SÜRÜM 1.0

26.10.2021

TEMEL ORACLE SQL

SQL'E DAİR HERŞEY

alisahan.com

@alisahancom

YAZAR ALİ ŞAHAN
alisahan.com

1.VERİ İŞLEME DİLİ (DML) / DATA MANİPULATİON LANGUAGE (DML)

2. VERİ TANIMLAMA DİLİ (DDL) / DATA DEFİNİTİON LANGUAGE (DDL)

3 VERİ KONTROL DİLİ (DCL) / DATA CONTROL LANGUAGE (DCL)

4 İŞLEM KONTROLÜ / TRANSACTİON CONTROL (TCL)

VERİ İŞLEME DİLİ (DML) / DATA MANİPULATİON LANGUAGE (DML)

DML , ”Veri işleme Dili” Tablo verilene erişmek, eklemek, değiştirmek ve silmek için kullanılan ifade türleridir. Tek bir satır yâda tabloların hepsi için aynı anda kullanıla bilir. Bu ifadeler insert, select, update, delete komutları olarak düşüne bilirsiniz.

SELECT

SQL SELECT deyimlerinin nasıl çalıştığı hangi sonuçlar döndürdüğünü ve neler yapacağınızı bu yazı ile geliştire bilirsiniz

Temel bir SELECT ifadesi alta belirtildiği gibi tanımlanır

Makale içinde kullanılacak tüm SQL cümlelerinde “hastane” varsayılan Oracle Schema “şema” adı, olarak belirtilmiştir. Örnek olarak hastane.kimlik,hastane.protokol vb.

SELECT * FROM SEMA_ADI.TABLO_ADI “tablo sahibi ile seçme belirleme”

SELECT * FROM TABLO_ADI “sahipsiz belirleme”

- **Select başlangıç komutu ile başlar**
- *** komutu ile gösterilecek kolon isimleri belirlenir**
- **FROM ise bağlanacak tablo adlarını belirler**

ÖRNEK

SELET * FROM KIMLIK	Basit SQL cümlesi
SELECT * FROM HASTANE.KIMLIK	Şema adı olarak tablo seçme
SELECT ADI, SOYADI, TC_KIMLIK_NO FROM HASTANE.KIMLIK	Tablodaki tüm alanlar yerine sadece belirli kolonları getirme

- SQL cümleleri birden fazla yazılması gerekiyor ise ile kapatılır
- SQL sonuçları tek bir sonuç yâda daha fazla satır döndüre bilir
- SQL cümlelerinde küçük büyük harf duyarlılığı yoktur. Eng. Karakter için

Bir sütunün adını değiştirmek istediğinizde kullana bilirsiniz. Hesaplama işlemlerinde kullanışlıdır. Ayrıca sütun adını takip ederek çıktı sonuçlarınıza düzen kata bilirsiniz

Boşluk içeren takma adları kullanmak istediğinizde ise **AS** komutu ile kullana bilirsiniz

ÖRNEK

SELECT ADI, SOYADI, TC_KIMLIK_NO FROM HASTANE.KIMLIK

VEYA

SELECT ADI HASTA, SOYADI SOYISIM , TC_KIMLIK_NO AS “TC NO” FROM HASTANE.KIMLIK

Kullana bilirsiniz.

Birden fazla sütunu birleştirmek istediğinde kullanacağınız operatör komutu.

ÖRNEK

Standart bir SQL cümlesi

```
SELECT ADI, SOYADI, TC_KIMLIK_NO  FROM HASTANE.KIMLIK
```

ADI	SOYADI	TC_KIMLIK_NO
bebek deneme	22	
bebek deneme	deneme	
deneme	test mod on	12365474120

Birleştirme || kullanıldığında ise SQL ve çıktı sonucu

```
SELECT ADI||' '|| SOYADI HASTA, TC_KIMLIK_NO  FROM HASTANE.KIMLIK
```

HASTA	TC_KIMLIK_NO
bebek deneme 22	
bebek deneme deneme	
deneme test mod on	12365474120

Sonuç çıktılarınızda okuna bilirliği artırmak veya tek tırnak ‘ karakterini kullanmak için q operatörünü kullana bilirsiniz. Birleştirilen ifade tırnak içerirse birleştirme öncesi tırnak iki kez kullanarak sorgulara dâhil edilir

ÖRNEK

```
SELECT ADI|| q'' Hasta'nın  Tc Numarası >>''|| tc_kimlik_no
as "Hasta Bilgisi"   FROM HASTANE.KIMLIK
```

Hasta Bilgisi
bebek deneme Hasta'nın Tc Numarası >>
bebek deneme Hasta'nın Tc Numarası >>
deneme Hasta'nın Tc Numarası >>12365474120

Sorgularınız şart kısımlarını oluşturan eşittir, büyüktür küçüktür içerir içinde yâda arasında vb. düzeydeki karşılaştırma operatörlerini kapsar

OPERATÖR	AÇIKLAMASI
=	Eşittir
>	Büyüktür
>=	Büyük ve eşittir
<	Küçüktür
<=	Küçük ve eşittir
<>	Eşit Değildir
BETWEEN ...AND	İki değer arasında (Başlangıç ve bitiş DAHİL)
IN (SET)	İçinde belirtilen değerlerden herhangi birine eşit
LIKE	Karakter değerler için % değeri ile belirli bir kısmı yada tamamını içerenleri kapsar Örneğin adi like 'A%' a ile başlayan sonu önemli olmayan
IS NULL	Değeri boş null olanlar
IS NOT NULL	Değeri boş olmayan dolu olanlar

ÖRNEK

= EŞİTTİR ÖRNEĞİ

```
SELECT adi,soyadi,tc_kimlik_no,memleket, dogum_tar FROM KIMLIK WHERE
adi='ali'
--adı ali olan kimlik bilgileri
```

ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
ali	akar	62202581414	BARTIN	20/11/1988
ali	akfırat	99525736843	ŞIRNAK	1/01/1945
ali	akın	76771251222	BOLU	4/05/1964
ali	alagöz	99262014196	RİZE	1/01/1960
ali	altın	59048699296	MUĞLA	5/05/1985

Kimlik tablosunda adi ali ismine eşit olan kimlik bilgilerini listeler

Tablo sadece adi,soyadi,tc_kimlik_no,memleket ve doğum tarihi alanları gösterir

SELECT *FROM kullanılmış olsaydı tüm alanları listeleyecektir

- > Büyüktür Örneği

```
SELECT dosya_no,adi, soyadi,tc_kimlik_no,memleket,dogum_tar FROM KIMLIK
WHERE  DOSYA_NO>23870
--dosya numarası 23870'dan büyük olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
23871	elif	uras	53512765043	EDİRNE	18/12/1991
23872	avaş	kaya	39281048365	HATAY	7/03/1958
23873	adnan eymen	kütük	29096267922	ÇANAKKALE	25/02/2015
23874	hatice	orak	17185176292	ADANA	14/01/1989
23875	feray	terlemez	81824072798	ARTVİN	2/06/1986
23876	ayşe gül	yaşar	11429019623	RİZE	30/12/2000

Kimlik tablosunda *dosya numarası 23870'dan büyük olan* kimlik bilgilerini listeler

>= Büyük ve Eşit

```
SELECT dosya_no,adi, soyadi,tc_kimlik_no,memleket,dogum_tar FROM KIMLIK
WHERE  DOSYA_NO>=23870
--dosya numarası 23870'dan büyük ve aynı zamanda eşit olan kimlik
bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
23870	zeynep	turan	30703249132	SİİRT	24/07/2008
23871	elif	uras	53512765043	EDİRNE	18/12/1991
23872	avaş	kaya	39281048365	HATAY	7/03/1958
23873	adnan eymen	kütük	29096267922	ÇANAKKALE	25/02/2015
23874	hatice	orak	17185176292	ADANA	14/01/1989
23875	feray	terlemez	81824072798	ARTVİN	2/06/1986
23876	ayşe gül	yaşar	11429019623	RİZE	30/12/2000

Kimlik tablosunda *dosya numarası 23870'dan büyük ve aynı zaman da eşit olan* kimlik bilgilerini listeler

< Küçüktür

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE DOSYA_NO <5
-- dosya_no 5 'den küçük olanlar
```

<= Küçüktür ve Eşit

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE DOSYA_NO <=5
-- dosya_no 5 'den küçük ve 5 eşit olanlar
```

<> Eşit Değildir

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE ADI<>'ali'
--adi ali 'ye eşit olmayanlar
```

BETWEEN başlangıç ve bitiş değerleri dâhil arasında olan

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE DOSYA_NO BETWEEN 1 AND 5
-- dosya_no değeri 1 ve 5 dahil arasında olanlar
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
1	meddata	meddata	12441144687	UŞAK	1/04/1991
2	deneme	umit	16915266483	ERZİNCAN	1/01/1991
3	hakan	kılıç	70550896722	ÇORUM	17/03/1986
4	ayça	kılıç	40650782738	EDİRNE	12/06/1992
5	muhammed ali	güler	13966456725	KONYA	18/05/1994

IN içinde belirtilen değerleri kapsayan

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE DOSYA_NO IN (1,3,5,8)
--dosya_no değeri 1,3,5 ve 8 eşit olanlar
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
1	meddata	meddata	12441144687	UŞAK	1/04/1991
3	hakan	kılıç	70550896722	ÇORUM	17/03/1986
5	muhammed ali	güler	13966456725	KONYA	18/05/1994
8	funda	uygur	16564524223	KIRŞEHİR	7/07/2001

IS NULL değeri boş null olanlar

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE SOYADI is null
--SOYADI değeri boş olanlar
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	MEMLEKET	DOGUM_TAR
432			71130151925	ESKİŞEHİR	
644			49294494054	KARS	
1045			89646131106	ERZURUM	
4748			72163751625	ZONGULDAK	2/02/1996

IS NOT NULL değeri boş olmayanlar dolu olanları kapsar

```
SELECT dosya_no, adi, soyadi, tc_kimlik_no, memleket, dogum_tar FROM KIMLIK
WHERE SOYADI is not null
--SOYADI değeri dolu olanlar
```

INSERT

Tablolara yeni satır eklemek için kullanılan ifade biçimi. **INSERT** komutu ile tablolara yeni satırlar oluştura bilirsiniz. Basit bir insert komutu ⇓

```
insert into kimlik (dosya_no,adi,soyadi,tc_kimlik_no) values
(1,'ali','şahan','22345678987');
```

`insert into`	İnsert komutu
`kimlik`	Yeni kayıtların oluşacağı tablo bilgisi
`(dosya_no,adi,soyadi,tc_kimlik_no)`	İlgili tablodaki ilgili alanlar
`values`	Values komutu
`(1,'ali','şahan','22345678987');`	İlgili alanlar için tanımlanmış değerler

```
INSERT INTO table [(column [, column...])] VALUES (value [, value...]);
```

- İlgili komut ile tek bir satır oluştura bilirsiniz,
- İnsert komutunda belirtilen her bir kolon için values alanında bir değer oluşturulmalıdır
- Kolon ve değerler sırası ile belirtilmelidir
- Karakter yazı string değerler ‘ tek tırnak ile belirtilmelidir Örneğin ali değeri ‘ali’ olarak yazılmalıdır
- Tarih değerleri de to_date yada ’01.01.2021’ vb. türde belirtilmelidir.
- Değerleri tanımlarken & ile yer tutucuları kullana bilirsiniz. &adi kullandığınızda sql editörünüz sizden değer isteyecektir. Örnek Yer Tutucular

  Belirtilen kurallara uyulduğu takdirde INSERT komutunuz hatasız çalışacaktır. Şimdi yukardaki örnekleri kapsayan birkaç SQL yazalım

ÖRNEK

TABLONUN TUM DEGELERİ YAZILIRSA KOLON BELİRTMEYE GEREK YOKTUR

```
insert into kimlik values
(1,'ali','sahan','22345678985','01.01.1986','YOZGAT');
```

TEK TIRNAK ÖRNEĞİ

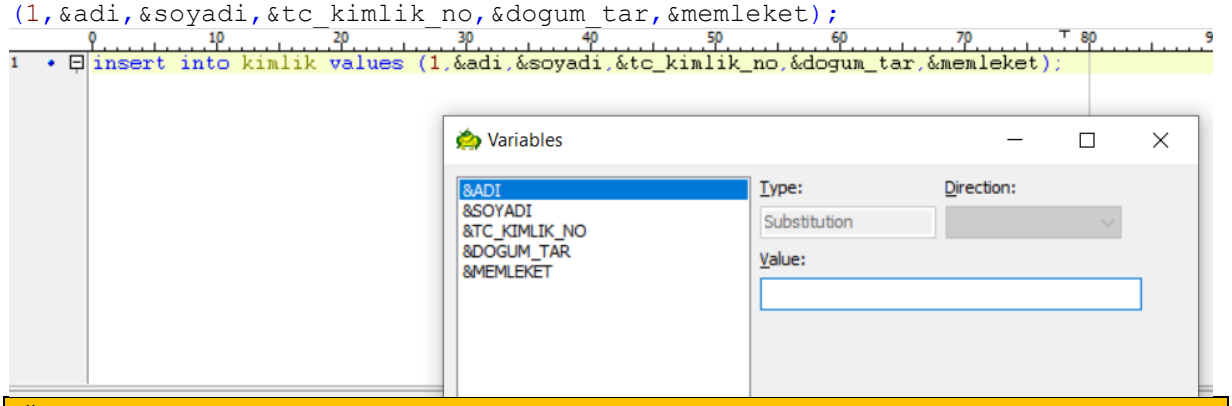
```
insert into kimlik (dosya_no,dogum_tar) values (5,'01.01.2021');
```

TO_DATE TARİH ÇEVİRME ÖRNEĞİ

```
insert into kimlik (dosya_no,dogum_tar) values
(5,to_date('01.01.2021','dd.mm.yyyy'));
```

& YER TUTUCULAR

```
insert into kimlik values
(1,&adi,&soyadi,&tc_kimlik_no,&dogum_tar,&memleket);
```



Örnekte belirtildiği gibi yer tuculara değer atanması yapılır.Yapılan atamalar ise sql cümlesi içinde ilgili kolana yönlendirilir.

Farklı bir **SQL** cümlesini insert komutunuza ekleye bilirsiniz. Kullanım açısından basit insert komutu ile aynı yapıda ilerler. Sadece değerlerini farklı bir **SELECT** komutu kullanarak oluşturur
Kısaca **INSERT** komutu bir alt sorgu ile doldurulur

