

PostgreSQL

Öncelikle bu not kağıdının SQL'i öğrenmek için yeterli olmadığını söylemeliyim :) Hazırladığım bu notu "Veritabanı Yönetim Sistemleri" dersime çalışmak için hazırlamıştım. Doğal olarak eksiklerin bulunması ve yanlışlarımın olması durumunda kusura bakmayın :) İyi çalışmalar dilerim.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
3	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
4	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
5	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
6	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

! KULLANACAGIMIZ TABLO

! KULLANACAGIMIZ TABLOLARI OLUŞTURAN KOMUTLARI GÖRMEK İÇİN TIKLAYINIZ

▼ Tablonun Verilerini Getirme



SELECT yazdırılmasını istediğiniz sütün adı/adları ,FROM tablo adı

▼ Where



select * from staff **where** fname='Susan'

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003

Where kullanımındaki amac yazığımız kodların satırlarınıfiltrelemek istediğimiz tablo verilerini getirmektir. Yukarıdaki örnekte sadece fname'i susan olan kişi getirilecektir. Tırnak içine yazdıklarımız tabloda yazdıklarımızla aynı olmalıdır.(case sensitive vardır).

! Whereden sonra küme fonksiyonları kullanılmaz. O fonksiyonlar Select'ten sonra yazılarak kullanılabilir.

 select * from staff where fname='Susan',sum(salary)

 select count(fname) from staff where fname='Susan'

	count	bigint	lock
1		1	

▼ Group By

Group by komutu aynı değerlere sahip verileri grüplamamızı sağlar.

 select fname from staff **group by** fname

	fname
1	Susan
2	Mary
3	David
4	Ann
5	John
6	Julie

Group by'da önemli olan bir nokta vardır

1-) Group by'dan sonra gelen sütün ismi select ettiğimiz değer sadece küme fonksiyonlarıyla yazılabilir.

 select fname ,**avg(salary)** from staff group by fname

	fname character varying (10) 	avg numeric 
1	Susan	24000.00000
2	Mary	9000.000000
3	David	18000.00000
4	Ann	12000.00000
5	John	30000.00000
6	Julie	9000.000000

Yukarıda örnekte fname e göre gruptadığımızda her gruptadığımız kişinin KENDİ ORTALAMASINI GETİRİR. Yani kendi maaşını getirir. Ama bu bizim dikkat etmemiz gereken nokta değildir. Görüldüğü küme fonksiyonu kullandığımızda hata vermezken aşağıdaki kullanımlarda hata veriyor.

 select fname,lname, avg(salary)from staff group by fname

 select * from staff group by fname,

▼ Having

 select fname from staff group by fname **having** fname='Susan'

Having SADECE group by kullandığımızda yazdığımız filtreleme komutudur. Yani group by kullandığımızda where yerine having kullanırız. Where **satırları** filtrelerken having **grupları** filtreler.

❗️❗️ Where ve having farkı whereden sonra küme fonksiyonları kullanılmazken having'ten sonra kullanılabilir.



SELECT position, sum(salary) FROM staff where sex='F' GROUP BY position

Yukarıda ilk olarak pozisyonları grupladık daha sonra grupladığımız pozisyonlarda çalışan SADECE kadın olanların maaşlarının ortalamasını verdi.

	position character varying (10)	sum bigint
1	Assistant	30000
2	Manager	24000



SELECT position, sum(salary) FROM staff where sex='F' GROUP BY position **having** sum(salary)>25000

Yukarıda ise yaptığımız grupladığımız pozisyonlardaki kadınların maaş ortalamasının 25000 üstü olanların sadece getirilmesini istedik.

	position character varying (10)	sum bigint
1	Assistant	30000



SELECT position, sum(salary) FROM staff where sex='F' GROUP BY position **having** position like 'M%'

Yukarıda ise pozisyonları grupladık daha sonrasında m harfi ile başlayan pozisyonlardaki kadınların ortlama maaşlarını yazdırıldı.

	position character varying (10)	sum bigint
1	Manager	24000

▼ Order By

Tabloyu istediğimiz gibi sıralamamızı sağlar. Buna göre;

- ASC ⇒ küçükten büyüğe/A'den Z'ye sıralar
- DESC ⇒ büyükten küçüğe/Z'dan A'ye sıralar.

 **SELECT * from staff *order by* fname desc**

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
2	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
3	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005
4	SL21	John	White	Manager	M	1965-10-01	30000	B005
5	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
6	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003

 **SELECT * from staff *order by* fname, salary asc**

Aynı anda iki tabloya göre de sıralayabiliyoruz görüldüğü üzere.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
2	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
3	SL21	John	White	Manager	M	1965-10-01	30000	B005
4	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005
5	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
6	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003

▼ Distinct

Tabloda belirtilen sütundaki verilerde, eğer aynı veriden birden fazla varsa onlardan sadece birer örek alır.

 **SELECT *distinct* position from staff *order by* position asc**

	position character varying (10) 
1	Assistant
2	Manager
3	Supervisor

Eğer birden fazla sütunu distinctliyorsak iki tabloyu primary key olarak alır.

 `SELECT distinct sex,position from staff order by position asc`

Hem cinsiyet hem pozisyon'a göre filtrelediğimizde ikisini tek değer gibi alır.

	sex character (1) 	position character varying (10) 
1	F	Assistant
2	M	Manager
3	F	Manager
4	M	Supervisor

 `SELECT count(distinct position) from staff`

	count bigint 
1	3

Kaç farklı pozisyon türü olduğunu bulmamız için kullandığımız bir yöntem

▼ Alias

Alias yani “**as**”, herhangi bir sorguyu, sütunu adlandırmak için kullandığımız kelimedir. Türkçe karakter kullanmamaya özen gösteriniz.

 select sex **as** cinsiyet ,count(salary) **as** maxs from staff group by sex

Cinsiyete göre gruptadığımızda maaşların sayısını count etmiştir. Böylece her cinsiyetten kaç kişinin maaş aldığı görebiliriz.

