farklı kullanımları vardır. Dilerseniz Microsoft SQL SERVER sitesinden daha detaylı bilgi alabilirsiniz.

28. LEFT

Eğer bir string içerisinde belli bir karakterin solunda kalan kısmı dondurmek istiyorsak LEFT komutunu kullanabiliriz. LEFT komutu iki parametre alır. İlk parametresi veriyi çekmek istediğimiz ana dizi, ikinci parametre ise index tir. Bu index'in solunda kalan kısım çekilecektir. Örneğin;

```
>> SELECT LEFT('deniz pinar', 5)  
sonuc;
```

deni z

```
>> SELECT LEFT('deniz pinar', CHARINDEX(' ', 'deniz pinar'))
```

Yukarıdaki kod ile de aynı sonucu dondurebiliriz. CHARINDEX ile aslında 'deniz' ile 'pinar' arasındaki boşluğun indexini dondurmış olduk. LEFT ile de bu indexin solundaki ifadeyi dondurmış olduk.

29. RIGHT

Eğer bir string içerisinde belli bir karakterin sağında kalan kısmı dondurmek istiyorsak RIGHT komutunu kullanabiliriz. RIGHT komutu iki parametre alır. İlk parametresi veriyi çekmek istediğimiz ana dizi, ikinci parametre ise index tir. Bu index'in sağında kalan kısım çekilecektir. Örneğin;

```
>> SELECT RIGTH('deniz pinar', 6)  
sonuc;
```

pina r

```
>> SELECT RIGHT('deniz pinar', LEN('deniz pinar') - CHARINDEX(' ', 'deniz pinar'))
```

Yukarıdaki kod ile de aynı sonucu dondurebilirdik. Eğer indexi bilmiyor ve programın bulmasını istiyorsak bu yöntem işi çözmektedir.

30. UPDATE

Veri tabanındaki herhangi bir veriyi güncellemek veya değiştirmek için update komutunu kullanırız. Eğer bütün tabloyu güncellemek istemiyorsak UPDATE komutunu genelde UPDATE ... SET ... WHERE ... şeklinde kullanırız. UPDATE' ten sonra güncelleme yapmak istediğimiz tablo ismini, SET'ten sonra tablo içerisinde nasıl bir güncelleme yapmak istediğimizi, WHERE'den sonra ise hangi koşullarda bu güncellemeyi yapmak istediğimizi belirtiyoruz.

Örneğin GRADES tablosundaki final notlarının 50'nin altında olanları 1 olacak şekilde güncelleyelim;

```
>> UPDATE GRADES SET Final = 1 WHERE Final < 50;  
      SELECT * FROM GRADES
```

sonuç;

	Last_...	First_na...	SSN	Test1	Test2	Test3	Test4	Final	Grade
	Alfalfa	Aloysius	123-45-67...	40	90	100	83	1	D-
	Alfred	University	123-12-12...	41	97	96	97	1	D+
	Gerty	Gamma	567-89-01...	41	80	60	40	1	C
	Android	Electric	087-65-43...	42	23	36	45	1	B-
	Bumpkin	Fred	456-78-90...	43	78	88	77	1	A-
	Rubble	Betty	234-56-78...	44	90	80	90	1	C-
	Noshow	Cecil	345-67-89...	45	11	-1	4	1	F
	Buff	Bif	632-79-99...	46	20	30	40	50	B+

31. DECLARE / SET

Declare ile değişken tanımlayabiliriz, SET ile de bu değişkene bir değer atayabiliriz. Declare'den sonra başına '@' koyarak değişken ismini, sonra da bu değişkenin tipini tanımlarız. MSSQL'de değişkenler başlarına '@' isareti konarak ifade edilirler. Örneğin 'cumle' diye bir değişken oluşturup bunu 'SQL Öğreniyorum' şeklinde bir dizi olarak tanımlayalım;

```
>> DECLARE @cumle AS nvarchar(11)  
      SET @cumle='deniz pinar'  
      SELECT @cumle
```


sonuc;

deniz
pinar

Yukarida nvarchar veri tipini kullanarak islem yaptik. nvarchar, girilen veri kadar yer kaplayan, en fazla 4000 karakter veri tutan, max depolama boyutu ise 2 GB olan bir veri tipidir. SQL dilinde kullanılan bir çok veri turu vardır. Asagida dizi veri tipleri örnek olarak verilmiştir;

Character Strings SQL Server Data types

Data Type	Lower Range	Upper Range	Storage	Remarks
Char(n)	0 characters	8000 characters	N bytes	1. It provides a fixed-width character data type.
Varchar(n)	0 characters	8000 characters	n bytes + 2 bytes	1. It is a variable length character data type. 2. N defines the string size.
Varchar(max)	0 characters	2 ³¹ chars	n bytes + 2 bytes ~ 2 GB	We should avoid using this data type unless required due to its huge storage requirement.
Text	0 chars	2,147,483,647 chars	n bytes + 4 bytes	1. It is a variable-length character data type. 2. We should avoid using this data type as it might get deprecated in future versions of SQL Server.