```
Insert into
Kimlik
(dosya_no, adi, soyadi, tc_kimlik_no, dogum_tar, memleket)
(
Select
dosya_no, adi, soyadi, tc_kimlik_no, dogum_tar, memleket
from yeni_kimlik
)
```

- Örnekte kimlik tablosunda ki 6 kolonun değeri farklı bir tablodaki 6 adet kolon temel alınarak oluşturulmasıdır
- Her iki tablonun da kolon değerleri eşit olarak belirtilmelidir
- **SELECT** sorgusu ile dönen tüm kayıtlar **İNSERT** komutu ile tabloya eklenecektir

UPDATE

UPDATE / güncellemek yâda değiştirmek. Bir Tablodaki verileri belirli bir şart kural dâhilinde yâda tamamını hiçbir şart belirtmeden güncellemek için kullanacağımız komut.

- Tek bir seferde tek bir satırı yâda tamamını güncelleye bilirsiniz
- Şart kullanmak gerektiğinde WHERE komutu kullanarak şart ekleye bilirisiniz
- Sütunları güncellerken alt sorgular kullana bilirsiniz

update	Güncelleme komutu
kimlik	ilgili tablo
Set	set komutu
adi='ali', soyadi='şahan'	İlgili tablodaki ilgili alanlar
where	Şart komutu
dosya_no=5	İlgili şart

ÖRNEK

```
update kimlik set dogum_tar='01.01.2021' where dosya_no=5

update kimlik set dogum_tar=to_date('01.01.2021','dd.mm.yyyy') where
dosya_no=5
```

ALT SORGU İLE GÜNCELLEME

```
update kimlik
set memleket=(select iladi from il where kodu=66)
where dosya_no=5
```

Dosya numarası 5 olan kimlik kaydının memleket bilgisini, il tablosundaki kodu 66 olan iladı ile güncelleme

DELETE

DELETE / tablodan satır silmek. Bir Tablodaki verileri belirli bir şart kural dâhilinde yâda tamamını hiçbir şart belirtmeden silmek için kullanacağımız komut.

WHERE komutu kullanırsanız belirtiğiniz şartı sağlayan satırlar silinirken, WHERE komutu kullanılmaz ise ilgili tablonun tamamını silecektir.

- Tek bir seferde tek bir satırı yâda tamamını sile bilirsiniz

- Şart kullanmak gerektiğinde WHERE komutu kullanarak şart ekleye bilirisiniz
- Sütunları silerken alt sorgular kullana bilirsiniz

ÖRNEK

```
delete kimlik where dosya_no=5;
--Dosya numarası 5 olan kayıtı silme

delete kimlik where memleket=(select iladi from il where kodu=66);

--memleketi il tablosundaki 66 koduna eşit kimlik tablosundaki eşleşen
kayıtlar
```

MERGE

MERGE / ekleme ve güncelleme komutu, 2 farklı tablodaki verileri ortak bir sütun ile kontrol ederek güncellemek yâda silmek kullanacağımız komut

TABLO1 Kimlik Yeni Tablomuz

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1		uygur	29348069760
2	deneme		16564524223
3		duru	55661919654
4	ayça		98734870438
5	miran		75246579987

TABLO2 Kimlik Eski Tablomuz

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	meddata	meddata	12441144687
2	deneme	umit	16915266483
3	hakan	kılıç	70550896722
4	ayça	kılıç	40650782738

ÖRNEK

```
 Senaryo: Kimlik yeni tablomuz da olası bir yanlış update sonucu bazı
verilerimiz silindi,kimlik eski tablosuna referans alarak verileri
güncellememiz gerekiyor

merge into KIMLIK_YENI t using kimlik_eski v
on (t.dosya_no = v.dosya_no)
when matched then
    update set t.adi = v.adi,t.soyadi=v.soyadi
--4 rows merged.

iki tablo dosya_no sütunun referans alarak tablo1 tablosunun daki adi
ve soyadı bilgilerini tablo2 tablosundan güncellemesi yapılıyor
```

TABLO1 sql sonrası | **TABLO2** Kimlik Eski Tablomuz

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	meddata	meddata	29348069760
2	deneme	umit	16564524223
3	hakan	kılıç	55661919654
4	ayça	kılıç	98734870438
5	miran		75246579987

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	meddata	meddata	12441144687
2	deneme	umit	16915266483
3	hakan	kılıç	70550896722
4	ayça	kılıç	40650782738

```
5 nolu satır tablo2 de yer almadığı için güncelleme gerçekleşmedi
```

WHERE

WHERE / sorgularınıza select /update /delete komutlarınıza şart eklemek istediğinizde ,belirli bir kriterde çalışma şartı koyduğunuzda kullanacağımız komut. Where komutunu ile

= Eşittir, <> eşit değildiri **between like in and or** ve **not** mantıksal koşullarını kullana bilirsiniz

ÖRNEKLER

BETWEEN

Between başlangıç ve bitiş değeri verilen değelerin arasında kalan satırları listeler

```
select * from kimlik where dosya_no between 50 and 60
-- Dosya_no numarası 50 ve 60 dahil arasın da  olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
50	muzaffer	cihan	49015892503	1.11.1958	GÜMÜŞHANE
51	ahmet	şapul	82821429108	1.10.1974	KIRIKKALE
52	dilek	keskin	26223403640	28.06.1994	ADANA
53	meliha	dözgün	28842005024	6.01.2006	AKSARAY
54	sabah	sabahg	65339207454	25.04.2020	KOCAELİ
55	nursel	genç	43652290093	1.01.1979	ANTALYA
56	azra ırmak	orhan	20329941738	19.10.2020	ÇANKIRI
57	eymenyagız	büyükkılcı	87913618784	10.02.2017	ÇORUM
58	perihan	alışkan	79778419193	1.08.1965	MALATYA
59	muhammed asaf	işçi	89075293402	12.09.2018	AKSARAY
60	ali rıza	bozaslan	57718451097	1.01.1960	KIRIKKALE

LIKE

LIKE joker karakter olarak tanımlanmış aramalar yapabileceğiniz komut satırı adi like ('%a%') vb. türde aramalar yapabilirsiniz belirtilmiş değelerin eşleşen satırları listeler.
Like içerisinde tek bir koşul eklene bilir Sayı ve karakter kullana bilir
Joker Karakterler
% dahil ediliğinde sonrasının daki değerleri önemsemez

_ bir karakteri belirtir. // adi like ('__ li') ilk 2 karakteri önemli olmayan ama son 2 karakteri li olarak biten adi sütunu temsil eder

```
select * from kimlik where adi like ('ali')
--adi ali olan kimlikler adi='ali' ile aynı sonuç döndürür
```

```
select * from kimlik where adi like ('li%')
--adi li olarak başlayan joker % kullanıp sonrası önemli olmayan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
3761	lina	kanber	65787222837	30.07.2020	KASTAMONU
3087	liya	çekiç	83166383713	31.10.2018	VAN
3511	liya	budak	83802601471	5.06.2020	KÜTAHYA
7459	lidya	besle	12597222020	23.03.2021	BİLECİK
6015	liva berfin	şimşek	82909126415	15.01.2017	HATAY
5426	lina	özkan	79523619436	11.10.2020	ISPARTA

```
select * from kimlik where adi like ('%li%')
--adi içerisin de li olarak eşleşen kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
18	elif	çakır	54713292318	1.01.1966	BİNGÖL
36	ali	ensar	34750136108	1.02.1990	AFYONKARAHİSAR
53	meliha	dözgün	28842005024	6.01.2006	AKSARAY
43	hafsa liya	şahin	62109462895	26.07.2020	ORDU
66	ali	aslan	37247133619	1.01.1982	GAZİANTEP
72	mehmet ali	karaer	56739165291	10.02.1987	MALATYA

```
select * from kimlik where dosya_no like (50)
--Sayı olarak kullanım
```

```
select * from kimlik where adi like ('__li')
-- ilk 2 karakteri önemli olmayan ama son 2 karakteri li  olarak biten adi sütunu temsil eder
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
118	veli	kalkan	84475837879	1.01.1991	MANİSA
406	seli	alaparmak	86459424826	28.07.2015	MERSİN
2240	güli	alp	73129692465	1.01.1966	ANTALYA
14718	veli	bozkurt	73735473415	1.01.1957	TOKAT
14749	güli	nazlıgül	78646747434	4.07.1963	MALATYA
15795	veli	karakuş	50080535146	20.09.1958	KARS

IN

IN liste içinde belirtilmiş değelerin eşleşen satırları listeler.Sayı ve karakter kullana bilir

```
select * from kimlik where dosya_no IN (50,51,52,58,59)
-- Dosya_no numarası sadece 50,51,52,58,59 olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
50	muzaffer	cihan	49015892503	1.11.1958	GÜMÜŞHANE
51	ahmet	şapul	82821429108	1.10.1974	KIRIKKALE
52	dilek	keskin	26223403640	28.06.1994	ADANA
58	perihan	alışkan	79778419193	1.08.1965	MALATYA
59	muhammed asaf	işçi	89075293402	12.09.2018	AKSARAY

```
select * from kimlik where adi in ('dilek','ali')
-- adi dilek ve ali olan tüm kimlik kayıtları
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
36	ali	ensar	34750136108	1.02.1990	AFYONKARAHİSAR
52	dilek	keskin	26223403640	28.06.1994	ADANA
66	ali	aslan	37247133619	1.01.1982	GAZİANTEP
110	ali	güney	27092942967	13.11.1994	MUĞLA
97	ali	yetğin	60773924318	1.01.1958	BURDUR

IS NULL

IS NULL belirtilen sutunun null boş değer olma şartını kapsar

```
select * from kimlik where adi is null
--Adı boş olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
432		şahan	71130151925		ESKİŞEHİR
644			49294494054		KARS
1045			89646131106		ERZURUM
4748			72163751625	2.02.1996	ZONGULDAK
5075			75468351003	19.11.1940	KAYSERİ
10557			17363145407	1.07.1985	ANTALYA

IS NOT NULL

IS NOT NULL belirtilen sutunun null dolu değer olma şartını kapsar

```
select * from kimlik where adi is null and soyadi is not null
--adi boş ve soyadi dolu olan kullanıcı bilgileri
*NOT ifadesi belirtilen şartının tersini ifade eder
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
432		şahan	71130151925		ESKİŞEHİR

AND

AND ve operatörü, birden fazla şart belirtildiğinde her iki şartında doğru değer üretmesinde çalışır.
Her iki şart doğru ise istenen işlemler select delete update gerçekleşir
SQL içinde kullanım sınırı yoktur .1 tane de olabilir 20 tane de kullana bilirsiniz

```
select * from kimlik where adi ='ali' and soyadi ='şahan'
--adi ali olan ve soyadi şahan olan kimlik bilgileri
1.şart adi ='ali'
2.şart soyadi='şahan'
1 ve 2 nolu şart sağlandığın da  elde edilen kimlik bilgisi
```

```
select * from kimlik where adi ='ali' and soyadi ='şahan' and dosya_no=432
and memleket='ESKİŞEHİR'
--and şartında bir sınır yoktur istenildiği kadar sorgularınıza ekleye
bilirsiniz
```

OR

OR veya operatörü, birden fazla şart belirtildiğinde şartlardan sadece birinin doğru /şartı sağlamasında çalışır. Şart doğru ise istenen işlemler select delete update gerçekleşir
SQL içinde kullanım sınırı yoktur. 1 tane de olabilir 20 tane de kullana bilirsiniz

*Belirtilen şartlar (a>1 or a<5) parantez içinde kullanılığında sadece parantez içinde ki şartlardan birinin doğru olması parantez içinde şartı ki sağlar

```
    select * from kimlik where adi ='ali' or  soyadi ='şahan'
    --adı ali olan veya soyadı şahan olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
14942	ali	sezer	42854205875	1.11.1986	KARAMAN
11116	ali	sönmez	88689519990	1.01.1986	KIRKLARELİ
22290	ali	sönmez	58261742350	1.02.1981	SAMSUN
432	ali	şahan	71130151925	1.01.1986	ESKİŞEHİR
19305	aybüke güler	şahan	68664095439	22.05.2021	AFYONKARAHİSAR
464	sevillay	şahan	19735644479	1.01.1982	BURSA
490	hiranur	şahan	25328967582	20.05.2014	OSMANİYE
545	ibrahim	şahan	44723662331	15.12.1992	ŞIRNAK

```
   select * from kimlik where (dosya_no =432 and  adi='ali' ) or
dosya_no=66;
--dosya no 432 ve adi ali olanlar veya dosya no 66 olan kimlik bilgileri
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
66	ali	aslan	37247133619	1.01.1982	GAZİANTEP
432	ali	şahan	71130151925	1.01.1986	ESKİŞEHİR

ORDER BY

ORDER BY / sorgu çıktılarının sıralamasında belirtilen sutunların artan yada azalan düzeyde sıralaması için kullanılır

Basit bir sql cümlesi yazdığınız ama elde ettiğiniz sonuc bir liste düzeninde değil.Örneğin dosya numarasına göre küçükten büyüğe doğru sıralamanız gerekiyor

ASC :Artan düzeyde sıralama yapar

DESC :Azalan düzeyde sıralama yapar

Sıralama yapılırken tek bir sütun yada daha fazlası kullanına bilir order by dosya_no,adi ASC vb.

ÖRNEKLER

```
select * from kimlik  order by dosya_no
```

dosya no sütunu varsayılan format DESC küçükten büyüğe doğru sıralar

Varsayılan olarak ASC değerini alır yani artan düzeyde sıralar

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
1	meddata	meddata	12441144687	1.04.1991	UŞAK
2	deneme	umit	16915266483	1.01.1991	ERZİNCAN
3	hakan	kılıç	70550896722	17.03.1986	ÇORUM
4	ayça	kılıç	40650782738	12.06.1992	EDİRNE
6	funda	uygur	29348069760	7.07.2001	BİNGÖL
7	burhan	duru	55661919654	1.11.1992	BURDUR
8	funda	uygur	16564524223	7.07.2001	KIRŞEHİR
9	beren	kılıç	98734870438	3.06.2016	ESKİŞEHİR
10	miran	kılıç	75246579987	16.08.2020	AĞRI

