

İçindekiler

Uygulamalarla SQL Öğreniyorum	4
Giriş	4
SQL Ne Demektir?	4
İlişkisel Veritabanı Sistemleri (RDMS).....	4
Temel SQL Komutları	5
Data Manipülasyon Komutları.....	5
Database Manipülasyon Komutları	5
Select Komutu	5
Insert Komutu.....	6
Update Komutu	6
Delete Komutu	7
Where Şartı Kullanımı.....	8
Distinct Komutu.....	14
Order By Komutu.....	15
Top Komutu	17
Örnek Satış Datası	18
Aggregate Fonksiyonlar	18
Group By Kullanımı	19
SQL Server Veri Tipleri	26
İlişkisel Veritabanı Sistemleri (RDMS)	29
SQL Server Tablo Oluşturma.....	29
Veri Oluşturma	33
Join İşlemleri	34
INNER JOIN	34
LEFT JOIN	34
RIGHT JOIN	35
FULL JOIN.....	35
ALIAS KULLANIMI.....	35
GROUP BY KULLANIMI.....	36
E-Ticaret Datası Sorgulama	37
Örnek-1.....	37
Örnek-2.....	38
Örnek-3.....	39
Örnek-4.....	39

Örnek-5.....	40
Örnek-6.....	40
Sub Query	41
SubQuery Giriş.....	41
Örnek.....	42
Örnek: Müşterinin Sepete Eklediği Son Ürün.....	43
String İşlemleri	44
ASCII VE CHAR.....	44
SUBSTRING	44
CHAR INDEX.....	45
CONCAT, CONCAT_WS	45
FORMAT.....	46
LEFT, RIGHT, LEN	46
TRIM, LTRIM, RTRIM.....	47
LOWER, UPPER, REVERSE, REPLICATE	48
REPLACE.....	49
Uçtan Uca SQL Server.....	50
Sistem Database'leri	50
Master DB.....	51
Model DB.....	51
MSDB.....	52
Temp DB	52
T-SQL Kodları.....	53
Data Definition Language (DDL) Kodları.....	53
Değişken Kullanımı	53
Date Time fonksiyonları.....	54
T-SQL'de Döngüler.....	56
Index.....	58
Büyütmen Veride Performans Hesaplama.....	59
Index Uygulama.....	60
INCLUDED COLUMN	61
Birden Fazla Index Oluşturma	63
Unique Index	65
Index Bozulmaları (Fragmantation).....	67
Index Fragmantation Uygulaması (Fill Factor).....	70

İstatistikler	72
View.....	75
User Defined Functions.....	76
Gün Adını Getiren Fonksiyon.....	78
İki Parametreli Fonksiyonlar.....	80
Table Valued Functions	81
Stored Procedure.....	83
Turnike Geçiş Sistemi Örneği.....	83
Stored Procedure Oluşturma	86
Stored Procedure Kullanmanın Faydaları.....	88
Trigger.....	91
Toplam Tablosu Oluşturma	92
Toplam Tablosu Insert Trigger.....	93
Toplam Tablosu Delete Trigger	94
Toplam Tablosu Update Trigger	94
Trigger ile Son Kaydı Loglama.....	95
Trigger ile Loglama	97
Instead of Trigger	100
Transaction Kavramı ve OLTP Sistemler.....	101
Database Oluşturma.....	103
Banka Transaction Uygulaması.....	104
Transaction ile Tabloyu Kilitleme (NOLOCK)	107
Backup / Restore İşlemleri	108
Backup Alma Yöntemleri	108
Backup Planlama Stratejisi	109
SQL Server Agent.....	112
SQL Server Agent Kavramı	112
Kaynakça.....	113

Uygulamalarla SQL Öğreniyorum

Giriş

SQL Ne Demektir?

SQL = Structured Query Language (Yapısal Sorgulama Dili)

Veritabanlarındaki verileri yönettiğimiz bir çeşit kodlama dilidir.