Ayrıca sex sütunun adını cinsiyet olarak değiştirmemizi sağlar.

	cinsiyet character (1)	maxs bigint
1	M	2
2	F	4

▼ Between

Between operatörü iki aralıktaki değerleri aramak için kullanırız.

 select * from staff where salary **between** 1000 **and** 25000

Maaşı 1000 ve 25000 arasında olan personellerin tüm bilgisini getiriyor

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
2	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
3	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
4	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
5	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

 select * from staff where fname **between** 'D' **and** 'S'

İsimleri d ve s harflerinin arasındaki harflerden başlayan kişilerin bilgileri getirilir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
3	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
4	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

 select * from staff where fname **not between 'D' and 'S'**

İsimleri d ve s harflerinin arasında olmayan harflerden başlayan kişilerin bilgileri getirilir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
2	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003

▼ In

In operatörünü belirtilen sütunda birden fazla arama yaptığımız zaman kullanırız.

 select * from staff where position **in ('Manager','Assistant')**(parantez koymassak çalışmaz)

Müdür ve asistan olanların bilgilerini getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
3	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
4	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
5	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



select * from staff where position='Manager' or position='Assistant'

Yukarıdaki soru da aynı tabloyu bize verir

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
3	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
4	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
5	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



`select * from staff where position not in ('Manager','Assistant')`

Müdür ve asistan olmayanların bilgilerini getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003

▼ Like



`SELECT * FROM staff WHERE position LIKE 'M%'`

M harfi ile başlayan pozisyonların bilgilerini getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003



`SELECT * FROM staff WHERE position LIKE '%an%'`

İçinde(sağında,solunda,ortasında) an kelimesi geçen tüm pozisyon bilgilerini getirir. cmd/ctrl + gibi düşün.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
3	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
4	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
5	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005



`SELECT * FROM staff WHERE position LIKE '_a%'`

2. karakteri a olan pozisyonları getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003



`SELECT * FROM staff WHERE position LIKE 'A%t'`

A ile başlayıp t ile biten karakterleri getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
2	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
3	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

▼ Küme Fonksiyonları

Count	Belirli sütundaki değerlerin sayısını verir.
Sum	Belirli sütundaki değerlerin toplamını verir.
Avg	Belirli sütundaki değerlerin ortalamasını verir.
Min	Belirli sütundaki değerlerin min değerini verir.
Max	Belirli sütundaki değerlerin max değerini verir.

- Küme fonksiyonları sadece **select listesinde** ve **having cümlesiğinin içinde** bulunabilir.
- Her biri tablonun tek sütunu için filtreleme yapar.
- count, min, max: Sayısal/sayısal olmayan alanlarda kullanılır
- sum ,avg: Sadece sayısal alanlarda kullanılır.
- Count hariç tüm fonksiyonlar önce null olan değerleri elimine eder, count ise null'ı normal veri olarak sayar.



`SELECT count(salary), fname from staff`

	count bigint	
1	6	

! ! Having cümlesiğindeki sütun adlarının , group by listesinde görünmesi gereklidir veya bir küme fonksiyonu içinde kullanılması gereklidir

SELECT fname, position, sum(salary) FROM staff where sex='F' GROUP BY position, fname having position like 'M%' order by fname

	fname character varying (10)	position character varying (10)	sum bigint
1	Susan	Manager	24000

SELECT fname, position, sum(salary) FROM staff where sex='F' GROUP BY position having position like 'M%'

▼ Alt Sorgular

- Order By bir alt sorguda kullanılmaz.
- Alt sorguda select listesi, **EXIST** kullanan alt sorgular dışında, tek sütun veya tek ifadeden oluşur.
- Bir alt sorgunun yanında başka bir sorgu yazdığımızda **and** ile yazmamız takdirinde kod çalışacaktır.

Yukarıda yazdığımız sorguda herhangi bir erkek maaşından daha fazla kazanan kadının bilgilerini getirmesini istemiştik.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003

SELECT * FROM staff WHERE (salary > Some(SELECT salary FROM staff WHERE sex = 'M')) , sex='F';

▼ Some/Any

=ANY(iç sorgu) : Alt sorgudan elde edilen sonuçlardan herhangi birine eşit olan kayıtları ifade eder. IN ifadesi ile aynı işlemi gerçekleştirir.

>ANY(iç sorgu) : Alt sorgudan elde edilen sonuçlardan herhangi birinden büyük kayıtları ifade eder.

>=ANY(iç sorgu) : Alt sorgudan elde edilen sonuçlardan herhangi birine eşit veya büyük kayıtları ifade eder.

<ANY(iç sorgu): Alt sorgudan elde edilen sonuçlardan herhangi birinden küçük kayıtları ifade eder.

<=ANY(iç sorgu): Alt sorgudan elde edilen sonuçlardan herhangi birine eşit veya küçük kayıtları ifade eder.

!=ANY(iç sorgu) veya **<> ANY(iç sorgu)**: Alt sorgudan elde edilen sonuçlardan herhangi birinden farklı anlamı katar ancak bu kullanım mantıklı değildir. Çünkü Herhangi birinden farklı şekilde kullanıldığında tüm kayıtlar istisnásız gelecektir. Dönüş herhangi kayıttan birine bile eşit olsa diğerlerinden farklı olacağı için şartımız tüm kayıtları listeleyecektir.



SELECT * FROM staff WHERE salary > ANY(SELECT salary FROM staff WHERE sex = 'M');

Herhangi bir erkekten daha fazla kazananların bilgilerini getirir.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003

▼ All

=ALL(iç sorgu): Alt sorgudan elde edilen sonuçlardan hepsine eşit olan kayıtları ifade eder. Ancak bu şekilde kullanılan şart ifadesinde herhangi bir sonuç dönmeyecektir. Herhangi bir değerin tüm dönen sonuçlara eşit olması mümkün değildir. ALL ifadesi ile eşit (=) operatörü kullanılmaz.

>ALL(iç sorgu): Alt sorgudan elde edilen sonuçların hepsinden büyük kayıtları ifade eder.

<ALL(iç sorgu): Alt sorgudan elde edilen sonuçların hepsinden küçük kayıtları ifade eder.