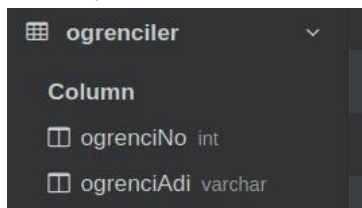
kaynak= <https://www.sqlshack.com/an-overview-of-sql-server-data-types/>

32. CREATE TABLE

Veri tabanı içerisinde yeni bir tablo oluşturmak istersek bu komutu kullanırız. Örneğin 'ogrenciler' diye bir tablo oluşturalım ve bu tablonun içerisinde öğrencilerin okul numaralarının bulunduğu 'ogrenciNo' ve öğrencilerin isimlerinin bulunduğu 'ogrenciAdi' sütunları olsun.

```
>> CREATE TABLE ogrenciler(  
        ogrenciNo int,  
        ogrenciAdi varchar(50)  
)
```

sonuc;



Column
ogrenciNo int
ogrenciAdi varchar

CREATE TABLE komutundan sonra olusturmak istedigimiz tablo adini girdik, daha sonra parantez acarak eklemek istedigimiz sutun isimlerini ve veri tiplerini girdik.

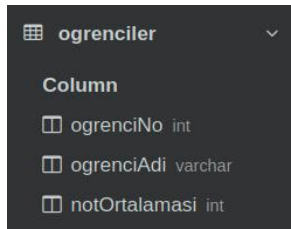
Goruldugu uzere veri tabaninda tablo olusturuldu ama ici bos bir tablo olusturuldu. Eger icerisine veri eklemek isterseniz daha once madde 7’de anlattigim kismi incelemenizi tavsiye ederim. INSERT INTO fonksiyonunu kullanarak tabloya veri ekleyebilirsiniz.

33. ALTER TABLE

Veri tabaninda bulunan herhangi bir tablonun ozelliklerini degistirmek istersek bu fonksiyonu kullaniriz. Ornegin yukarida olusturdugumuz ogrenciler tablosuna ‘notOrtalamasi’ adinda bir sutun daha eklemek istersek;

```
>> ALTER TABLE ogrenciler ADD notOrtalamasi int;
```

sonuc;

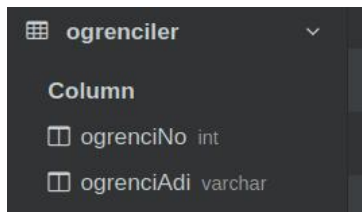


Column
ogrenciNo int
ogrenciAdi varchar
notOrtalamasi int

ALTER TABLE ile tablolarda istedigimiz degisikligi yapabilecegimizi soylemistik. Ornegin herhangi bir sutunu nasil sileriz?

```
>> ALTER TABLE ogrenciler DROP COLUMN notOrtalamasi;
```

sonuc;

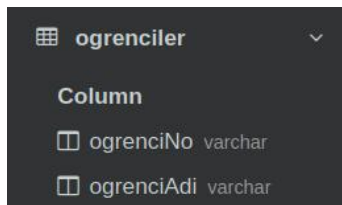


Column
ogrenciNo int
ogrenciAdi varchar

Peki var olan bir sutunun veri tipini degistirmek istersek;

```
>> ALTER TABLE ogrenciler ALTER COLUMN ogrenciNo varchar;
```

sonuc;



Column
ogrenciNo varchar
ogrenciAdi varchar

Peki 'ogrenciNo' nun primary key oldugunu varsayalım ve buraya mutlak bir degerin girilmesi gerektiği sartini koyalım;

```
>> ALTER TABLE ogrenciler ALTER COLUMN ogrenciNo int IS NOT NULL;
```

34. DROP TABLE

Tablo silmek için bu fonksiyonu kullanırız.

```
>> DROP TABLE Grades
```

şeklinde yazarsak Grades tablosunu tamamen kaldırmış oluruz.

```
>> DROP TABLE IF EXISTS Grades
```

şeklinde yazarsak Grades tablosu yoksa bile hata almadan eğer Grades tablosu varsa bu komutla silinir.

35. CREATE INDEX