```
select * from kimlik  order by dosya_no  desc
```

dosya no alanını azalan düzeyde listele
En büyük dosya numarasından en küçük dosya numarası gore sıralar

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
23876	ayşe gül		11429019623	30.12.2000	RİZE
23875	feray	terlemez	81824072798	2.06.1986	ARTVİN
23874	hatice	orak	17185176292	14.01.1989	ADANA
23873	adnan eymen	kütük	29096267922	25.02.2015	ÇANAKKALE
23872	avaş	kaya	39281048365	7.03.1958	HATAY
23871	elif	uras	53512765043	18.12.1991	EDİRNE
23870	zeynep	turan	30703249132	24.07.2008	SİİRT
23869	samet	kurtoğlu	10300029136	5.01.2012	DİYARBAKIR

```
select * from kimlik order by 6;
```

3 numaralı sütun değerine göre ASC komutunu uygular

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
5	miran		75246579987	16/08/2020	AĞRI
1	meddata	meddata	29348069760	7/07/2001	BİNGÖL
3	hakan	kılıç	55661919654	1/11/1992	BURDUR
4	ayça	kılıç	98734870438	3/06/2016	ESKİŞEHİR
2	deneme	umit	16564524223	7/07/2001	KIRŞEHİR

```
select * from kimlik order by adi, soyadi asc;
select * from kimlik order by 2,3 asc;
```

Her iki komutta aynıdır.Biri sütun ismi ile değeri ise sütun numarası ile sıralama yapar

GROUP BY

GROUP BY / sorgu çıktılarının satır başına tek bir sonuç üretmesi, aynı değerlerin gruplanması için kullanılır. AVG• COUNT • MAX • MIN • STDDEV • SUM • VARIANCE grup fonksiyonlarını kullanarak sorgularınızı zenginleştire bilirsiniz

ÖRNEK

Kasadetail tablosunda yer alan ve tarih olarak 02.05.2021 ile 03.05.2021 tarihleri arasında hareket görmüş kasaların kasa numarasına göre gruplamasını yapalım.

```
select * from kasadetail
where tarih between
to_date('02.05.2021','dd.mm.yyyy') to_date('03.05.2021','dd.mm.yyyy')
```

Öncelikle gruplama yapılmadığında elde ettiğimiz sonuç

KASA_NO	TARIH	DOSYA_NO	ODEME_TIPI	UCRET	VADE	GIRIS_CIKIS	ACIKLAMA
1808	2/05/2021 02:51:45	7790	K	60	3/05/2021	G	
1808	2/05/2021 03:04:15	7791	P	120	2/05/2021	G	
1808	2/05/2021 03:08:59	7791	P	8,87	2/05/2021	G	
1808	2/05/2021 03:18:47	7792	K	60	3/05/2021	G	
1808	2/05/2021 03:42:34	7793	P	60	2/05/2021	G	
1808	2/05/2021 04:45:43	7794	K	60	3/05/2021	G	
1808	2/05/2021 06:29:58	7795	P	60	2/05/2021	G	
1808	2/05/2021 07:40:27	7797	P	0	2/05/2021	G	Ücret=0 işlem Kapatma
1808	2/05/2021 07:42:49	7795	P	195	2/05/2021	G	

Şimdi ise kasa numarasına göre gruplamasının kodu yazalım

```
select kasa_no, sum(ucret) from kasadetail
where tarih between
to_date('02.05.2021','dd.mm.yyyy') and to_date('03.05.2021','dd.mm.yyyy')
 group by kasa_no order by sum(ucret) desc
```

Şimdi ise kasa numarası baz alınarak önce gruplama yapıldı sonuç elde ede bilmek içinde ücret sütunu sum() komutu ile toplandı ve order by ile büyükten küçüğe doğru sıralaması yapıldı elde edilen sonuç ;

KASA_NO	SUM(UCRET)
1808	1305,93
1809	990
1810	950
99	0

Sum(ucret) için takma isim kullana bilir
Bunun için yapılması gereken;

```
select kasa_no, sum(ucret) Toplam from
kasadetail
where tarih between
to_date('02.05.2021','dd.mm.yyyy') and
to_date('03.05.2021','dd.mm.yyyy')
 group by kasa_no order by sum(ucret) desc
```

HAVING

HAVING komutu **GROUP BY** ile gruplanmış verilerinin kısıtlanması için kullanılır. AVG, SUM, MIN, MAX, COUNT değerini having komutu ile bir sütuna bağlayarak group by için kısıtlama yapabilirsiniz

ÖRNEK

Kasadetail tablosunda kasa_no sütuna göre gruplanıp ucret sütunu max değeri 9500 TL ‘den yüksek olan ücrete göre büyükten küçüğe doru sıralanan SQL

```
select kasa_no ,sum(ucret) Toplam from kasadetail
 group by  kasa_no having max(ucret)>9500 order by sum(ucret) desc
```

`select kasa_no ,sum(ucret) Toplam from kasadetail`	*Normal select sorgumuz*
`group by  kasa_no`	*Kasa_no sütuna göre gruplama fonksiyonu*
`having max(ucret)>9500`	*Gruplanan veri içersinde maksimum ücreti 9500 TL ‘den büyük*
`order by sum(ucret) desc`	*Ücrete göre büyükten küçüğe doğru sıralanmış veri*

DEFINE

DEFINE komutu ile SQL leriniz için değer oluştura bilir ve atama işlemleri yapabilirsiniz. SQL’lerinize değişken eklemek için DEFINE kaldırmak için ise UNDEFINE komutları kullanılır. Tanımladığınız değişkeni SQL içine çağırmak için ise &isim kullanılır
Kullandığınız editöre göre değerlere veri aktarma ekranı farklılık göstere bilir
SQLPLUS , **TOAD vb.**

ÖRNEK

```
define aranan dosya =95
select * from kimlik  where dosya_no>&aranan_dosya;
 undefine aranan_dosya
```

arana_dosya adında bir değer oluşturuldu ve değer olarak 95 atandı.
Where komutu kullanarak dosya_no> atanan değere göre sorgu çalıştırıldı
arana_dosya değeri tekrar bırakıldı

VERIFY

VERIFY atama yapılan değerleri kontrol etmek ve kullanılan sql içerisinde doğrulamak için kullanılır

```
SET VERIFY ON
define aranan_dosya =95
select *  from kimlik  where dosya_no=&aranan_dosya;
 undefine aranan_dosya
```

DESCRIBE

DESCRIBE komutu ile bir tablonun yapısını listeye bilirsiniz. Seçili tabloların sütun adı ,sütunun tipi hakkında bilgiler txt olarak listelenecektir

```
describe kimlik
```

Kimlik tablosuna bilgi

```
TABLE kimlik
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DOSYA_NO                                           NUMBER
 ADI                                                VARCHAR2(50)
 SOYADI                                             VARCHAR2(50)
 TC_KIMLIK_NO                                       NUMBER
 DOGUM_TAR                                          DATE
 MEMLEKET                                           VARCHAR2(16)
```

DÖNÜŞTÜRME FONKSİYONLARI

Karakter dönüştürme fonksiyonları LOWER “Küçük harfe çevir ” UPPER “Küçük harfe çevir” INITCAP “yazım denetimi” fonksiyonları kullana biliriz

LOWER

```
LOWER komutu karakter dizileri için dönüşüm sağlar. Belirtilen string
/karakter dizisini küçük harfe çeviri
```

```
select lower ('ali ŞAHAN.com ') sonuc from dual
```

SONUÇ: ali şahan.com

```
select lower ('ALİ ŞAHANNN.deneme') sonuc from dual
```

SONUÇ: ali şahannn.deneme

UPPER

UPPER komutu karakter dizileri için dönüşüm sağlar. Belirtilen string /karakter dizisini büyük harfe çeviri

```
select upper ('alisahan') sonuc from dual
```

SONUÇ: ALISAHAN

```
select upper ('ali şahan ile oracle SQL öğreniyorum') sonuc from dual
```

SONUÇ: ALI ŞAHAN ILE ORACLE SQL ÖĞRENIYORUM

INITCAP

INITCAP komutu karakter dizileri için dönüşüm sağlar. Belirtilen string /karakter dizisine yazım kuralı uygular. Kelimelerin ilk harfi büyük diğer harfler küçük olacak şekilde sonuç üretir

```
select  initcap('SQL DÖNÜŞTÜRME FONKSİYONLARI') sonuc  from dual
```

SONUÇ: Sql Dönüştürme Fonksiyonlari

DEĞİŞTİRME FONKSİYONLARI

Karakter işleme / değiştirme fonksiyonları

CONCAT " Birleştirme", SUBSTR "Kesme ", LENGTH "Uzunluk ", INSTR " İçinde Arama ", LPAD " Soldan Maskeleme", RPAD " Sağdan Maskeleme " , REPLACE "Değiştir " TRIM" Çıkarma" komutları olarak karakter değiştirme fonksiyonları olarak kullana bilirsiniz

CONCAT

CONCAT Birleştirme komutu, iki farklı kelimenin yâda sütunun birleştirmesi için kullana bilir

```
select concat(adi, soyadi) sonuc  from kimlik where dosya_no=42
```

SONUÇ: Nedimekorkmaz

SUBSTR

SUBSTR kesme komutu, bir kelime kümesi içinden başlangıç ve bitiş noktaları belirtilerek kesme işlemi için kullanılır

```
select concat('ali', 'şahan') sonuc  from dual
```

SONUÇ: alişahan

LENGTH

LENGTH, belirtilen kelimenin karakter sayısını almak için kullanılır.

```
select length('alişahan') sonuc  from dual
```

SONUÇ: 8

INSTR

INSTR, kelime içerisin de farklı bir kelimemin başlangıç noktasını verir.

```
select instr('alişahan','ş') sonuc  from dual
```

SONUÇ: 4

LPAD

LPAD, belirli bir karakter kümesinin sol tarafını belirtilen karakter ile doldurmak için kullanılır

```
select lpad('alişahan',9,'*') sonuc  from dual
```

*SONUÇ: *alişahan*

RPAD

RPAD, belirli bir karakter kümesinin sağ tarafını belirtilen karakter ile doldurmak için kullanılır

```
select rpad('alişahan',9,'*') sonuc  from dual
```

*SONUÇ: alişahan**

TRIM

TRIM, belirli bir karakter kümesinin eşleştiği karakterleri ilk belirtilen karakter kümesinin kaldırılması için kullanılır

```
select trim('a' from 'alisahan') sonuc  from dual
```

SONUÇ: lisahan

```
//Kesme işlemini Sağdan Başlat trailing
select trim(trailing 'n' from 'alisahan') sonuc  from dual
```

SONUÇ: alisaha

```
//Kesme işlemini Soldan Başlat leading
select trim(leading 'a' from 'alisahan') sonuc  from dual
```

SONUÇ: alisaha

REPLACE

REPLACE, belirli bir karakter kümesinin eşleştiği karakterleri farklı bir karakter kümesi ile değiştirilmesi için kullanılır

```
select replace('ali şahan','şahan','öztürk') sonuc  from dual
```

```
SONUÇ:  ali öztürk
```

RAKAMSAL FONKSİYONLARI

Sayı işleme fonksiyonları ROUND " Ondalık Sayıları Yuvarlama" TRUNC " Değeri belirtilen ondalık sayları keser " MOD "Bölmenin kalanını döndürür " sayı fonksiyonları olarak kullana bilirsiniz

ROUND

ROUND Ondalık Sayıları Yuvarlama ve kesme fonksiyonu olarak kullanılır, Ondalıklı sayıları belirtilen hane olarak önce yuvarlar sonra keser

```
select ROUND(45.926587, 3) sonuc  from dual
```

```
SONUÇ:  45,927
```

```
select ROUND(45.926587, 2) sonuc  from dual
```

```
SONUÇ:  45,93
```

TRUNC

TRUNC Ondalık Sayıları sadece kesme fonksiyonu olarak kullanır, Ondalıklı sayıları belirtilen haneden itibaren keser

```
select trunc(45.926587, 2) sonuc  from dual
```

```
SONUÇ:  45,927
```

MOD

MOD bir sayının farklı bir sayıya bölündüğünde kalanı verir

```
select mod(100, 6) sonuc  from dual
```

```
SONUÇ:  4
```

TARİH FONKSİYONLARI

Tarih değerleri ile işlem yapmak, gün ay yıl vb. formatlarda ekleme çıkarma yapmak görüntüleme formatını değiştirmek için tarih formatlarını **SYSDATE, MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC,** kullanılırız.

SYSDATE

SYSDATE, system bilgisini kullanarak şuan ki, tarih ve saatini döndüren fonksiyondur

To_date('01.01.2021') kullanarak ekleme çıkarama +/- işlemleri kullanıla bilir

```
select sysdate from dual
```

SONUÇ: 19/11/2021 10:38:26

```
select (SYSDATE-to_date('01.01.2021'))/7 AS WEEKS from dual
```

Yılın Kaçıncı Haftası

SONUÇ: 46,0639880952381

MONTHS_BEWEEN

MONTHS_BETWEEN, belirtilen iki tarih arasında ki ay farkını verir

```
select months_between (SYSDATE,to_date('01.01.2021')) AS ay from dual
```

01.01.2021 'den başlayıp 19.11.2021 tarihine göre aradaki ay farkı

SONUÇ: 10,5951982526882

```
select trunc(months_between (SYSDATE,to_date('01.01.2021'))) AS AY from dual
```

Trunc komutu ile sayı olarak alına bilir

SONUÇ: 10

ADD_MONTHS

ADD_MONTHS belirtilen tarihe ay eklemek için kullanılır

```
select add_months ('01.01.2021',2) AS sonuc from dual
```

SONUÇ: 1/03/2021

NEXT_DAY

NEXT_DAY, belirtilen tarihin haftadaki gün sayısı

1	Pazartesi	2	Salı	3	Çarşamba	4 Perşembe
5	Cuma	6	Cumartesi	7	Pazar	

Örneğin 20.11.2021 için 2 Günü sorguladığınızda o haftanın 2. Gününün tarihi verir