!=ALL(iç sorgu) veya **<>ALL(iç sorgu)**: Alt sorgudan elde edilen sonuçların hepsinden farklı olan kayıtları ifade eder.

 SELECT * FROM staff WHERE salary > **All**(SELECT salary FROM staff WHERE sex = 'F');

Tüm kadınlardan daha fazla kazanan kişinin bilgilerini getirir

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005

▼ Exists/Not Exists

Alt sorguda eğer aranan değer bir tane dahi varsa **ana sorguyuyla alt sorguyu birleştirerek** çalıştırır.

 SELECT * FROM staff where exists (select * from staff where position='Supervisor')

Yukarıdaki döngüde dediği şey, eğer alt sorgu doğruysa bizden üst sorguyu yani tablonun hepsinin verisini getirmemizi istiyor. Tabloda da bir tane pozisyonu supervisor olduğu için bize tüm tabloyu getiriyor.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SL21	John	White	Manager	M	1965-10-01	30000	B005
2	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
3	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
4	SA9	Mary	Howe	Assistant	F	1990-02-19	9000	B007
5	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
6	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

▼ Tablo Birleştirme

Join'leme mantığını daha iyi anlaşılması için tıklayınız.

Yukarıda verilen link üstünden tablo birleştirme mantığı rahatça kavranılabilir. Yine de biz kendi oluşturduğumuz tablolar üzerinden de örnek çözmeye çalışalım :)

▼ Inner Join

Tabloların kesişim noktasını gösterir.

 SELECT client.clientno, fname, lname, viewing.comment FROM client **INNER JOIN** viewing **ON client.clientno=viewing.clientno** order by fname asc

Client ve viewing tablosundaki ortak değerleri getirdik

	clientno character (5) 	fname character varying (10) 	lname character varying (10) 	comment character varying (15) 
1	CR56	Aline	Steward	too small
2	CR56	Aline	Steward	
3	CR56	Aline	Steward	
4	CR76	John	Kay	too remote
5	CR62	Mary	Tregear	no dining room

Ama eğer yazdıracağımız tabloda **Select *** yapıp tüm değerleri istersek aynı sütundan ortaklaşa olanlar iki defa gelir.



```
SELECT *FROM client INNER JOIN viewing ON client.clientno=viewing.clientno  
order by fname asc
```

	clientno character (5) 	fname character varying (10) 	lname character varying (10) 	telno character (15) 	preftype character varying (10) 	maxrent integer 	email character varying (50) 	clientno character (5) 
1	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	CR56
2	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	CR56
3	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	CR56
4	CR76	John	Kay	0171-774-5632	Flat	425	john.kay@gmail.com	CR76
5	CR62	Mary	Tregear	01224-196720	Flat	600	marty@hotmail.co.uk	CR62

Gördüğü üzere clientno iki defa geldi bu tüm joinler için geçerlidir.

▼ Left Join

Sorguda fromun sağında olan tablonun değerlerini kaydeder, bu değerleri fromun solundaki tabloda arar.Eğer soldaki tabloda da örtüsen sütün bulursa örtüsen sütunun satırını yani, örtüsen değerlerin sol tablodaki değerleri getirir.Yani sağ tablodaki değerlerden eğer solda uyusan değer bulursa soldaki tablodan uyusan değerleri çeker.



```
SELECT *FROM viewing Left JOIN client ON client.clientno=viewing.clientno  
order by fname asc
```

Yukarıda viewing tablosundaki değerlerle alakalı değerler client tablosunda var mı diye bakmış, uyusanı bulduktan sonra birlikte yazdırmış.

	clientno character (5)	propertyno character (5)	viewdate date	comment character varying (15)	clientno character (5)	fname character varying (10)	lname character varying (10)	telno character (15)	preftyp character varying (10)
1	CR56	PA14	2015-05-24	too small	CR56	Aline	Steward	0141-848-1825	Flat
2	CR56	PG4	2015-05-26		CR56	Aline	Steward	0141-848-1825	Flat
3	CR56	PG36	2015-04-28		CR56	Aline	Steward	0141-848-1825	Flat
4	CR76	PG4	2015-04-20	too remote	CR76	John	Kay	0171-774-5632	Flat
5	CR62	PA14	2015-05-14	no dining room	CR62	Mary	Tregear	01224-196720	Flat

▼ Right Join

Left Join'in aynısı fakat tam tersi :)

▼ Full Outer Join

Tabloları direk birleştirir. Null olan yerleri de getirir.



SELECT * FROM viewing **full outer JOIN** client ON client.clientno=viewing.clientno

	clientno character (5)	propertyno character (5)	viewdate date	comment character varying (15)	clientno character (5)	fname character varying (10)	lname character varying (10)	telno character (15)	preftyp character varying (10)
1	CR56	PA14	2015-05-24	too small	CR56	Aline	Steward	0141-848-1825	Flat
2	CR76	PG4	2015-04-20	too remote	CR76	John	Kay	0171-774-5632	Flat
3	CR56	PG4	2015-05-26		CR56	Aline	Steward	0141-848-1825	Flat
4	CR62	PA14	2015-05-14	no dining room	CR62	Mary	Tregear	01224-196720	Flat
5	CR56	PG36	2015-04-28		CR56	Aline	Steward	0141-848-1825	Flat
6	[null]	[null]	[null]	[null]	CR74	Mike	Ritchie	01475-943-17...	House

▼ Natural Join

Yine aynı şekilde tabloları birleştirse de **Full Outer Joinden farkı** NULL değerleri getirmemesidir. Aynı zamanda on yapmama hakkımız var ama istenilen durumlarda da yapabiliriz.



SELECT * FROM client **NATURAL JOIN** viewing

	clientno character (5)	fname character varying (10)	lname character varying (10)	telno character (15)	preftyp character varying (10)	maxrent integer	email character varying (50)	propertyno character (5)	viewdate date	c c
1	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	PA14	2015-05-24	t
2	CR76	John	Kay	0171-774-5632	Flat	425	john.kay@gmail.com	PG4	2015-04-20	t
3	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	PG4	2015-05-26	
4	CR62	Mary	Tregear	01224-196720	Flat	600	marty@hotmail.co.uk	PA14	2015-05-14	r
5	CR56	Aline	Steward	0141-848-1825	Flat	350	astewart@hotmail.com	PG36	2015-04-28	

Aynı zamanda kötü tarafı ortak noktası olmayan tabloları kafasına göre birleştirir.