İlişkisel Veritabanı Sistemleri (RDMS)

Tekrar eden verileri tekilleştirmek amacıyla yapılandırılan veritabanı sistemleridir.

A	B	C	D
LOGICALREF	CODE	DEFINITION_	CITY
1	1	Serhat GÜndoĞan	Rize
2	2	Irmak TAHSİNOĞLU	Erzincan
3	3	Yaşar SAVURGAN	Isparta
4	4	Remzi ELYİĞİT	Giresun
5	5	Mehmet Akif POLAST	Bitlis
6	6	Sıh FAYDALI	Ordu
7	7	Münevver AYAOKU	Artvin
8	8	Muhammet KUZUCUOĞLU	Aydın
9	9	Nazlican ÖZSİMİTÇİ	Şanlıurfa
10	10	Arya UNLUMAMULERİ	Samsun
11	11	Döne GURBETOĞLU	Kütahya
12	12	Emirhan SELİM	Samsun
13	13	Soner ÜLGEN	Elazığ
14	14	Cetin BÖRKÜLÜ	Eskişehir
15	15	Ezgi İBUKÜRTÜNCÜ	Çanakkale
16	16	Hazal ÜREGİL	Tunceli
17	17	Aykut SUYUR	Rize
18	18	Oya KİLT	

Temel SQL Komutları

Data Manipülasyon Komutları

- **SELECT** : Veritabanından kayıtları çeker.
- **UPDATE** : Bir tablodaki kaydın bir ya da daha fazla alanını günceller, değiştirir.
- **DELETE** : Bir tablodan kayıt siler.
- **INSERT INTO** : Tabloya yeni kayıt ekler.
- **TRUNCATE TABLE** : Tablonun içini boşaltır.

Database Manipülasyon Komutları

- **CREATE DATABASE** : Yeni veritabanı oluşturur.
- **ALTER DATABASE** : Bir veritabanının özelliklerini değiştirir.
- **CREATE TABLE** : Yeni bir tablo oluşturur.
- **ALTER TABLE** : Bir tablonun özelliklerini değiştirir.
- **DROP TABLE** : Bir tabloyu tamamen siler.
- **CREATE INDEX** : Index oluşturur.
- **DROP INDEX** : Index'i siler.

Select Komutu

Select komutu veritabanından kayıtları çekmemizi sağlar.

The screenshot shows the SSMS interface with a query window titled 'SQLQuery1.sql - re...|recepaydogdu (51)*' containing the following SQL code:

```
RECEPAYDOGDU.E...E - dbo.CUSTOMER
SELECT ID, CUSTOMERNAME, CITY /* Column isimleri*/
FROM CUSTOMER

SELECT * FROM CUSTOMER /* '*' Sembolu ile tum column'lari birden cekebiliriz. */
```

The 'Results' tab is selected, displaying two tables of data:

ID	CUSTOMERNAME	CITY
1	ÖMER ÇOLAKOĞLU	İSTANBUL
2	AYŞE DENİZ	İSTANBUL
3	AHMET BURSALI	BURSA
4	ZEYNEP DEVECİ	ANKARA

ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER
1	ÖMER ÇOLAKOĞLU	İSTANBUL	KADIKÖY	1980-12-11	E
2	AYŞE DENİZ	İSTANBUL	ÜSKÜDAR	1995-12-23	K
3	AHMET BURSALI	BURSA	NİLÜFER	1992-10-18	E
4	ZEYNEP DEVECİ	ANKARA	ÇANKAYA	1994-07-15	K

Below the results, a note is displayed:

SELECT [KOLON İSMİ] FROM CUSTOMER
/*Eğer kolon ismi boşluk, turkçe karakter gibi seyler içeren bir yapıda ise koseli parantez içerisinde yazmamız gereklidir. */

Insert Komutu

Bir tabloya kayıt eklediğimiz komut.