```
select next_day ('20.11.2021',2) AS sonuc from dual
```

SONUÇ: 23/11/2021

LAST_DAY

LAST_DAY, belirtilen tarihin ayının son gününü verir

```
select last_day ('20.11.2021') AS sonuc from dual
```

11. Ayın Son Günü

SONUÇ: 30/11/2021

ROUND

ROUND, belirtilen tarihi yuvarlar işlemini gerçekleştirir

```
select ROUND(to_date('16.11.2021'),'MONTH')  AS sonuc from dual;
```

SONUÇ: 1/12/2021

```
SELECT ROUND(TO_DATE('15.11.2021'),'MONTH')  AS SONUC FROM
DUAL;
```

SONUÇ: 1/11/2021

Ayı baz alarak ayın ortasını geçmiş ise sonraki ayın ilk gününe ,ayın ortasında önce ise geçerli ayın il gününü eşit kabul eder
Baz alınan tarih öncesi >15 16<sonrası

Aynı döngü yıl içinde geçerlidir. Yılın ortası Kabul edilen 01 Temmuz öncesini geçerli yılın ilk gününe , Sonra zaman dilimi ise sonraki yılın ilk gününe eşit kabul edilir

```
SELECT ROUND(TO_DATE('30.06.2021','DD.MM.YYYY
HH24:MI'),'YEAR')  AS SONUC FROM DUAL;
```

SONUÇ: 1/01/2021

```
SELECT ROUND(TO_DATE('01.07.2021','DD.MM.YYYY
HH24:MI'),'YEAR')   AS SONUC FROM DUAL;
```

SONUÇ: 1/01/2022

TRUNC

TRUNC, belirtilen tarihin geçerli zaman diliminde ilk gününü verir. Ay yâda yıl bazında belirtilen tarihin ilk günü

```
SELECT TRUNC(TO_DATE('30.11.2021'),'MONTH')  AS SONUC FROM
DUAL;
```

SONUÇ: 1/11/2021

```
SELECT TRUNC(TO_DATE('30.11.2021'),'YEAR')  AS SONUC FROM
DUAL;
```

SONUÇ: 1/01/2021

DÖNÜŞÜM FONKSİYONLARI

Oracle veri tiplerini farklı formatlara dönüştüre bilirsiniz. Dolaylı dönüşüm fonksiyonları

TO_CHAR TO_DATE TO_NUMBER

TO_CHAR

TO_CHAR tek tırnak içinde belirtilen metin, tarih ve sayıları string formatında görüntülemek için kullanılır

```
SELECT to_char(to_date('15.11.2021'),'YYYY')  AS SONUC FROM DUAL;
```

SONUÇ: 2021

To_date() olarak verilen tarihi belirtilen formatta char türünde dönüştürür

KOMUT	SONUÇ
YYYY	Yıl Bilgisini rakam olarak verir
YEAR	Yıl Bilgisini Yazı olarak verir
MM	AY Bilgisini rakam olarak verir

MONTH	AY Bilgisini Yazı olarak verir
MON	AY Bilgisini Kısa olarak Yazı formatında verir
DAY	GÜN Bilgisini Yazı olarak verir
DY	GÜN Bilgisini Kısa olarak Yazı formatında verir

```
SELECT to_char(to_date('15.11.2021'),'YYYY')  AS SONUC FROM DUAL;
```

```
SONUÇ:  2021
```

```
SELECT to_char(to_date('15.11.2021'),'YEAR')  AS SONUC FROM DUAL;
```

```
SONUÇ:  TWENTY TWENTY-ONE
```

```
SELECT to_char(to_date('15.11.2021'),'MONTH')  AS SONUC FROM DUAL;
```

```
SONUÇ:  KASIM
```

```
SELECT to_char(to_date('15.11.2021'),'DAY')  AS SONUC FROM DUAL;
```

```
SONUÇ:  PAZARTESİ
```

TO_NUMBER

TO_NUMBER karakter dizesini sayıya çevirmek için kullana bilirsiniz

KOMUT	SONUÇ
9	Bir sayıyı temsil eder
0	Bir sıfırın görüntülenmesini zorlar
$	Kayan bir dolar işareti yerleştirir
L	Değişken yerel para birimi simgesini kullanır
.	Bir ondalık nokta yazdırır
,	Binlik göstergesi olarak virgül yazdırır

```
TO_NUMBER(char[, 'format_model'])
```

TO_DATE

TO_DATE karakter dizesini tarihe çevirmek için kullana bilirsiniz

```
SELECT TO_CHAR(sysdate, 'dd.mm.yyyy')  AS sonuc FROM dual;
```

```
19.11.2021
```

```
SELECT TO_CHAR(sysdate, 'DD-Mon-YYYY')  AS sonuc FROM dual;
```

```
19-Kas-2021
```

KOMUT	SONUÇ
MM	Sayısal Yıl Bilgisi Üretir
MON	Kısaltılmış Ay Bilgisi Üretir
MONTH	Uzun Ay Bilgisi
DD	Ay günü
DY	Günün kısaltılmış adı
YYYY	4 basamaklı yıl
YY	Yılın son 2 hanesi
HH	Günün saati
HH24	Günün saati
MI	Dakika
SS	İkinci

GENEL FONKSİYONLARI

Oracle veri türlerini kontrol ederek, boş yada dolu olmasına göre sonuç üretene fonksiyonlardır.

NVL, NVL2, NULLIF, COALESCE olarak kullana bilirsiniz

NVL

NVL, boş bir değeri gerçek bir değere dönüştürür. Kullanıldığı veri türleri tarih karakter ve sayı olabilir. Genel mantığı okuduğum değer boş ise bunu göster formatında ilerler

```
NVL(OKUNANDEGER,BOS ISEGOSTERILECEK DEGER)
```

```
NVL(tutar,1)                    //tutar boş gelir ise 1 al
NVL(dogum_tarihi,'01.01.2021')//doğum tarihi boş ise 01.01.2021
NVL(adi,'İsim Yok')             //adi boş ise isim yok yaz
```

```
SELECT A.*, NVL(A.DOGUM_TAR,'01.01.2021') AS "DOGUM TARIHI" FROM KIMLIK A
WHERE A.DOSYA_NO<5
```

Dosya numarası 5 küçük olan kimlik listesin de doğrum tarihi boş ise 01.01.2021 yazan sql

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	Dogum Tarihi
1	ali	şahan	12441144687	1/04/1991	UŞAK	1/04/1991
2	deneme	umit	16915266483		ERZİNCAN	1/01/2021
3	hakan	kılıç	70550896722		ÇORUM	1/01/2021
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	12/06/1992

NVL2

NVL2, aldığı değeri okur boş yâda dolu olmasına göre farklı sonuçlar üretir. NLV komutu sadece boşu kontrol ederken NVL2 boş için ayrı dolu için ayrı bir sonuç ürete bilir

```
nvl2(ifade1,'dolu ise','boş ise')
```

```
select a.*, nvl2(a.dogum_tar,'DOLU','BOŞ') as "Dogum Tarihi" from kimlik a
where a.dosya_no<5
```

Dosya numarası 5 küçük olan kimlik listesin de doğrum tarihi boş ise boş dolu ise dolu yazan sql

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	Dogum Tarihi
1	ali	şahan	12441144687	1/04/1991	UŞAK	DOLU
2	deneme	umit	16915266483		ERZİNCAN	BOŞ
3	hakan	kılıç	70550896722		ÇORUM	BOŞ
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	DOLU

NULLIF

NULLIF, belirtilen 2 adet ifadeyi karşılaştırır. İfadeler birbirine eşit ise null eşit değil ise birinci ifadeyi döndürür. Dikkat edilmesi gerek nokta her iki ifadenin de aynı türde veri tipi içermesidir. İfade 1 sayı ise,ifade2 de sayı olmak zorundadır

NULLIF(ifade1, ifade2)

```
select a.*,nullif(a.avans,a.maas) nullif from kimlik a where a.dosya_no<=5
```

avans ve maaslar karşılatırır, eşit olanlar null değer eşit olmayanlar ise 1. Ifade olan avans değerini döndüren sql

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS	NULLIF
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000	100
2	deneme	denem	16915266483		ERZİNCAN	0	5000	0
3	hakan	kılıç	70550896722		ÇORUM	7500	7500	
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500	

COALESCE

COALESCE, listede belirtilen tüm ifadeleri kontrol eder ve ilk boş olmayan ifadeyi döndürür. Eğer listedeki tüm ifadeler boş ise kendi null olarak tanımlanan değerini döndürür

COALESCE(ifade1,ifade2,ifade3….. ,'Boş Değer Tanımı')

```
select a.*,coalesce(a.avans,a.maas,50) coalesce from kimlik a where
a.dosya_no<=5
```

Avans ve maaş bilgisini control ederek her iki ifade de boş ise tanımlanmış değeri döndürecek

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS	COALESCE
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000	100
2	deneme	denem	16915266483		ERZİNCAN			50
3	hakan	kılıç	70550896722		ÇORUM	7500	7500	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500	97500

KOŞULLU İFADELER

Programla dillerinde kullandığımız IF THEN ELSE mantığının kullanmasını sağlayan fonksiyonlardır.

CASE ve DECODE fonksiyonları olarak kullana bilirsiniz

CASE

CASE, belirtilen ifade değerini baz alarak eklenen şartları sırası ile kontrol eder. Eğer ifade ile belirtilen şart eşleşir ise diğerlerine bakmadan aldığı sonuç ile görevini tamamlar. Kod Mantığı

```
CASE [ ifade ]

  WHEN şart_1 THEN sonuc_1
  WHEN şart_2 THEN sonuc_2
  …
  WHEN şart_n THEN sonuc_n

  ELSE sonuç //şartların dışında kalan sonu.

END
```

Farklı Türlerden Kullanıla bilir Aynı SQL değerinin farklı yazılımları

```
SELECT A.*,
      CASE A.AVANS
      WHEN 100 THEN '100 AVANS ALDI'
      WHEN 150 THEN '150 AVANS ALDI'
      WHEN 7500 THEN '7500 AVANS ALDI'
      ELSE 'ALMADI'
      END
WHEN
  FROM KIMLIK A WHERE A.DOSYA_NO<=5
```

2.Yöntem

```
SELECT A.*,
      CASE
      WHEN A.AVANS=100 THEN '100 AVANS ALDI'
      WHEN A.AVANS=150 THEN '150 AVANS ALDI'
      WHEN A.AVANS=7500 THEN '7500 AVANS ALDI'
      ELSE 'ALMADI'
      END
WHEN
  FROM KIMLIK A WHERE A.DOSYA_NO<=5
```

3.Yöntem

```
SELECT A.*,
      CASE
      WHEN A.AVANS<=100 THEN '100 AVANS ALDI'
      WHEN A.AVANS<=150 THEN '150 AVANS ALDI'
      WHEN A.AVANS>=7500 THEN '7500''DEN BÜYÜK AVANS ALDI'
      ELSE 'ALMADI'
      END
WHEN
  FROM KIMLIK A WHERE A.DOSYA_NO<=5
```

DECODE

DECODE, CASE benzen şekilde ilerlese de CASE kadar detaylı değildir. İlk belirtilen veri türü hangi türde ise sonraki verilen tüm değerler aynı veri türüne dönüştürülür. Case farklı ifadeler karşılaştırıla bilirken a>b vb. türde DECODE bu yapı mevcut değildir

```
SELECT A.*,
DECODE (A.AVANS,
’100’,'100 AVANS ALDI',
'150','150 AVANS ALDI',
'7500','7500''DEN BÜYÜK AVANS ALDI',
'','ALMADI') DECO_SQL
  FROM KIMLIK A WHERE A.DOSYA_NO<=5
```

GROUP FONKİYONLARI

Group fonksiyonları gruplama yapılan SQL'lerde satır başına veri üretecek şekilde sonuçlar üreten fonksiyonlardır.

- AVG "Ortalama Değer"
- COUNT "Adet Sayısı"
- MIN "Minimum Değer"
- VARIANCE "Varyant Değeri"
- SUM "Toplama"
- MAX "Maksimum Değer"
- STDDEV "Standart Sapma"
- DISTINCT "Varyant Değeri"

AVG

AVG, fonksiyonu belirtilen sütun adının ortalama değerini döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
SELECT DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS,
AVG(UCRETI) ORTALAMA
FROM KIMLIK_AVANS
GROUP BY
DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	ORTALAMA
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	49,5055555555556
3	hakan	kılıç	70550896722		ÇORUM	7500	17,8510891089109
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	4,46309523809524
2	deneme	denem	16915266483		ERZİNCAN		0

SUM

SUM, fonksiyonu belirtilen sütun adının toplam değerini döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
select dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas,
sum(ucreti) Toplam
from kimlik_avans
group by
dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	TOPLAM
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	445,55
3	hakan	kılıç	70550896722		ÇORUM	7500	1802,96
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	187,45
2	deneme	denem	16915266483		ERZİNCAN		0

MIN

MIN, fonksiyonu belirtilen sütun en küçük değerini döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
select dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas,
min(ucreti) MinimumDeger
from kimlik_avans
group by
dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	MINIMUMDEGER
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	0
3	hakan	kılıç	70550896722		ÇORUM	7500	0
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	0
2	deneme	denem	16915266483		ERZİNCAN		0

MAX

MAX, fonksiyonu belirtilen sütun en büyük değerini döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
select dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas,
max(ucreti) MaksimumDeger
from kimlik_avans
group by
dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	MAKSIMUMDEGER
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	161,38
3	hakan	kılıç	70550896722		ÇORUM	7500	80,69
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	47,52
2	deneme	denem	16915266483		ERZİNCAN		0

COUNT

COUNT, fonksiyonu belirtilen sütun adının toplam adet değerini döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
SELECT DOSYA_NO, ADI, SOYADI, TC_KIMLIK_NO, DOGUM_TAR, MEMLEKET, MAAS,
COUNT(UCRETI) TOPLAM
FROM KIMLIK_AVANS
GROUP BY
DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	TOPLAM
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	9
3	hakan	kılıç	70550896722		ÇORUM	7500	101
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	42
2	deneme	denem	16915266483		ERZİNCAN		17

DISTINCT

DISTINCT, fonksiyonu belirtilen sütun adının yenilenen satırları kaldırmak için kullanılır. Select ifadesinde kullanılan bir fonksiyondur