SELECT * FROM staff **NATURAL JOIN** viewing

▼ Union/ Intersect/ Except

▼ Union

- İki ya da daha fazla sorgu cümlesini bağlayarak yeni bir tablo oluşturur.
- Kısmen join gibi davranış ama **join sütunları birleştirirken, union satırları birleştirir**.
- Union aynı zamanda satırları districtler yani satırlar tekrar etmez, union all ise bunu umursamaz, tüm verileri getirir.
- Sorguları yazarken parantez kullanmak şart değildir.



(select city from branch where city is not null) **union**

(select city from propertyforrent where city is not null)

Şubesı veya emlaklı olan şehirleri getirilmesini sağladık

city		🔒
	character varying (10)	
1	London	
2	Bristol	
3	Glasgow	
4	Aberdeen	

▼ Intersect

- İki ya da daha fazla sorgu cümlesiinde kesişen verileri getirerek yeni bir tablo oluşturur.



(select city from branch where city is not null) **intersect**

(select city from propertyforrent where city is not null)

Yukarıda iki sorguda kesişen yerleri getirdik, yani hem şubesı hem emlaklı olan yerleri getirdik.

city	
	character varying (10)
1	Aberdeen
2	London
3	Glasgow

 select distinct city from branch b where **exists** (select city from propertyforrent p where p.city=b.city)

 select distinct p.city from branch b ,propertyforrent p where p.city=b.city

İkisi de yukarıdaki tabloyu verir.

▼ Except

- İki sorgu cümlesinin birbiriyle olan farklarını getirir. Yani kümelerde yaptığımız fark işleminin aynısıdır: kendisinde olup, diğer kümede olmayan verileri getirir.
- NOT IN de aynı şekilde davranışır fakat farkı, except districtlerken not in hepsini getirir.

 (select city from branch where city is not null) **except**
(select city from propertyforrent where city is not null)

Şubesı olup emlakı olmayan şehirleri getirmeyi istedik.

city	
	character varying (10)
1	Bristol

▼ Insert

Tabloya veri eklemek için kullandığımız komuttur.

 **INSERT INTO tablo_adi (kolon1,kolon2,kolon3, ...)**
VALUES(değer1,değer2,değer3, ...) (into eklenmesi şart değil)

 **INSERT INTO tablo_adi VALUES (değer1,değer2,değer3, ...)**

▼ Update

Tablodaki verileri güncellemek için kullandığımız komuttur.

 **UPDATE staff *SET lname='ucar'* WHERE staffno='SA9'**

Staffno'su SA9 olan satırın verisini güncellemiş olduk.

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SA9	Mary	ucar	Assistant	F	1990-02-19	9000	B007

 **UPDATE staff SET salary=salary*1.3**

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SA9	Mary	ucar	Assistant	F	1990-02-19	9000	B007
2	SG14	David	Ford	Supervisor	M	1978-03-24	18000	B003
3	SG37	Ann	Beech	Assistant	F	1980-11-10	12000	B003
4	SG5	Susan	Brand	Manager	F	1960-06-03	24000	B003
5	SL21	John	White	Manager	M	1965-10-01	30000	B005
6	SL41	Julie	Lee	Assistant	F	1985-06-13	9000	B005

	staffno [PK] character (5)	fname character varying (10)	lname character varying (10)	position character varying (10)	sex character (1)	dob date	salary integer	branchno character (5)
1	SA9	Mary	ucar	Assistant	F	1990-02-19	11700	B007
2	SG14	David	Ford	Supervisor	M	1978-03-24	23400	B003
3	SG37	Ann	Beech	Assistant	F	1980-11-10	15600	B003
4	SG5	Susan	Brand	Manager	F	1960-06-03	31200	B003
5	SL21	John	White	Manager	M	1965-10-01	39000	B005
6	SL41	Julie	Lee	Assistant	F	1985-06-13	11700	B005

▼ Constraints(SQL kısıtlamaları)

SQL kısıtlama mantığını daha detaylı öğrenilmesi için bıraktığım link için tıklayınız.

Constraints, bir tablodaki veriler için kurallar belirtmek için kullanılır.

Constraints ALTER TABLE ya da CREATE TABLE ifadeleri ile tanımlanır
 *alandan kastettiği sütundur.

NOT NULL	Bir sütunun değerİNİN BOŞ OLMAMASINI SAĞLAR	
INDEX	Veritabanından çok hızlı bir şekilde veri almak için kullanılır.	
DEFAULT	Alana bir değer girilmediğinde oto olarak değer atar.	SÜTUN
CHECK	Bir alana girilen değerin o alan için uygun olup olmadığını tartar	
REFERENTIAL	Bir tablonun değerini referans olarak bir diğer tablonun değerini belirler.	
PRIMARY KEY	Bir alanda girilen değerlerin birbirinden farklı olmasını sağlar, bir tabloda sadece bir tane bulunur.	SATIR
UNIQUE	Bir tablonun bir ya da daha fazla alanında aynı değerden tekrar girilmesini engeller.	
FOREIGN KEY	Bir tablonun primary key olan alanı ile aynı ya da farklı bir tablonun bir alanındaki değerlerin birbiri ile eşleştirilmesini sağlar.	DİĞER TABLOLARLA İLİŞKİLİ
CHECK	Bir tabloda bir alana girilen değerleri, diğer alanlardaki değerleri göz önüne alarak geçerliliğine bakar.	

▼ Domain

Domain mantığını daha detaylı açıklayan website linki için tıklayınız.

Sql'de kolonları oluştururken onlara bir çok özellik gireriz. Bu özellikler hangi veri tipinde olacağı (char, numeric vs), not null olacağı gibi özellikler olabilir. Eğer bir özellik kümesini bir çok kolonda kullanıyorsak bu özellikleri tek tek yazmak yerine kendi veri tipimizi oluşturabiliriz ve bu kolonlarda oluşturduğumuz veri tipini kullanabiliriz. Yani CREATE DOMAIN yeni bir veri alanı tanımlar.