CUSTOMER isimli tablomuzda şuan 4 kaydımız var.

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER
1	1	ÖMER ÇOLAKOĞLU	İSTANBUL	KADIKÖY	1980-12-11	E
2	2	AYŞE DENİZ	İSTANBUL	ÜSKÜDAR	1995-12-23	K
3	3	AHMET BURSALI	BURSA	Nilüfer	1992-10-18	E
4	4	ZEYNEP DEVEÇİ	ANKARA	ÇANKAYA	1994-07-15	K

INSERT INTO komutu ile yeni bir kayıt ekleyelim.

```

    SELECT
        [ID],
        [CUSTOMERNAME],
        [CITY],
        [DISTRICT],
        [BIRTHDATE],
        [GENDER]
    FROM
        CUSTOMER
    WHERE
        ID = 5
    
```

The screenshot shows an SQL query window with the following code:

```

    INSERT INTO
        CUSTOMER([CUSTOMERNAME], [CITY], [DISTRICT], [BIRTHDATE], [GENDER])
    VALUES
        ('BURCU CANDAN', 'KOCAELİ', 'MERKEZ', '1994-05-08', 'K')
    
```

Below the query window is a results grid showing the updated table:

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER
1	1	ÖMER ÇOLAKOĞLU	İSTANBUL	KADIKÖY	1980-12-11	E
2	2	AYŞE DENİZ	İSTANBUL	ÜSKÜDAR	1995-12-23	K
3	3	AHMET BURSALI	BURSA	Nilüfer	1992-10-18	E
4	4	ZEYNEP DEVEÇİ	ANKARA	ÇANKAYA	1994-07-15	K
5	5	BURCU CANDAN	KOCAELİ	MERKEZ	1994-05-08	K

Update Komutu

Veritabanındaki tablolarda herhangi bir ya da birden fazla alanı değiştirmek istediğimizde **UPDATE** komutunu kullanırız.

AGE adında yeni bir sütun ekledik.

The screenshot shows an SQL query window with the following code:

```

    SELECT * FROM CUSTOMER
    
```

Below the query window is a results grid showing the updated table:

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	1	Volkan ÇEKİP	Bartın	Kurucaşile	1997-10-24	E	NULL
2	2	Serhat GÜNDÖĞAN	Rize	Pazar / Rize	1966-02-03	E	NULL
3	3	İlmak TAHSINOĞLU	Erzincan	Çayırlı	1940-06-23	K	NULL
4	4	Yaşar SAVURGAN	Isparta	Gelendost	1991-08-07	E	NULL
5	5	Remzi ELYİĞİT	Giresun	Çanakkale	1953-09-14	E	NULL
6	6	Mehmet Akif POLAST	Bitlis	Mutki	1992-12-25	E	NULL
7	7	Salih FAYDALI	Ordu	Çatalpınar	1996-08-03	E	NULL
8	8	Münevver AYAOĞLU	Artvin	Artvin Merkez	1954-04-25	K	NULL
9	9	Muhammet KUZUCUOĞLU	Aydın	Germencik	1989-07-10	E	NULL
10	10	Nazlıcan ÖZSİMITÇİ	Şanlıurfa	Viranşehir	1951-12-29	K	NULL

```

--Yas hesaplayip AGE kolonuna yazdirmak istiyoruz.
--SQL'in kendi fonksiyonlarından olan DATEDIFF fonksiyonunu kullanacagiz.

UPDATE CUSTOMER
SET AGE=DATEDIFF(YEAR,BIRTHDATE,GETDATE())

SELECT * FROM CUSTOMER