```
SELECT
DISTINCT  DOSYA_NO
DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS

FROM KIMLIK_AVANS
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000
3	hakan	kılıç	70550896722		ÇORUM	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500
2	deneme	denem	16915266483		ERZİNCAN	

VARIANCE

VARIANCE, fonksiyonu belirtilen bir dizi sayının varyansını döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
SELECT
```

```
DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS,
STDDEV(MAAS) OVER (ORDER BY MAAS) STDDEV

FROM KIMLIK_AVANS
GROUP BY
DOSYA_NO,ADI,SOYADI,TC_KIMLIK_NO,DOGUM_TAR,MEMLEKET,MAAS
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	STDDEV
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	0
3	hakan	kılıç	70550896722		ÇORUM	7500	1060,66017177982
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	52399,9045800658
2	deneme	denem	16915266483		ERZİNCAN		52399,9045800658

STDDEV

STDDEV, fonksiyonu belirtilen sütun adının standart sapmasını döndürür. GROUP BY ifadesinden kullanılan fonksiyonlardandır

```
select

dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas,
variance(maas)  variance

from kimlik_avans
group by
dosya_no,adi,soyadi,tc_kimlik_no,dogum_tar,memleket,maas
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	MAAS	VARIANCE
1	ali	şahan	12441144687	1/04/1991	UŞAK	6000	0
3	hakan	kılıç	70550896722		ÇORUM	7500	0
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	0
2	deneme	denem	16915266483		ERZİNCAN		

BİRLEŞİM SQL'LERİ

Çoklu tabloları listelemede birleşim SQL lerini kullana bilirsiniz. Birden fazla tablodan veri sorgulamak için birleştirme kullanmanız gerekmektedir.

Tablo oluşturup, SQL yazarken dikkat etmeniz gereken püf noktalar;

- Tablolarınızı oluşurken benzersiz sütun alanları kullanın
  (Bu birleşim SQL'leri yazarken size kolaylık ve hız katacaktır)

- Tam tablo adı önekleri yerine tablo takma adlarını kullanın.
  (Select komutlarında takma isim kullanmak tablolaya ait verileri listelerken size kolaylıklar sağlar)

- Tablo diğer adı, tabloya daha kısa bir ad verir
- SQL kodunu daha küçük tutar, daha az bellek kullanır
- Aynı sütunları ayırt etmek için sütun takma adlarını kullanın:

Doğal Birleşim	**USING clause ON clause**	
Kendi kendine katılma	**Selft Join**	
Eşsiz birleşimler	**Nonequijoins**	
DIŞ Birleşimler	**OUTER join**	**(Left Right Full QUTER**

DOĞAL BİRLEŞİM

USING ve ON komutları ile 2 farklı tablonun doğal birleşimin sağlaya bilirsiniz.2 farklı tablodan eşit değerlere sahip satırları seçer. Eşitlenen sütunlar farklı veri tiplerine ait ise hata verir

NATURAL JOIN

NATURAL JOİN, birleştirilen tabloların içerisinde bire bir tüm kolanların eşit olduğu benzer satırları getirir

```
select * from hasta A  NATURAL join hasta2
```

- Aranan tablo adi hasta
- Birleştirilmesi istenen tablo adı hasta2
- Birleşim sütunu ise tüm alanların eşit olması
- Natural birleşimlerde tablo takma isimlerini kullanamazsınız

ÖRNEK

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1
3	hakan	kılıç	70550896722		ÇORUM	7500	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500

HASTA TABLOSU

Yukarıda ki tablo HASTA tablomuza ait verileri içeriyor

Aşağıda ki tablo HASTA2 tablomuza ait verileri içeriyor

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1
3	hakan	kılıç	70550896722	21/11/2021	ÇORUM	7500	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500

HASTA2 TABLOSU

Her ikin tablonun NATURAL JOIN kullanarak birleşim ise aşağıda verilmiştir

HASTA	select * from hasta a natural join hasta2 b	HASTA2

Birleşim sonucu oluşan veride dosya numarası 3 olan hasta birebir eşleşmediği için birleşime alınmadı. Sütunları birebir eşit olanlar ise birleşimde yer aldı

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500

USING

USING ile birleşimlerde belirtilen bir sütun eşlemesi yeterli olacaktır. Belirtilen sütun adı her iki tabloda da olmalı ve aynı veri türünde tanımlanmalıdır.NATURAL JOIN 'de tüm sütunların eşlenmesi istenirken USING JOIN de belirtilen sütunun eşitlenmesi size benzer satırları getirecektir.

SQL KALIBI

```
SELECT * FROM TABLO1 JOIN TABLO2 USING(SÜTUN_ADI)
```

ÖRNEK

Yukardaki örnek baz alındığında oluşan SQL alta belirtilmiştir

```
select * from hasta  join hasta2 using (dosya_no)
```

- Aranan tablo adi hasta
- Birleştirilmesi istenen tablo adı hasta2
- Birleşim sütunu ise her iki tabloda mevcut olan dosya_no
- USING birleşimlerde tablo takma isimlerini kullanamazsınız

Elde Edilen Sonuç

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS	ADI_1	SOYADI_1	TC_KIMLIK_NO_1	DOGUM_TAR_1	MEMLEKET_1	AVANS_1	MAAS_1
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1
3	hakan	kılıç	70550896722		ÇORUM	7500	7500	hakan	kılıç	70550896722	21/11/2021	ÇORUM	7500	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500

HASTA HASTA2

ON

JOIN ON, birleşiminde birleşim şartlarınızı tek bir sütun üzerinden değil 2 farklı sütunun eşleştirilmesi ile birleşim kura bilirsiniz

JOIN ON SQL KALIBI

SELECT * FROM TABLO1 A JOIN TABLO2 B ON (A.SUTUN_ADI=B. SUTUN_ADI)

ÖRNEK

```
select * from hasta a   join hasta2 b  on (a.dosya_no=b.dosya_no)
```

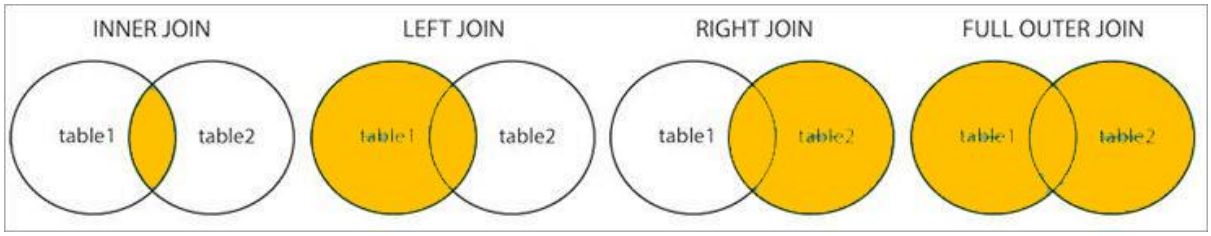
- Aranan tablo adi hasta
- Birleştirilmesi istenen tablo adı hasta2
- Birleşim sütunu ise her iki tabloda mevcut olan dosya_no
- USING birleşimlerde tablo takma isimlerini kullana bilirsiniz

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET	AVANS	MAAS	ADI_1	SOYADI_1	TC_KIMLIK_NO_1	DOGUM_TAR_1	MEMLEKET_1	AVANS_1	MAAS_1
1	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000	ali	şahan	12441144687	1/04/1991	UŞAK	100	6000
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1	deneme	denem	16915266483	21/11/2021	ERZİNCAN	100	1
3	hakan	kılıç	70550896722		ÇORUM	7500	7500	hakan	kılıç	70550896722	21/11/2021	ÇORUM	7500	7500
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500	ayça	kılıç	40650782738	12/06/1992	EDİRNE	97500	97500

-

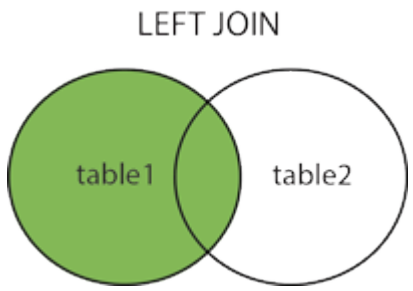
DIŞ BİRLEŞİMLER

LEFT ,RIGHT ve Full JOIN türlerinin kullanıldığı birleşim türüne verilen addır. Eşleşen satırları ve eşleşmeyen satırları da join türüne göre getirebilir. İç birleşimden eşleşmeyen satırlar mantığı gereği farklıdır



LEFT JOIN

LEFT Join "SOL DIŞ KATILMA" sorgulanan 2 tabloda SOLDA olan tablo tamamen katılırken, sağda olan tablo ise sadece eşleşenler katılır



Tablo 1 otomatik olarak dâhil olur.

Tablo 2 ise sadece tablo 1 ile eşleşen kayıtları üzerinden sorguya dâhil olur

Oluşan Küme resmi solda belirtilmiştir

LEFT OUTER JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A LEFT OUTER JOIN TABLO2 B
 ON A.SUTUN_ADI_NO=B.SUTUN_ADI
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
8	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

```
SELECT * FROM TABLO1 A LEFT OUTER JOIN TABLO2 B
ON A.DOSYA_NO=B.DOSYA_NO;
```

Tablo1 tablosunda yer alan + Tablo2 deki dosya no ile eşleşen kayıtların listesi

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
4	ayça	kılıç	40650782738				
3	hakan	kılıç	70550896722				
1	ali	şahan	12441144687				
2	deneme	denem	16915266483				

ÖRNEK

```
SELECT * FROM TABLO1 A ,TABLO2 B WHERE A.DOSYA_NO=B.DOSYA_NO(+);
```

Üsteki örnek ile birebir aynı sonucu üretir. Tek farkı yazılım biçimidir Tablo adında LEFT OUTER JOIN demek yerine Where 'den sonra a.dosya_no=b.dosya(+) "+" parametresidir

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
4	ayça	kılıç	40650782738				
3	hakan	kılıç	70550896722				
1	ali	şahan	12441144687				
2	deneme	denem	16915266483				

ÖRNEK

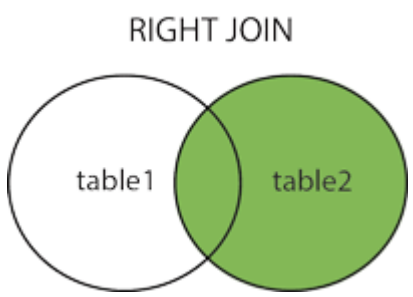
```
SELECT * FROM TABLO1 A ,TABLO2 B WHERE A.DOSYA_NO=B.DOSYA_NO
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1

Eğer ki left join yazılmaz ve (+) operatörü kullanılmaz ise her iki tabloda dosya numaraları eşit olanlar sorgulanır .Ve left join olmaktan çıkar .Tam birleşim ister

RIGHT JOIN

RIGHT Join "SAĞ DIŞ KATILMA" sorgulanan 2 tabloda SAĞDA olan tablo tamamen katılırken, solda olan tablo ise sadece eşleşenler katılır



Tablo 2 otomatik olarak dâhil olur.

Tablo 1 ise sadece tablo 2 ile eşleşen kayıtları üzerinden sorguya dâhil olur

Oluşan Küme resmi solda belirtilmiştir

RIGHT OUTER JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A RIGHT OUTER JOIN TABLO2 B
 ON A.SUTUN_ADI_NO=B.SUTUN_ADI
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
8	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

```
SELECT * FROM TABLO1 A RIGHT OUTER JOIN TABLO2 B
ON A.DOSYA_NO=B.DOSYA_NO;
```

Tablo2 tablosunda yer alan + Tablo1 deki dosya no ile eşleşen kayıtların listesi

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
				6	funda	uygur	29348069760
				7	burhan	duru	55661919654
				8	funda	uygur	16564524223
				10	miran	kılıç	75246579987
				9	beren	kılıç	98734870438

ÖRNEK

```
SELECT * FROM TABLO1 A ,TABLO2 B WHERE A.DOSYA_NO(+)=B.DOSYA_NO;
```

Üsteki örnek ile birebir aynı sonucu üretir. Tek farkı yazılım biçimidir

Tablo adında RIGHT OUTER JOIN demek yerine
Where 'den sonra a.dosya_no(+)=b.dosya "+" parametresidir

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
				6	funda	uygur	29348069760
				7	burhan	duru	55661919654
				8	funda	uygur	16564524223
				10	miran	kılıç	75246579987
				9	beren	kılıç	98734870438

ÖRNEK

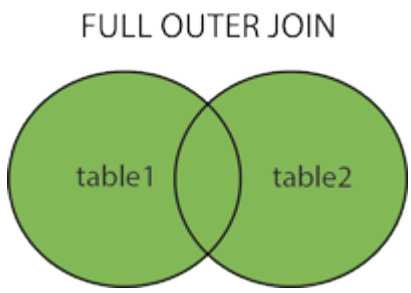
```
SELECT * FROM TABLO1 A ,TABLO2 B WHERE A.DOSYA_NO=B.DOSYA_NO
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1

Eğer ki right join yazılmaz ve (+) operatörü kullanılmaz ise her iki tabloda dosya numaraları eşit olanlar sorgulanır. Ve right join olmaktan çıkar .Tam birleşim ister

FULL JOIN

FULL Join "TAM DIŞ KATILMA" sorgulanan 2 tabloda ki tüm verileri kapsar. Aralarında eşleşen satırları tek bir satırda getirirken, sonuca eşleşmeyen satırları da dâhil eder



Tablo 1 otomatik olarak dâhil olur.

Tablo 2'de otomatik olarak dâhil olur.

Eşleşen veriler tek bir satırda gelirken sonuca eşleşmeyen satırlarda dâhil olur

Oluşan Küme resmi solda belirtilmiştir

FULL OUTER JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A FULL OUTER JOIN TABLO2 B
 ON A.SUTUN_ADI_NO=B.SUTUN_ADI
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

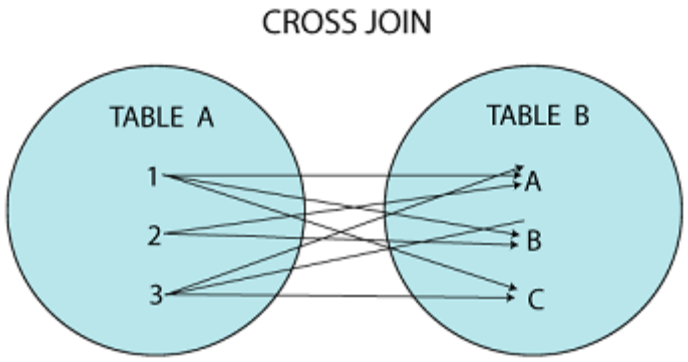
```
SELECT * FROM TABLO1 A FULL OUTER JOIN TABLO2 B
ON A.DOSYA_NO=B.DOSYA_NO;
```

Tablo1 ve Tablo2 tablosunda yer alan eşleşen veriler tek satırda + eşleşmeyen diğer satırlar

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
				6	funda	uygur	29348069760
4	ayça	kılıç	40650782738	4	funda	uygur	16564524223
				7	burhan	duru	55661919654
				9	beren	kılıç	98734870438
				10	miran	kılıç	75246579987
3	hakan	kılıç	70550896722				
1	ali	şahan	12441144687				
2	deneme	denem	16915266483				

CROSS ÇAPRAZ JOIN BİRLEŞİMLER

CROSS JOIN, çapraz birleşim belirtilen iki tabloyu tablo1 ve tablo2 sırası ile tabloda birde ki her bir satıra ,tablo2 deki tüm satırların sırası ile getirilmesine CROSS ,Çapraz birleşim adı verilir. Ortak bir sütun kullanılmaz



Tablo 1 deki her bir satır

Sırası ile tablo2 de satırlara bağlanır

Oluşan Küme resmi solda belirtilmiştir

CROSS JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A CROSS JOIN TABLO2 B
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