CREATE DOMAIN yas AS integer NOT NULL CHECK(value >= 18);



```
CREATE TABLE Kisi(
  kisi_tc char(11) PRIMARY KEY,
  kisi_ad varchar(30),
  kisi_yas yas);
```

kisi_tc

[PK] character (11)

kisi_ad

character varying (30)

kisi_yas

integer

Verdiğimiz domain sayesinde yas için teker teker veri girmektense, bir veri türü yerine koyarak sütunumuzu tanımladık.

▼ NOT NULL

- `NOT NULL` bir sütunu NULL değerleri kabul etmemeye zorlar.
- Bu, bir alanı her zaman bir değer içermeye zorlar; bu, bu alana bir değer eklenmeden yeni bir kayıt ekleyemeyeceğiniz veya bir kaydı güncellemeyeceğiniz anlamına gelir.

 CREATE TABLE Persons (
ID int NOT NULL);

 ALTER TABLE Persons
ALTER COLUMN Age int NOT NULL;

▼ UNIQUE

- `UNIQUE` bir sütundaki tüm değerlerin farklı olmasını sağlar.
- Bir `PRIMARY KEY` kısıtlamanın otomatik olarak bir `UNIQUE` kısıtlaması vardır.
- Primary Key'den farkı; bir tabloda sadece bir primary key varken, unique birden fazla olabilir, sınırlama yoktur yani.

 CREATE TABLE Persons (
ID int NOT NULL UNIQUE);

Yeni oluşturduğumuz tablodaki ID değerinin unique ve not null olmasını sağladık.

 ALTER TABLE Persons
ADD UNIQUE (ID);

Eski tablodaki ID değerinin unique ve not null olmasını sağladık.

```
 ALTER TABLE Persons  
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

Eski tablodaki ID ve LastName değerinin unique ve not null olmasını sağladık. Bu işleme de UC_Person adını verdik.

```
 ALTER TABLE Persons  
DROP CONSTRAINT UC_Person;
```

UC_Person işlemini Persons tablosundan kaldırındı.

▼ PRIMARY KEY

- Kısıtlama, bir tablodaki **PRIMARY KEY** her kaydı benzersiz bir şekilde tanımlar.
- Bir tabloda sadece bir sütuna primary key verilebilir.
- Insertleme işlemi yapılrken **primary key olan sütuna null yazılamaz.**
- Harddisk'te primarykey değerleri sıralı şekilde tutulur.

```
 CREATE TABLE Persons (  
ID int NOT NULL PRIMARY KEY );
```

ID'ye primary key tanımladık.

```
 CREATE TABLE Persons (  
ID int NOT NULL,  
LastName varchar(255) NOT NULL,  
CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)  
);
```

- Yukarıdaki kodda tek primary key vardır, ama primary key birden fazla alanı içine almıştır. Yani ikisini bir kalıp olarak almıştır. Aşağıdaki satırda açıklanmıştır.
- (ID:2, LastName: Esen),(ID:2, LastName: Dalar)

```
✓ ALTER TABLE Persons  
ADD PRIMARY KEY (ID);
```

Eski tablodaki ID değerinin primary key olmasını sağladık.

```
✓ ALTER TABLE Persons  
DROP CONSTRAINT PK_Person;
```

Primary key kısıtlamasını kaldırmak için yazılan komut.

▼ FOREIGN KEY

- **FOREIGN KEY** Kısıtlama, tablolar arasındaki bağlantıları yok edecek eylemleri önlemek için kullanılır .

Kişiler Tablosu

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Sipariş Tablosu

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

"Siparişler" tablosundaki "PersonID" sütununun "Persons" tablosundaki "PersonID" sütununu gösterdiğine dikkat edin.

"Kişiler" tablosundaki "PersonID" sütunu, "Kişiler" tablosundaki sütundur **PRIMARY KEY**.

"Siparişler" tablosundaki "PersonID" sütunu, "Siparişler" tablosundaki bir sütundur **FOREIGN KEY**.

Eğer biz kişiler tablosundan birisini silmeye çalışırsak sipariş tablosu hata verir. Çünkü birisi silindiğinde person id bilgisi de silinir bu yüzden sipariş tablosundaki person Id' de silinmek zorunda kalır. Bunu önlemek için foreign key kullanız.

```
 CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

PersonIDyi Persons tablosundaki PersonID'den alarak foreign key yapmıştır.

```
 ALTER TABLE Orders
ADD constraint
FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

Aynı kod, sadece hazır tabloda uyarladığımız halde.

```
 ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

Foreign key kısıtlamak için yazdığımız komut.

▼ CHECK

 **CHECK** Kısıtlama, bir sütuna yerleştirilebilecek değer aralığını sınırlamak için kullanılır .

```
 CREATE TABLE Persons (
    Age int CHECK (Age>=18)
);
```