```

133 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	1	Volkan ÇEKİP	Bartın	Kurucaşile	1997-10-24	E	23
2	2	Serhat GÜNDÖĞAN	Rize	Pazar / Rize	1966-02-03	E	54
3	3	İlmak TAHSINOĞLU	Erzincan	Çayırılı	1940-06-23	K	80
4	4	Yaşar SAVURGAN	Isparta	Gelendost	1991-08-07	E	29
5	5	Remzi ELYİĞİT	Giresun	Çanakkale	1953-09-14	E	67
6	6	Mehmet Akif POLAST	Bitlis	Mutki	1992-12-25	E	28
7	7	Salih FAYDALI	Ordu	Çatalpinar	1996-08-03	E	24
8	8	Münevver AYAOĞLU	Artvin	Artvin Merkez	1954-04-25	K	66
9	9	Muhammet KUZUCUOĞLU	Aydın	Gemencik	1989-07-10	E	31
10	10	Nazlıcan ÖZSİMİTÇİ	Şanlıurfa	Viranşehir	1951-12-29	K	69
11	11	Arya UNLUMAMULERİ	Samsun	Teme	1957-03-25	K	63
12	12	Döne GÜRBETOĞLU	Kütahya	Emet	1986-08-10	K	34
13	13	Emirhan SELİM	Samsun	Ladik	1993-01-21	E	27
14	14	Soner ÜLGEN	Elaçığ	Alacakaya	1940-12-30	E	80
15	15	Çetin BÖRKLU	Eskişehir	Çifteler	1946-09-28	E	74
16	16	Ezgi İBUKÜRTÜNCÜ	Çanakkale	Gökçeada	1985-02-16	K	35
17	17	Hazal ÖREGİL	Tunceli	Tunceli Merkez	1999-10-19	K	21
18	18	Aykut SUYUR	Rize	Rize Merkez	1964-05-17	E	56
19	19	Onur KIRIT	Knkkale	Balıgeyh	1972-02-25	E	48
20	20	Songül TÜKEZİM	Burdur	Burdur Merkez	1983-01-28	K	37
21	21	Berkay PİRİNÇAL	Sivas	Akincilar	1963-08-06	E	57
22	22	Anıl GÜLDÜ	Antalya	Gündoğmuş	1941-12-06	E	79

Delete Komutu

Tablonun içindeki verileri siler.

```

--Delete komutunu denemek icin tablomuzu kopyalayalim
SELECT * INTO CUSTOMERYEDEK FROM CUSTOMER
--CUSTOMERYEDEK isimli tabloya CUSTOMER tablosu kopyalandi.

```

DELETE FROM CUSTOMERYEDEK --Tablodaki veriler silindi.

SELECT * FROM CUSTOMERYEDEK

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE

CUSTOMERYEDEK isimli tabloya yeni bir kayıt eklediğimizde içi boş olduğundan, ID değerinin 1'den başlamasını bekleriz. Fakat öyle olmaz, önceden 1000 adet kaydımız vardı. Tabloyu DELETE ile boşalttıktan sonra yeni bir kayıt eklediğimizde ID 1001'den saymaya başlayacaktır.

Bu durumun önüne geçmek için **TRUNCATE TABLE** komutunu kullanmamız gereklidir.



TRUNCATE TABLE CUSTOMERYEDEK

Where Şartı Kullanımı

Normalde **SELECT * FROM CUSTOMER** derken tüm müşterileri çekteriz.

İsmi *Ömer Çolakoğlu* olan müşteriyi çekmek istediğimizde **WHERE** şartı bize yardımcı olacaktır.

Ya da yaşı 20'den büyük olan müşterileri çekmek istediğimiz zaman yine **WHERE** şartı kullanırız.

WHERE şartı sadece SELECT'de değil UPDATE ve DELETE komutlarında da geçerlidir.