```
SELECT * FROM TABLO1 A cross JOIN TABLO2 B
```

Tablo1 deki her bir satır tablo 2 deki her bir satıra sırası ile çapraz olarak bağlanıyor

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
1	ali	şahan	12441144687	6	funda	uygur	29348069760
1	ali	şahan	12441144687	4	funda	uygur	16564524223
1	ali	şahan	12441144687	7	burhan	duru	55661919654
1	ali	şahan	12441144687	9	beren	kılıç	98734870438
1	ali	şahan	12441144687	10	miran	kılıç	75246579987
2	deneme	denem	16915266483	6	funda	uygur	29348069760
2	deneme	denem	16915266483	4	funda	uygur	16564524223
2	deneme	denem	16915266483	7	burhan	duru	55661919654
2	deneme	denem	16915266483	9	beren	kılıç	98734870438
2	deneme	denem	16915266483	10	miran	kılıç	75246579987

SELF JOIN

SELF JOIN tablonun kendisi ile birleşimidir. Yani tablo birde olan bir değeri yine tablo1 de yer alan bir değer ile şartlamak ve elde edilen sonucu listelenmesidir.

Tablo 1 yine kendisi ile birleştirilir.

Belirli bir kolon üzerinden ayrıştırılarak sorgu amaçlı kullanılır

Oluşan Küme resmi solda belirtilmiştir

SELF JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A CROSS JOIN TABLO1 B
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

```
SELECT * FROM TABLO1 A ,tablo1 b
where a.dosya_no<>b.dosya_no and a.soyadi=b.soyadi
```

Tablo1 kendisi ile tekrar birleşiyor. Dosyaları farklı ama soyadları aynı olan satırlar olarak

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1
4	ayça	kılıç	40650782738	3	hakan	kılıç	70550896722
3	hakan	kılıç	70550896722	4	ayça	kılıç	40650782738

NON-EQUİ JOİN

NON EQUi JOIN Eş olmayan birleşimler, Birleştirme işlemi eşitlik üzerinden değil eşitsizlik üzerinden yapılır.Tablo1 deki ad ve soyadı tablo2 deki farklı bir alan ile eşleştirdiğimizi varsayalım.Tablo2 deki alanın farklı olmasına göre ilerleyen join türüdür.

NON-EQUİ JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A  join tablo2 b
on A.sütun1 between B. sütun2 and B. sütun3
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	ID	SAYI
6	funda	uygur	29348069760	55	1
4	funda	uygur	16564524223	2	66
7	burhan	duru	55661919654	57	3
9	beren	kılıç	98734870438	58	68
10	miran	kılıç	75246579987	59	69

ÖRNEK

```
SELECT * FROM TABLO1 A  join tablo2 b
on A.DOSYA_NO between B.id and B.sayi
```

Tablo1 yer alan dosya_no alanın tablo2 deki id ve sayı ile eşleşmesi durumunda sonuç üreten sql

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOSYA_NO_1	ADI_1	SOYADI_1	TC_KIMLIK_NO_1	ID	SAYI
4	ayça	kılıç	40650782738	4	funda	uygur	16564524223	2	66
3	hakan	kılıç	70550896722	4	funda	uygur	16564524223	2	66
2	deneme	denem	16915266483	4	funda	uygur	16564524223	2	66

SUBQUERY JOİN

SUBQUERY Alt/İç Sorgu, olarak adlandırılan sorgu türü. Başka bir sorgu ifadesinin içinde görülen sorgulara alt yâda iç sorgu adı verilir. Ona sahip olan ifadeye ise dış yâda ana sorgu olarak adlandırılır

Alt Sorgular, ana sorgulardan daima önce yürütülür

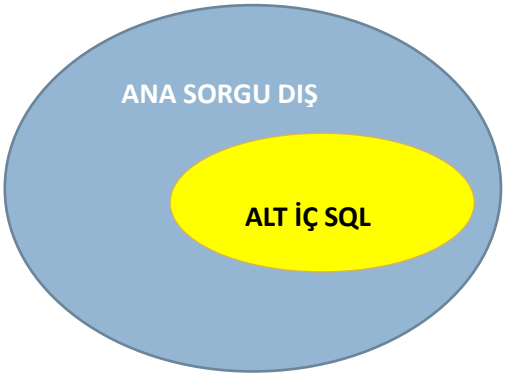
Alt sorgudan elde edilen sonuç, ana sorgu tarafından kullanılır

Alt sorgular parantez içine alınır

Alt sorgularının sonucu kullanılırken, belirtilen şartın sağ tarafında kullanılır

Tek satırlı sorgularda tek satır operatörü kullanın = , >, < , vb.

Çok satır döndüren alt sorgularda ise çok satırlı operatörler kullanılın in, not in vb.



SUBQUERY Alt sorgu kümesi

SELF JOIN SQL KALIBI

```
SELECT * FROM TABLO1 A  WHERE A.SUTUN_ADI IN (SELECT SUTUN_ADI FROM
TABLO3 )
```

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

ÖRNEK

```
SELECT * FROM TABLO1 A  WHERE A.DOSYA_NO IN (SELECT DOSYA_NO FROM TABLO3 )
```

Tablo3 de yer alan dosya numaralarının tabloda 1 de ki dosya no alanı ile eşleşen IN () içeren sql sonucu

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	dene...	denem
3	hakan	kılıç
4	ayça	kılıç

TEK SATIRLI SORGULAR

Alt sorgunuz size tek bir satır ürettiğinde kullanacağız operatörler [=, >, >=, <, <=, <>] ile sınırlıdır. Bu tarz sorgulara tek satırlı alt sorgu adı verilir. Eğer sorgunuz birden fazla sonuç üretir ise [=, >, >=, <, <=, <>] bu operatörlerde ORA-01427 hatası alırsınız

```
SELECT * FROM TABLO1 A  where a.dosya_no >  (select dosya_no from tablo3
where dosya_no=1)
```

2	dene...	denem
3	hakan	kılıç
4	ayça	kılıç

ÇOK SATIRLI SORGULAR

Alt sorgunuz size tek bir satır değil liste üretirse ORA-01427: tek satırlık alt sorgu birden fazla satır döndürüyor hatası alırsınız. Bu durumda [=, >, >=, <, <=, <>] şartlarını kullanamazsınız .Birden fazla dönen satırlar da ANY ALL yâda IN komutları kullanmanız gerekir.

ANY

(Alt sorguda üretilen tüm satırların tek tek sırası ile yukardaki şart ile karşılaştırır ve eşit olan satırları döndürür)

ANY komutunun başına Başına =,! =,>, <, <=,> = gelmelidir.

```
SELECT * FROM TABLO1 A  where a.dosya_no = ANY  (select dosya_no from
tablo3)
```

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	dene...	denem
3	hakan	kılıç
4	ayça	kılıç

ALL

ANY farklı olarak sonuçları tek bir satırda değil, dönen tüm satırların aynı anda şartı sağlamasında geriye değer döndürür

ALL komutunun başına Başına =,! =,>, <, <=,> = gelmelidir.

```
SELECT * FROM TABLO1 A  where a.dosya_no = all  (select dosya_no from
tablo3 where dosya_no=1 )
```

DOSYA_NO	ADI	SOYADI
1	ali	şahan

IN

IN alt sorguda dönen listenin belirtilen sütünün içerip içermediğine bakar IN yada NOT IN kullanıla bilir

```
SELECT * FROM TABLO1 A  where a.dosya_no in  (select dosya_no from tablo3)
```

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	dene...	denem
3	hakan	kılıç
4	ayça	kılıç

BİRLEŞİM OPERATÖRLER

Tablo birleşimlerini kullandığımız gibi yazılan SQL SELECT komutlarını da birleştire biliriz. Bu işlemi yapan komutlara birleşim operatörü adı verilir. Kullanılan operatörler UNION ve UNION ALL **INTERSECT** ve **MINUS**

Birleşim operatörlerin JOIN 'lerden farkı nedir.

JOIN'ler YATAY birleşim sağlarken Birleşim Operatörleri DİKEY birleşim sağlar

UNION

Birden fazla SELECT komutunu birleştirerek yenilen kayıtları teke düşürülmesi ile oluşan birleşim operatörüdür.

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

```
SELECT * FROM TABLO1 A
union
SELECT * FROM TABLO3 B
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	ali	şahan	12441144687
2	deneme	denem	16915266483
3	hakan	kılıç	70550896722
4	ayça	kılıç	40650782738

UNION ALL

Birden fazla SELECT komutunu birleştirerek yenilen kayıtları teke düşürmeden, TÜM SATIRLARI ile sonuç üretene birleşim operatörüdür.

TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

TABLO 2

DOSYA_NO	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

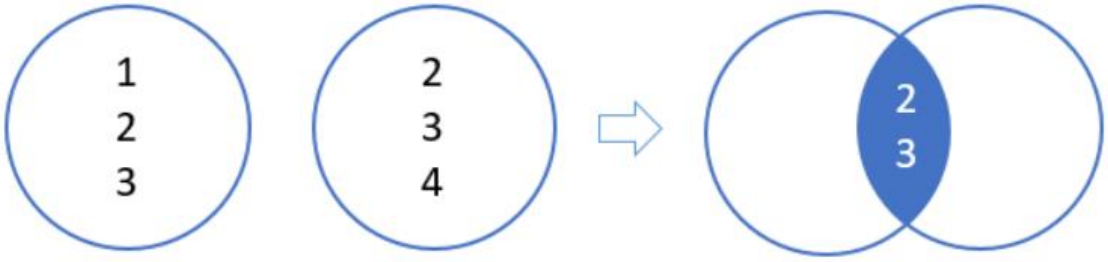
DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

```
SELECT * FROM TABLO1 A
UNION ALL
SELECT * FROM TABLO3 B
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	ali	şahan	12441144687
2	deneme	denem	16915266483
3	hakan	kılıç	70550896722
4	ayça	kılıç	40650782738
1	ali	şahan	12441144687
2	deneme	denem	16915266483
3	hakan	kılıç	70550896722
4	ayça	kılıç	40650782738

INTERSECT

KEŞİSİM, birleştirilmek istenen SQL'lerin ürettiği sonuçları kontrol eder. Tüm satırların eşit, yani SQL'lerin kesiştiği sonuçları listeler



TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	funda	uygur

TABLO 2

DOSYA...	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

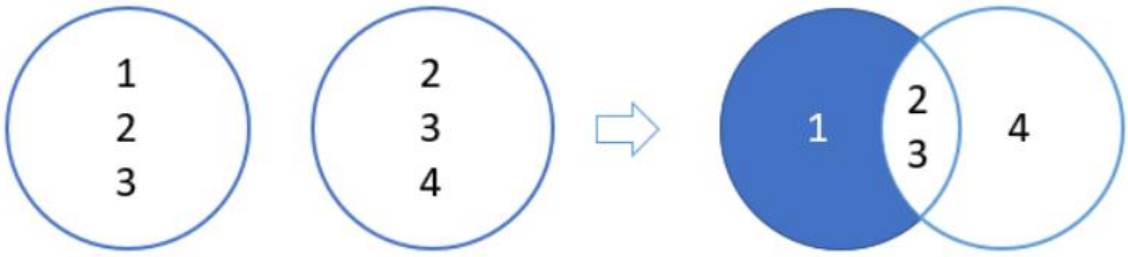
DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

```
SELECT * FROM TABLO1 A
INTERSECT
SELECT * FROM TABLO2 B
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
4	funda	uygur	16564524223

MINUS

MINUS, birleştirilmek istenen SQL'lerin ürettiği sonuçları kontrol eder. 1.SQL belirtilen sonuçlar 2.SQL de var ise bunları çıkartır. Eşleşmeyen ama 2.SQLde yer alanları da çıkartarak sonuçları listeler



TABLO 1

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	funda	uygur

TABLO 2

DOSYA...	ADI	SOYADI
6	funda	uygur
4	funda	uygur
7	burhan	duru
9	beren	kılıç
10	miran	kılıç

TABLO 3

DOSYA_NO	ADI	SOYADI
1	ali	şahan
2	deneme	denem
3	hakan	kılıç
4	ayça	kılıç

```
SELECT * FROM TABLO1 A
minus
SELECT * FROM TABLO2 B
```

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO
1	ali	şahan	12441144687
2	deneme	denem	16915266483
3	hakan	kılıç	70550896722

VERİ TANIMLAMA DİLİ (DDL) / DATA DEFİNİTİON LANGUAGE (DDL)

Veri tabanın da Tablolar üzerinde oluşturmak, silmek ve değiştirmek üzerine yapılan ifadelerdir. Temel olarak [CREATE ALTER DROP RENAME TRUNCATE COMMENT] üzerine inşa edilmiştir.