Veri girerken yaşı 18 ve 18'den büyük olmazsa kayıt yapılmasını engeller.

```
 CREATE TABLE Persons (
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

Aynı anda birden fazla check işlemi yapmak istediğimizde yukarıdaki gibi yaparız.

 ALTER TABLE Persons
ADD CHECK (Age>=18);

Varolan tabloda yaptığımız check işlemi.

 ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');

Yaptığımız check işlemine isim verdik.

 ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;

İsim verdigimiz check işlemini sildik.

▼ DEFAULT

- **DEFAULT** Kısıtlama, bir sütun için varsayılan bir değer ayarlamak için kullanılır .
- Başka bir değer belirtilmemişse, varsayılan değer tüm yeni kayıtlara eklenecektir.

 CREATE TABLE Persons (
City varchar(255) DEFAULT 'Sandnes'
);

Veri girilmediği takdirde şehir sütunu için, oto olarak Sandnes yazdırır.

 CREATE TABLE Orders (
OrderDate date DEFAULT GETDATE()
);

Fonksiyon kullanarak da yazabiliriz.

 ALTER TABLE Persons
ADD CONSTRAINT df_City
DEFAULT 'Sandnes' FOR City;

dfcity işlemin adıdır.

 ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT;

Sildik.

▼ Create Table

 **CREATE TABLE** Branch
(branchNo char(5) PRIMARY KEY,
street varchar(35),
city varchar(10),
postcode varchar(10)
);

Başka bir tablonun verisi kullanarak tablo oluşturma

 **CREATE TABLE** new AS
SELECT fname, lname
FROM staff;

Böylece yeni oluşturulan new tablosunun sütunları fname ve lname olmuştur. bu tablonun sütunları ve verileri de staff tablosundan çeki

▼ Alter Table

Alter Table komutu sütunları ekler, siler veya değiştirir.

 **ALTER TABLE** staff
ADD Email varchar(255);

Burada staff tablosuna email sütunu ekledik.

email
character varying (255)
[null]
[null]
[null]
[null]
[null]

 **ALTER TABLE** staff **RENAME COLUMN** "email" **TO** "newemail"

newemail
character varying (25)
[null]
[null]
[null]
[null]
[null]

 **ALTER TABLE** staff
ALTER COLUMN "newemail" **type** varchar(50)

Böylece email sütunun karakter sınırını 25'ten 50yaptık.

newemail
character varying (50) 

[null]

[null]

[null]

[null]

[null]

 ALTER TABLE staff
DROP COLUMN newemail

newemail sütununu sildik.

▼ Drop Table

 **DROP TABLE Shippers;**

Tabloyu direk olarak siler.

▼ View

- SQL'de görünüm, bir SQL ifadesinin sonuç kümesine dayalı sanal bir tablodur.

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

Customers tablosunda countrysi Brazil olan verilerin customername ve contactnameini alıp, oluşturduğumuz Brazil Customer tablosuna atar.

 **CREATE VIEW** [Products Above Average Price] AS
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);

Product tablosunda ortalama price'tan daha yüksek olan priceların productnameini ve priceini alır yeni oluşturduğu viewin(Products Above Average Price) içine atar

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

Brazilya customers viewine city sütunu eklenmiştir.

 **DROP VIEW** [Brazil Customers] [CASCADE OR RESTRICT];

CASCADE: Silinen bir görünüm başka bir viewdeki veriye referans oluyorsa yani başka view , silinecek olan tabloya bağlısa ; bağlılığı kaldırır, görünümü siler.(foreign key olması durumunda örneğin)

RESTRICT: Yukarıdaki durumun tam tersidir. Yani başka bir view bu silinecek olan viewe bağlısa silme işlemini reddeder.

- **Görünüm çözünürlüğü ile, görünüm üzerinde herhangi bir işlem, türetildiği ilişkiler üzerindeki operasyonlara otomatik olarak dönüştürülür.**
- **Görünüm maddileştirme ile, görünüm bir geçici tablo olarak depolanır, alta yatan temel tablolar güncellenmiş olarak tutulur.**

▼ With Check Option

	branchno [PK] character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	56 Clover Dr	ankara	NW10 6EU
2	B004	32 Manse Rd	Bristol	BS99 1NZ
3	B005	22 Deer Rd	London	SW1 4EH
4	B007	16 Argyll St	Aberdeen	AB2 3SU
5	B010	163 Main St	Glasgow	G11 9QX
6	B044	dfldskf	klsfkll	ds
7	B046	bostanbaşı	malatya	44000
8	BB06	mehmetakif	ankara	06000

Kullanacağımız branch tablosu



create or replace view deneme as select * from branch where city='ankara' or city='malatya'

Branch tablosunda şehri malatya ve ankara olan şehirleri verileriyle birlikte getirip yeni oluşturduğumuz deneme viewine attık.

	branchno character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	56 Clover Dr	ankara	NW10 6EU
2	B046	bostanbaşı	malatya	44000
3	BB06	mehmetakif	ankara	06000



insert into deneme(branchno,street,city,postcode) values ('a','b','c','d')

Deneme viewine değer girmeye çalıştık ama girmedi.Ama brach tablosuna eklendi.

	branchno character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	56 Clover Dr	ankara	NW10 6EU
2	B046	bostanbaşı	malatya	44000
3	BB06	mehmetakif	ankara	06000

	branchno [PK] character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	56 Clover Dr	ankara	NW10 6EU
2	B004	32 Manse Rd	Bristol	BS99 1NZ
3	B005	22 Deer Rd	London	SW1 4EH
4	B007	16 Argyll St	Aberdeen	AB2 3SU
5	B010	163 Main St	Glasgow	G11 9QX
6	B044	dfldskf	klsfkll	ds
7	B046	bostanbaşı	malatya	44000
8	BB06	mehmetakif	ankara	06000
9	a	b	c	d

Şaşırıcı.



UPDATE deneme SET street='Nehaa',city='all',postcode='dadas' WHERE branchno='B002'

Deneme viewinde branchno'su B002 olan satırı güncellemeye çalıştık. Ama onun yerine branchno'su B002 olan satırı sildi.

	branchno character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B046	bostanbaşı	malatya	44000
2	BB06	mehmetakif	ankara	06000

Ama aynı anda branch tablosunda da güncellendi.

	branchno [PK] character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	Nehaa	all	dadas
2	B004	32 Manse Rd	Bristol	BS99 1NZ
3	B005	22 Deer Rd	London	SW1 4EH
4	B007	16 Argyll St	Aberdeen	AB2 3SU
5	B010	163 Main St	Glasgow	G11 9QX
6	B044	dfldskf	klsfkll	ds
7	B046	bostanbaşı	malatya	44000
8	BB06	mehmetakif	ankara	06000
9	a	b	c	d

 CREATE or replace VIEW denemewith AS

SELECT * FROM branch

WHERE (city='all') WITH CHECK OPTION

	branchno character (5)	street character varying (35)	city character varying (10)	postcode character varying (10)
1	B002	Nehaa	all	dadas
2	B046	Nehaa	all	44000
3	BB06	Nehaa	all	dadas

 insert into denemewith(branchno,street,city,postcode) values ('s','b','c','d')

With check option kullandığımız için yükleme yapmamıza izin vermez.

 UPDATE denemewith SET street='Nesda',city='afsl',postcode='güncelleme'
WHERE branchno='BB06'

With check option kullandığımız için güncelleme yapmamıza izin vermez.

CHECK OPTION

Güncellenenbilir sanal tablolar yapar. Sanal tablo üzerindeki tüm **INSERT** ve **UPDATE** komutlarının sanal tablo tanımlama koşullarına uygunluğu sınanacaktır (yani, yeni verinin sanal tablo üzerinden görünür olması gereklidir). Sınama başarısız olursa, güncelleme reddedilecektir.

LOCAL

Sanal tablonun kendi bütünlüğü sınanır.

CASCADE

Sanal tablonun diğer sanal tablolarla bütünlük içinde olup olmadığı sınanır. Ne **CASCADE** ne de **LOCAL** belirtilmişse, **CASCADE** öntanımlıdır.

▼ View Kısıtlamları

- viewe erişmek için; Sütun, sadece select ve order by cümleciğinde çağrırlabilir. Yani mesela sum, count gibi küme fonksiyonlarını çağrıramayız.
- Sütun, where'de ve görünümde dayalı herhangi bir sorguda bir küme fonksiyonuna argüman olarak getiremez.

● Örneğin, aşağıdaki sorgu başarısız olur:

```
SELECT COUNT(cnt)
  FROM StaffPropCnt;
```

● Benzer şekilde, aşağıdaki sorgu da başarısız olur:

```
SELECT *
  FROM StaffPropCnt
 WHERE cnt > 2;
```

Görünüm Güncelleştirilebilme

- Temel tablodaki tüm güncellemler, temel tabloyu kapsayan tüm görüntülerde yansıtılır.
- Benzer şekilde, görünüm güncellendiğinde, temel tablo(lar) değişikliği yansıtmalı diye beklenebilir.

- ISO görünümün güncelleştirilebilir olduğunu ancak ve ancak şu şartlar için belirtir:
 - DISTINCT belirtilmemişse.
 - Tanımlayan Sorgunun SELECT listesinde her eleman bir sütun adıdır ve hiçbir sütun birden fazla olamaz.
 - FROM cümlecığında; bir birleştirme, birleşim, kesişim veya farka dayalı herhangi bir görünüm hariç, yalnızca bir tablo belirtilir.
 - Dış tabloya başvuran hiçbir iç içe SELECT olamaz.
 - GROUP BY veya HAVING cümlecekleri olamaz.
 - Ayrıca, görünüme eklenen her satır, temel tablonun bütünlük kısıtlamalarını ihlal etmemelidir.

- Görünümde ekleme/güncelleme yapıldığında, WHERE şartını sağlayan yeni satırlar görünümde görünür.
- Bir görünüme yeni giren veya çıkan satırlara, göç eden satırlar denir.

- Tanımlanan sorgunun WHERE koşulunu karşılayan Satırlar bir görünümde var olur.
- Bir satır değiştiğinde ve artık koşulu sağlamıyorsa, görünümden kaybolur.
- Görünümde ekleme/güncelleme yapıldığında, WHERE şartını sağlayan yeni satırlar görünümde görünür.
- Bir görünüme yeni giren veya çıkan satırlara, göç eden satırlar denir.

Görünüm Gerçekleme

- Görünüm çözünürlüğü mekanizması, eğer görünüm sık erişiliyorsa, yavaş olabilir.
- Görünüm ilk sorgulandığında, Görünüm gerçekleme geçici tablo olarak görünümü depolar.
- Bundan sonra, gerçeklenmiş görünümeye dayalı sorgular, her zaman, yeniden hesaplanan görünümden daha hızlı olabilir.
- Sorun, Temel tablolar güncellenirken, görünümün geçerliliğinin sürdürülmesidir.

Görünüm Bakımı

- Görünüm bakımı, görünümü güncel tutmak için gerekli değişiklikleri uygulamayı hedefler.

- Aşağıdaki görünümü düşünün:

```
CREATE VIEW StaffPropRent(staffNo)
AS SELECT DISTINCT staffNo
        FROM PropertyForRent
        WHERE branchNo = 'B003' AND
              rent > 400;
```

staffNo
SG37
SG14

Görünüm Gerçekleme

- **PropertyForRent** içine Kira <= 400 ile satır eklenirse, görünüm değişmeyecektir.
- **emlak PG24** için şube B003 de StaffNo=SG19 ve kira=550 ile satır eklenirse, satır gerçeklenen görünümde görünür.
- **emlak PG54** için şube B003 de StaffNo=SG37 ve kira=450 ile satır eklenirse, yeni satırın gerçeklenen görünümde eklenmesi gerekmektedir.
- **PG24** özelliği Silinirse, satır gerçeklenen görünümde silinmelidir.
- **PG54** özelliği Silinirse, **PG37** için satır silinmemelidir (**emlak PG21** mevcut olduğundan).

▼ View Avantajı/ Dezavantajı

View Avantajı	View Dezavantajı
Veri bağımsızlığı	Güncelleme Kısıtlaması
Geçerlilik	Yapı Kısıtlaması
Geliştirilmiş Güvenlik	Performan
Azaltılmış Karmaşıklık	
Kolaylık	
İsteğe göre düzenleme	
Veri bütünlüğü	

▼ View'in Avantajları/ Dezavantajları

View Avantajı	View Dezavantajı
Veri bağımsızlığı	Güncelleme Kısıtlaması
Geçerlilik	Yapı Kısıtlaması
Geliştirilmiş Güvenlik	Performan
Azaltılmış Karmaşıklık	
Kolaylık	
İsteğe göre düzenleme	
Veri bütünlüğü	

▼ Yetkilendirme

Sql tablolarımızı birden fazla kullanıcı kullanabilir ve bu kullanıcıların tablolar üzerinde tüm işlemleri yapmasını genelde istemeyiz. Bu gibi durumlarda kullanıcılar yetkiler tanımlayarak tablolar üzerinde yaptıkları işlemleri sınırlayabiliriz.

Read	Tablodan veri okuma
Insert	Tabloya veri yazma
Update	Tablodaki verileri güncelleme
Delete	Tablodaki verileri silme
Index	Tablo üzerinde index tanımlayabilme
Resources	Tablolar üzerinde ilişki tanımlayabilme
Alteration	Tablonun değiştirilmesi
Drop	Tablonun silinmesi
All Privileges	Tüm yetkiler tanınır
Select	Bir tablodan veri alma yetkisi
References	Bütünlük kısıtlamalarında belirtilen tablonun referans sütunları(?)
Usage	Alan, tanımlama ve karakter kümeleri kullanma yetkisi



GRANT yetki **ON** tablo_adi **TO** kullanıcılar

Yetki verme komutu cümlesi.



REVOKE yetki **ON** tablo_adi **FROM** kullanıcılar

Yetkiyi kaldırma komutu cümlesi.

▼ Rol Tanımlama

Genellikle veritabanı üzerinde işlem yapan kullanıcıların benzer yetkileri olur. Yani aynı işi yapan birden çok kişi olabilir. Örnek verecek olursak veritabanı üzerinden veri okuyarak rapor oluşturan onlarca kişi olabilir. Bu gibi durumlarda her kullanıcı için ayrı ayrı yetki tanımlaması yapmak yerine rol tanımlaması yapılabilir. Bu rollere yetki tanımlaması yapılabilir ve kullanıcılar yetki vermek yerine rol verilebilir.

View ve tablo üzerinden rol tanımlanabilir.(tablo adı yerine view adı yazabilirsiniz)

Rol tanımlama

 **CREATE ROLE rol_adi (Röl Oluşturma)**

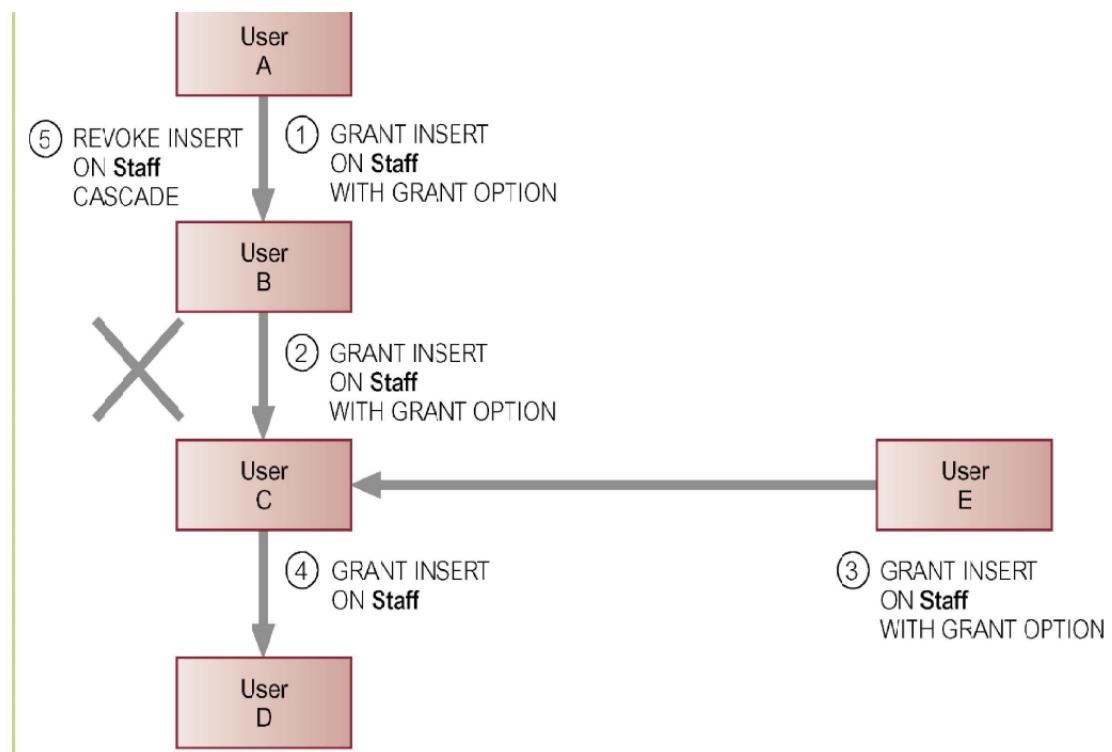
 **GRANT yetki ON tablo_adi TO rol_adi (Role Yetki Atama)**

rol_adi public yapılrsa herkes erişebilir.

 **GRANT rol_adi TO kullanıcılar (Role Kişi Atama)**

Röl Tanımından Yetki Silme

 **REVOKE yetki FROM rol_adi**



- REVOKE, GRANT ile verilen yetkileri geri alır.

```
REVOKE [GRANT OPTION FOR]
{YetkiListesi | ALL PRIVILEGES}
ON Nesneİsmi
FROM {YetkildList | PUBLIC}
[RESTRICT | CASCADE]
```

- ALL PRIVILEGES kullanıcı iptal etme yetkileri ile bir kullanıcıya verilen tüm yetkileri ifade eder.

REVOKE

- GRANT OPTION FOR, GRANT'ın WITH GRANT OPTION yoluyla geçen yetkilerin ayrı iptal edilmesini sağlar.
- CASCADE anahtar kelimesi belirtilmediği sürece, REVOKE geri bırakılan görünüm gibi bir nesneye neden olursa başarısız olur.
- Diğer kullanıcılar tarafından bu kullanıcıya verilen yetkiler etkilenmez.

Rol Tanımı Silme

 DROP ROLE *rol_adi*

Başka Tablonun Rolünü , Başka Bir Tabloya Verme

 GRANT ALL PRIVILEGES ON STAFF **TO** MANAGER **WITH GRANT OPTION**

Yukarıda yazan komut ile Staff tablosundaki tüm kısıtlamaları Manager tablosu için de geçerli kılmış oluruz.



GRANT ALL select, update(salary) on staff **TO** personel, director

Staff tablosundaki salary'e Select' leme ve Update'leme yetkisini Personel ve Director'e verir.