CREATE

CREATE, oluşturmak komutu. Oracle üzerinde TABLE, INDEX VIEW SEQUNCE vb türde yapılar oluşturmak için kullanılan komut. CREATE komutu kullanıldığı türe göre farklılıklar göstere bilir. Kısaca bir tablo oluşturmak ile index arasında aynı komut olmasına karşılık farklılıklar mevcuttur.

TABLE

Tablo nedir? Tablo verilerin koleksiyon türünde saklanması. Sütun olarak başlıklardan satır olarak datalardan oluşan veri kümeleridir.

Oracle ’da tablo oluşturmak için CREATE TABLE komutunu kullanılır. İlk olarak oluşturmak istediğiniz tablo adını belirtin. Parantez içinde tabloya ait olan sütunları ekleyin. Sütunlar birden çok ise arasında virgül kullanarak birbirinden ayırmanız gerekir. Veri tipleri olarak number,varchar2 vb. alanları yâda kısıtlama olarak NOT NUL, PRİMARY KEY vb. kullana bilirsiniz

Genel olarak yazım biçimi aşağıda belirtildiği gibidir. Kullandığınız yapıya göre farklılıklar göstere bilir

VERI TIPI	AÇIKLAMA
VARCHAR2(size)	Karakter türünde belirtilen size kadar alan oluşturur
CHAR(size)	Sabit uzunlukta karakter verileri
NUMBER(p,s)	Sayılar için kullanılan değişken alanı
DATE	Tarih ve Saat değişken alanı
LONG	2 GB kadar karakter değişken alanı
CLOB	4 GB kadar karakter değişken alanı
RAW and LONG RAW	Ham ikili veri
BLOB	Ham ikili veri alan 4 GB kadar
BFILE	Harici bir dosyada saklanan ikili veriler (4 GB'a kadar)
ROWID	Benzersizliği temsil eden bir taban-64 sayı sistemi
TIMESTAMP	Kesirli saniye ile tarih değişken alanı
INTERVAL YEAR TO MONTH	Yıl ve AY aralığı olarak saklanan değişken alanı
INTERVAL DAY TO SECOND	Gün, saat, dakika aralığı olarak saklanan değişken alanı

ÖRNEK

```
CREATE TABLE KIMLIK
(
ID NUMBER,
ADI VARCHAR2 (50),
SOYADI VARCHAR2 (50),
SEHIR VARCHAR2 (50),
TARIH DATE DEFAULT SYSDATE
);
```

Oluşturulan tablonun DESC komutu karşılığı

DESC KIMLIK

```
TABLE KIMLIK
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
-----
 ID                                                 NUMBER
 ADI                                                VARCHAR2(50)
 SOYADI                                             VARCHAR2(50)
 SEHIR                                              VARCHAR2(50)

 TARIH                                              DATE
```

VIEW

View nedir? View Dataları Fiziksel olarak tutmaya yarayan SQL ile oluşturulmuş SANAL veri kümelerine verilen addır. Tablo fiziksel olarak yer kaplarken View'ler tamamen sanaldır. SQL yazılarak içinde ki kayıtlar oluşturulur. Genellikle karmaşık birden fazla tablonun bağlandığı sorguları tekrar yazmak yerine hızlıca erişim sağlamak için kullanılır

Genel yapısı `CREATE VIEW VI_KIMLIK`, yâda `CREATE OR REPLACE VIEW VI_KIMLIK`

Alt sorgularda takma adlar kullanarak View için sütun adlarını oluştura bilirsiniz

ÖRNEK

```
CREATE VIEW VI_KIMLIK                    Oluşturulan View Adı
AS

SELECT * FROM KIMLIK WHERE               View için yazılan alt sorgu
DOSYA_NO<6
```

View çağrılması

```
SELECT * FROM VI_KIMLIK
```

WITH CHECK OPTION

View olarak oluşturulan dataya insert yada update yapılırken SQL içersinde belirtilmiş şarta bağlı kalınması istenirse bu komutu kullana bilirsiniz

ÖRNEK

```
CREATE OR REPLACE FORCE VIEW HASTANE.VI_KIMLIK
(
   DOSYA_NO,
   ADI,
   SOYADI,
   TC_KIMLIK_NO,
   DOGUM_TAR,
   MEMLEKET

)
```

```
AS
   SELECT "DOSYA_NO",
          "ADI",
          "SOYADI",
          "TC_KIMLIK_NO",
          "DOGUM_TAR",
          "MEMLEKET"

     FROM kimlik

    WHERE dosya_no < 6 WITH CHECK OPTION;
```

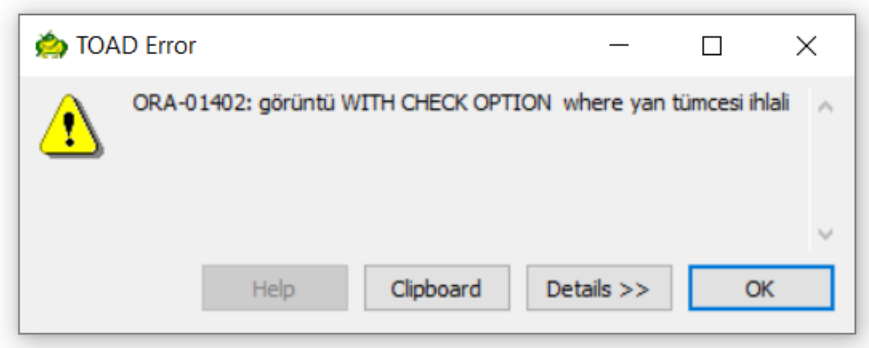
Elde edilen sonuç tablosu

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
1	ali	ff	12441144687	1/04/1991	UŞAK
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN
3	hakan	kılıç	70550896722		ÇORUM
4	ayça	kılıç	40650782738	12/06/1992	EDİRNE

WITH CHECK OPTION komutu dosya_no<6 şartıyla ekleme ve değişikliklere izin verecektir.

İhlal edildiği durumda ise aşağıda ki hata ile ilerleyişi durduracaktır

DOSYA_NO	ADI	SOYADI	TC_KIMLIK_NO	DOGUM_TAR	MEMLEKET
1	ali	ff	12441144687	1/04/1991	UŞAK
2	deneme	denem	16915266483	21/11/2021	ERZİNCAN
3	hakan	kılıç	70550896722		ÇORUM
7	ayça	kılıç	40650782738	12/06/1992	EDİRNE



WITH READ ONLY

Bu komut ile oluşturulan View tablosunda güncelleme yapılamaz. Kullanıcı sadece veriyi okuya bilir. Değişlik yapmak istediğinde ise aşağıdaki hata kodunu alacaktır

```
ORA-42399: SALT-OKUNUR GÖRÜNÜMDE DML IŞLEMI GERÇEKLEŞTIRILEMEZ
```

SEQUNCE

SEQUNCE sıralı diziler anlamına gelir. Otomatik olarak benzersiz numaralar üretir. Üretilen numaralar select olarak çağrıla bildiği gibi insert komutu içinde kullanıla bilir Peki bu numaralar nerede kullanılır. Oluşturduğunuz bir tabloda (benzersiz/unique) alana ihtiyaç duya bilirsiniz. Bu değeri kullanıcıya girdirmek ve takip etmek imkânsız bir hal alır. Bu durumda aradığınız yapı SEQUNCE

KOMUTLAR	AÇIKLAMASI
CREATE SEQUENCE SEQ_KIMLIK	SEQ Adı
START WITH 1	Başlangıç Değeri
INCREMENT BY 1	Artış Değeri
MAXVALUE 99	Bitiş Değeri
MINVALUE 1	Minimum Değeri
NOCYCLE	Seq Max Değere ulaştığında başa dön yada dönme olarak çalışır
NOCACHE	Cache özelliği devre dışı bırakır. Varsayılan 20 alır
NOORDER	Cluster yapılar için geçerli zorlama görevi

ÖRNEK

```
CREATE SEQUENCE HASTANE.SEQ_KIMLIK
  START WITH 2
  INCREMENT BY 2
  MAXVALUE 99
  MINVALUE 1
  NOCYCLE
  NOCACHE
  NOORDER;
```

SEQ yeni bir numara üretilmesi

```
SELECT SEQ_KIMLIK.NEXTVAL FROM DUAL
```

Yeni bir Numara üreterek değer döndürür

```
    NEXTVAL
----------
         2
1 row selected.
```

= SEQ ŞUAN Kİ DEĞERİ, DEĞER ÜRETMEDEN

```
SELECT SEQ_KIMLIK.CURRVAL FROM DUAL
```

Yeni bir Numara üretilmeden en son alınan değeri döndürür

```
    NEXTVAL
----------
         2
1 row selected.
```

= SEQ INSERT İCİNDE KULLANMA

```
INSERT INTO KIMLIK
(DOSYA_NO, ADI, SOYADI)
VALUES (SEQ_KIMLIK.NEXTVAL,'ALI','ŞAHAN')
```

İNDEX

İndex tablolarda ki verilere daha hızlı erişmek için oluşturulmuş yapılardır. Data boyutu büyük tablolarda kullanılarak hız ve performans kalitesini oldukça etkiliye bilir. Doğru tasarlanmış bir index size ++ özellikler sunacaktır.

İndex i daha iyi anlamak için kısa bir örnek;

Elinizde 50.000 adet kimlik bilgisi olan bir tablomuz mevcut. Alanlarımız [ADİ, SOYADİ, TC VE ŞEHİR] içermekte. Siz bu tablodaki şehir bilgisinde YOZGAT yazan kayıtları bulmanız için 50.000 kişi tek tek sorgulamanız gerekir.

Burada hız ve zaman kazanmak için ilgili tabloya şehir bazında bir index oluşturursanız 50.000 kayıt şehirlere göre kümelere çevrilir. Ve tablonun bir yansıması oluşturulur

Aynı sorgu çalıştığında ise 50.000 satırı tek tek gezmek yerine kümelerden YOZGAT bulunur ve kayıtlar listelenir. İndex tam olarak bunu yapar büyük tablolardaki verileriniz belirli sütun ve kurallara göre sıralamasını yaparak kayıt altına alır.

Eğer ki tablodaki sütun alanız çok farklı değerlerden oluşuyorsa indexi tercih edebilirsiniz.

SQL cümleniz WHERE cümlesinden sonra çok fazla sütün ile ilişkilendirilmiş ise

SQL'de sorgulanan veri tablonun %15 eşit ve daha az ise, ve tabloda index mevcut ise ben tabloyu taramak yerine indexi kullanır hızlıca sonuca erişim mantığı ile ilerler

SQL'de ile istenen kayıt sayısı genel tablonun %15 den daha fazla ise ,Oracle index olup olmadığına bakmaksınız Full Table Scan yaparak tablonun tamamını tarar ve sonuç üretir

İndex yada tablo üzerinden gidilip gidilmeyeceğine Query Optimizer karar verir.

Tablodaki dizin anahtarı,erişim hızı ve verilerin dağılımı yüzde farklılık göstere bilir

AVANTAJLARI

- Kayıtlara erişimde hız kazandırır
- Performans kullanımı azaltarak sistemi yormaz
- Unique kullanarak yenilenen satırları engelleye bilirsiniz.
- SQL cümleniz, sadece indexli alanları sorguluyorsa, tablo hiç kontrol edilmeden sonuç index üzerinden sağlanabilir.
- İndexler tetiklenmez ve SQL olarak çağrılmaz. Oracle tarafından otomatik kullanılır ve bakımı yapılır

DEZAVANTAJLARI

- Alan kapladığı için gereksiz kullanımda disk maliyeti oluştura bilir
- İlgili tabloda çok fazla insert update ve delete yapıyorsa index sizi yavaşlata bilir
- Tabloda yapılan DML işlemlerinde (INSERT UPDATE DELETE) indexler yenilenir. Ne kadar çok hareket görürse aynı sayıda index yenilenecektir.

ÖRNEK

```
CREATE INDEX IN_KIMLIK
ON KIMLIK
(MEMLEKET, ADI)
```

IN_KIMLIK adında kimlik tablosu verileri için memleket ve adi sütunlarına göre index oluşturulması

```
DROP INDEX IN_KIMLIK_SOYADI;
```

Oluşturulan indexin silinmesi için DROP index komutu

```
CREATE UNIQUE  INDEX IN_KIMLIK
ON KIMLIK
(MEMLEKET, ADI)
```

Komutu ise ilgili sütunların tekrar edilmesi engeller satırları benzersiz bir küme oluşturur

11

SYNONYM

Oracle objelerine alternatif olarak verilen takma isim (alias).Bu objeler tablo index sequnce procedure index vb. objeleri içere bilir.

Elinizde çok uzun yada karmaşık gelen bir obje ismi varsa bu komut ile takma isimler oluştura bilirsiniz

ÖRNEK

```
CREATE  SYNONYM  F1  FOR KIMLIK;
```

KIMLIK tablosuna kısa bir takma isim olması için F1 adını atamış olduk

SELECT * FROM KIMLIK	Her iki komutta aynı verileri sonuç olarak listelemekte
SELECT * FROM F1	

```
DROP  SYNONYM  F1
```

Oluşturulan SYNONYM silinmesi için DROP komutu

CONSRAINTS

CONSTRAİNTS kısıtlamalar. Tablolar üzerinde veri işleme yapılırken insert update delete kontroller ekleye bilirsiniz. Kısıtlamalar boş olmasın, benzersiz olsun vb. [NOT NUL, UNIQUE, PRIMARY KEY, CHECK CONSRAINTS] türlerinde eklene bilir

NOT NUL

Sütun için boş değer girişini engeller. Tablo oluşturulurken yada sonrası içinde aşağıdaki komut ile eklene bilir

```
ALTER TABLE  KIMLIK MODIFY DOSYA_NO NOT NULL
ENABLE VALIDATE;
```

UNIQUE

Tabloya kayıt girilirken belirli kolonların unique (tekil) olması sağlanır. Bir tabloda birden fazla unique alan tanımlanabilir

```
ALTER TABLE KIMLIK ADD (
CONSTRAINT CS_DOSYA   UNIQUE (DOSYA_NO));
```

```
ENABLE VALIDATE);
```

PRIMARY KEY

Yapı olarak **UNIQUE** ile aynı mantık ile çalışır. Ayırt edici özelliği ise tabloda bir tane primary key tanımlaya bilirsiniz

```
ALTER TABLE KIMLIK ADD (
CONSTRAINT CS_DOSYA   PRIMARY KEY ( DOSYA_NO));
```

FOREIGN KEY

Bir kolonun başka bir tablodaki primary key ile ilişkilendirmek için kullanılır. Yani 2 farklı tabloyu seçilen kolonlar üzerinden diğeri primary key olmak zorunda kısıtlamak için kullanılır

```
ALTER TABLE KIMLIK ADD
CONSTRAINT FK_DOSYA_NO
 FOREIGN KEY (DOSYA_NO)
 REFERENCES HASTA (DOSYA_NO)
 ENABLE
 VALIDATE
```

CHECK

Tablodaki sütunun belirlenen değerlerde kayıt yapmasını sağlar.

```
ALTER TABLE KIMLIK ADD (
 CONSTRAINT CH_MAAS
 CHECK  (maas BETWEEN 1 and 99)
       ENABLE VALIDATE);
```

DATA TYPES

Bir veri kaynağından okunurken Oracle tarafından gelen veri, desteklenen veri tiplerine eşleştirmeyi dener. Kısaca date olarak tanımlanmış bir alan için sadece tarih değerlerini gire bilirsiniz. Grup düzeyinde bakıldığında ise, aşağıdaki liste göz önüne alına bilir

SAYI TÜRLERİ - SMALLINT, SMALLUNIT, TINYINT, TINYUINT, UINT, BIT, FLOAT, INT, NUMERIC, DOUBLE

TARİH TÜRLERİ - DATE, DATETIME, TIMESTAMP, TIME

DİZE TÜRLERİ - LONGVARCHAR, CHAR, VARCHAR

VARCHAR2

Değişken uzunlukta karakter dizilerini saklar. Uzunluk 1 BYTE ile 4000 BYTE arasında olabilir. Buda VARCHAR2 için karakter uzunluğu en fazla 4000 karakter anlamına gelir. Belirtiğiniz uzunluktan daha uzun bir karakter saklamak istediğinizde Oracle size bir hata verir

VARCHAR2(max_size BYTE)

```
CREATE TABLE TESTTABLO (
    adi CHAR(10),
    soyadi VARCHAR2(10)
);
```

CHAR

SABIT uzunlukta karakter dizilerini saklar. Uzunluk 1 BYTE ile 2000 BYTE arasında olabilir. Siz tanımladığınız uzunluktan eksik veri girerseniz Oracle eksik alanları boşluk doldurarak tanımlanan uzunluğa eşitler

CHAR(max_size BYTE)

```
CREATE TABLE TESTTABLO (
    adi CHAR(10),
    soyadi VARCHAR2(10)
);
```

NUMBER

Pozitif yâda negatif olabilen sayı değerlerini depolamak için kullanılır. Oracle NUMBER türünün kesinliği ve ölçek tanımı vardır.

Kesinlik bilgisi olarak basamak sayısı 1 ile 38 arasında değişe bilir

Ölçek bilgisi ise, bir sayının ondalık sağında kalan basamak sayısıdır -84 ile 127 arasında değişe bilir

Örneğin 1234,56 sayınının tanımı

Number (6,2) olarak hesaplanır. Eğer ki tanımdan daha fazla girilir ise,

“Değer, bu sütun için belirlenmiş izin verilen duyarlılıktan daha fazla” hatası verir

Tablo oluştururken SQL

```
CREATE TABLE KIMLIK (
    maas NUMBER
);
```

Tablo için sonradan alan ekleme SQL

```
ALTER TABLE KIMLIK    ADD (maas NUMBER);
```

DATE

Date veri türü, bir sn. hassasiyet ile hem tarih hem de saat bilgisini kapsar. Tarih bilgilerini saklamak için kendi özel formatı vardır yıl ay gün saat dakika ve saniye karşılık gelen 7 byte boyutunda sabit alanları kullanır. Tablaya eklemek için ;

```
ALTER TABLE KIMLIK     ADD (tarih date);
```

SQL olarak, tarih bilgisini öğrenmek için

```
SELECT SYSDATE FROM DUAL;
```

LONG

LONG veri türü 2 GB kadar veri içeren değişken uzunluktaki karakter verilerini depolaya bilir. LONG değerinin uzunluğu bilgisayarınızın belleği ile sınırlı olabilir

Tablo başına 1 adet kullanım ile sınırlıdır

SELECT update INSERT olarak kullanılır

WHERE GROUP BY ORDER BY DISTINCT operatörleri ile kullanılamaz

SELECT alt sorgularda, birleştirilmiş SQL cümlelerinde UNION, UNION ALL, INTERSECT veya MINUS tarafından kullanılamaz

```
ALTER TABLE KIMLIK     ADD (METIN LONG);
```

CLOB

CLOB büyük karakter nesneleri anlamına gelir. Maksimum 4 GB boyutunda olup, CLOB hem sabit karakterli hem de değişkenli karakter kümelerini desteklediğini unutmayın

```
ALTER TABLE KIMLIK     ADD (VIDEO CLOB);
```

RAW AND LONG

RAW and LONG veri türleri belge, ses, video dosyaların içeriği gibi ikili verileri veya bayt dizilerini depolamak için kullanılır

Alan boyutu 2GB kadar veri boyutu içere bilir

Tablo başına 1 adet kullanım ile sınırlıdır

SELECT update INSERT olarak kullanılır

WHERE GROUP BY ORDER BY DISTINCT operatörleri ile kullanılamaz

SELECT alt sorgularda, birleştirilmiş SQL cümlelerinde UNION, UNION ALL, INTERSECT veya MINUS tarafından kullanılamaz

```
ALTER TABLE KIMLIK     ADD (VIDEO LONG RAW);
```

BLOB

BLOB ikili büyük nesneler (tek bir varlık olarak) depolan nesneler (SES, Video yada multimedya) anlamına gelir. Maksimum boyutu 4 GB olarak belirlenmiştir.

```
ALTER TABLE KIMLIK     ADD (DOSYA BLOB);
```

BFILE

BFILE veri türü, database içerisinde değil database dışında dizinlerde klasörlerde saklanan sadece adresleri bu alanda tutulan veri tipleridir.

```
ALTER TABLE KIMLIK    ADD (ADRES BFILE);
```

ROWID

ROWID veri tabanında ilgili satırın adresini tutar. Değer atandıktan sonra değeri üzerinde değişiklik yapmak doğru değildir. Hızlı erişimler için ideal bir veri türüdür.

```
ALTER TABLE KIMLIK    ADD (ID ROWID);
```

ALTER

ALTER "DDL" yani "Veri tanımlama dili" ifadelerinde biridir. ALTER komutu, değiştirmek için kullanılır. Kullana bileceğiniz alanlar

ALTER komutu ile şemaları "SCHEMA " ,rolleri " ROLES ", tabloları "TABLE" ,alter SESION, alter SYSTEM alter USER vb. amaçlar için kullana bilirsiniz

ÖRNEK

```
ALTER ANY TRIGGER
ALTER ANY TYPE
ALTER DATABASE
ALTER LOCKDOWN PROFILE
ALTER PROFILE
ALTER RESOURCE COST
ALTER ROLLBACK SEGMENT
ALTER SESSION
ALTER SYSTEM
ALTER TABLESPACE
ALTER USER
```

DROP

DROP, yetkisi dâhilinde belirtilen nesneleri bırakmak için kullanılır. Kullanım formatı
DROP TYPE [schema.]type_name [**FORCE** | **VALIDATE**] ;
TYPE/ türe göre farklılık göstere bilir

```
DROP TABLE KIMLIK;
DROP FUNCTION FN_KIMLIK
```
vb. Formata kullana bilirsiniz

RENAME

RENAME, yeniden adlandırmak için kullanılan bir DDL komutudur. Bir tabloyu , index sequence yeniden adlandırmak için kullana bilirsiniz. Kullanım formatı

```
RENAME OLD_NAME     TO NEW_NAME ;

RENAME SEQ_DOSYA TO SEQ_DOSYA_YENI
RENAME TABLO_ESKI TO TABLO_YENI
```

TRUNCATE

TRUNCATE, bir tabloya ait tüm satıları hızlı verimli bir şekilde silmek için kullanılır. Örneğin bir tablodan tüm verileri silmek istediğinizde DELETE komutu iyi bir iş çıkarsa da tüm satırlar söz konusu olduğunda verimli değildir. Bir tabloyu tüm satıları ile silmek için TRUNCATE komutu kullanılır

```
TRUNCATE TABLE  TABLO_ADI
```

VERİ TANIMLAMA DİLİ (DDL) / DATA DEFİNİTİON LANGUAGE (DDL)

DCL, Veri Kontrol Dili veri tabanı ile ilişkili kullanıcıları ve rollerin izinlerini değiştirmek yani verilere erişim yetkilerini düzenlemek amacıyla kullanılır **GRANT** vermek REVOKE almak üzerine kullanılan komutlardır

Bu yetkiler ROLLER , SYSTEM yetkileri, OBJE yetkilerini içere bilir

SYSTEM YETKİLERİ	NESN YETKİLERİ
CREATE SESSION	INSERT
CREATE TABLE	UPDATE
CREATE VIEW	DELETE
CREATE PROCEDURE	INDEX
SYSDBA	EXECUTE
SYSOPER	

GRANT

GRANT ifadesi belirli bir kullanıcıya bir veya daha fazla ayrıcalıklar atar. GRANT komutunun genel yapısı

```
GRANT {SYSTEM_PRİVİLEGES | OBJECT_PRİVİLEGES } TO USER  [WITH ADMIN OPTION]
```

GRANT anahtar kelimesi

Sistem yada nesne ayrıcalıklarından kullanılacak özellik

Hangi kullanıcı ise, kullanıcı adı

Öncelik TEST adında bir kullanıcı oluşturuyoruz

```
CREATE USER TEST
  IDENTIFIED BY Pas123
  DEFAULT TABLESPACE USERS
  TEMPORARY TABLESPACE TEMP
  PROFILE DEFAULT
  ACCOUNT UNLOCK;
```

Kullanıcı login olmak istediğinde yetkileri verilmediği için başta session hata alırsınız

ORA-01045: TEST kullanıcısının CREATE SESSION ayrıcalığı yok; oturum açma reddedildi

```
GRANT CREATE SESSION TO TEST /*kullanıcıya sesion oluşturma yetkisi*/
```

Kullanıcıya yetki vermek için ise yukarıdaki komutu kullana bilirsiniz

Artık kullanıcı login ola bilir.

Bir kullanıcıda hangi yetkiler mevcut kontrol etmek için ise aşağıdaki komutu kullana bilirsiniz

```
SELECT * FROM SESSION_PRIVS;
```

REVOKE

REVOKE ifadesi bir kullanıcının sistem ve nesne ayrıcalıklarını iptal eder. GRANT ile tanımladığınız yetkileri REVOKE ile geri ala bilirsiniz

```
REVOKE {SYSTEM_PRİVİLEGE | OBJECT_PRİVİLEGE } FROM USER;
```

İlk olarak iptal etmek istediğiniz sistem yada nesne ayrıcalıklarını belirtin

İkinci olarak ise iptal etmek istediğiniz kullanıcıyı belirtin

Son olarak bu yetkileri almak için komutu çalıştıran kullanıcının sistem ayrıcalıklarına sahip bir kullanıcı olması gerekir

GRANT ile yetki verirken

```
GRANT SELECT, INSERT, UPDATE, DELETE ON KIMLIK TO TEST
```

REVOKE ile yetki almak için

```
REVOKE SELECT, INSERT, UPDATE, DELETE ON KIMLIK from TEST
```

İŞLEM KONTROLÜ / TRANSACTİON CONTROL TCL

Transaction birden çok SQL sorgularının çalıştırıldığı ama Oracle bu soncuları bir ünite gibi algıladığı mantıksal bir sorgu bloğudur. Bu blok COMMIT ve ROLLBACK ile parçana bilir veya sonlandırıla bilir. Sonuç olarak işlem yönetimi olarak düşüne bilirsiniz. Veri İşleme Dili DML (insert update delete) ile yapılanları yönetmemize yarayan kontrol ifadeleridir. **COMMIT**, **SAVEPOINT**, **ROLLBACK** ifadelerinden oluşur

Veri Kontrol Dili DCL ve Veri Tanımalam Dili DDL bölümleri (alter drop create vb.) için geçerli değildir.

COMMIT

COMMIT Transaction olarak işlenen veririnin sonlandığını, yapılan değişikliklerinin kalıcı olarak kayıt eden komuttur. COMMIT ifadesi kullanılmadığı durumda veri sadece sizde gözükür. Veri tabanına kayıt etmek için COMMIT ifadesi ile sonlandırılmalıdır

```
UPDATE KIMLIK SET ADI='ŞAHAN' WHERE ADI='ALİ';

/*yukarıdaki SQL bilgisinin veri tabanına kayıt olması için sonraki komut
Olarak commit ile Transaction kapatılmalıdır*/

COMMIT;
```

SAVEPOINT

SAVEPOINT kaydetme noktası. Transaction ile işlediğiniz bir sorgu bloğunu kaydetme noktası olarak saklaya bilirsiniz. Peş peşe aynı isimler ile savepoint noktası oluşturursanız en son belirlenen ismi kullanacaktır

Bir kayıt noktası oluşturulduktan sonra tüm işlemleri geri ala bilir yada kayıt noktasına döne bilirsiniz

```
UPDATE MAAS
    SET ASGARI = 7000
    WHERE GOREVI = 'IT';
SAVEPOINT IT_KAYIT_NOKTASI;

ROLLBACK TO SAVEPOINT IT_KAYIT_NOKTASI;
```

ROLLBACK

ROLLACK yapılan Transaction sorgu bloğunu geri almak için kullanılır.

```
UPDATE KIMLIK SET ADI='ŞAHAN' WHERE ADI='ALİ';

/*yukarıdaki SQL bilgisinin veri tabanına kayıt olması için sonraki komut
```

```
Olarak commit ile Transaction kapatılmalıdır*/

ROLLBACK;
```

Ayrıca bir SAVEPOINT kayıt noktasına geri almak için SQL kullana bilirsiniz

ROLLBACK TO SAVEPOINT KAYİT_NOKTASİADİ;