WHERE ŞARTLARI

=	Eşittir
<>	Eşit değildir
>	Büyükür
<	Küçükür
>=	Büyükür ya da eşittir
<=	Küçükür ya da eşittir
BETWEEN	Arasındadır
LIKE	İle başlar, İle biter, İçerir
IN	İçindedir

```
--Salih FAYDALI isimli musteriyi cekelim.  
SELECT * FROM CUSTOMER  
WHERE CUSTOMERNAME='Salih FAYDALI'  
  
161 %  
Results Messages  


|   | ID | CUSTOMERNAME  | CITY | DISTRICT   | BIRTHDATE  | GENDER | AGE |
|---|----|---------------|------|------------|------------|--------|-----|
| 1 | 7  | Salih FAYDALI | Ordu | Çatalpınar | 1996-08-03 | E      | 24  |


```

```
SELECT * FROM CUSTOMER  
WHERE ID=18
```

161 %

Results Messages

ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	18 Aykut SUYUR	Rize	Rize Merkez	1964-05-17	E	56

```
SELECT * FROM CUSTOMER  
WHERE CITY='SAKARYA'
```

161 %

Results Messages

ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	35 Muhammet Ali REK.TAN.ORG.HİZ.SAN.TİC.LTD.ŞTİ.	Sakarya	Taraklı	1978-11-05	E	42
2	58 Sevda AKÇAN	Sakarya	Geyve	1964-05-12	K	56
3	69 Nazife DEVE	Sakarya	Sapanca	1967-10-13	K	53
4	140 Bema EFENDİOĞLU	Sakarya	Sapanca	1950-08-07	K	70
5	259 Sedef KÖSENÇİĞ	Sakarya	Geyve	1956-09-30	K	64
6	283 Ferhat KASAROĞLU	Sakarya	Ferizli	1996-07-30	E	24
7	354 Eray ŞAHLAN	Sakarya	Serdivan	1998-05-19	E	22
8	413 Ayhan GENÇ	Sakarya	Kaynarca	1998-03-01	E	22
9	535 Şenol GÖZEN	Sakarya	Arifiye	1955-11-21	E	65
10	594 Doruk CALARGÜN	Sakarya	Serdivan	1972-08-21	E	48
11	599 Doğukan KÜÇÜKEV	Sakarya	Kocaali	1993-02-28	E	27
12	682 Erva TACIM	Sakarya	Söğütlü	1954-11-17	K	66
13	773 Ayhan KALPAKOĞLU	Sakarya	Pamukova	1965-11-28	E	55
14	778 Gülay ÇAĞLIATALAY	Sakarya	Ferizli	1988-02-28	K	32
15	859 Sude PASİNLI	Sakarya	Adapazar	1958-10-25	K	62
16	884 İlker ANAS	Sakarya	Akyazı	1964-01-14	E	56
17	962 Sare ÇAYKUŞ	Sakarya	Erenler	1961-10-20	K	59

--Pazar / Rize ilcesi disinda Rize'deki musteriler.

```
SELECT * FROM CUSTOMER
WHERE CITY='Rize' and DISTRICT<>'Pazar / Rize'
```

.61 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	18	Aykut SUYUR	Rize	Rize Merkez	1964-05-17	E	56
2	42	Nazlı ÖRNEK	Rize	Derepazar	1949-08-24	K	71
3	57	Emin TİC.ATEŞELİĞİ	Rize	Fındıklı	1965-11-27	E	55
4	92	Helin TERKES	Rize	Fındıklı	1975-12-16	K	45
5	111	Ada VAPUR	Rize	Derepazar	1949-01-18	K	71
6	116	Güler NURKAN	Rize	Hemşin	1987-03-21	K	33
7	132	Yağız BEĞENDİ	Rize	Rize Merkez	1980-10-15	E	40
8	216	Cemre MECİT	Rize	Güneysu	1997-05-11	K	23
9	252	Ersin SEKMEN	Rize	İyidere	1975-10-19	E	45
10	269	Sebahat ELGÜN	Rize	Kalkandere	1943-11-26	K	77
11	357	Melek SÜROĞU	Rize	Çamlıhemşin	1998-10-04	K	22
12	481	Polat KIRMIZİGÜL	Rize	Çamlıhemşin	1959-05-21	E	61
13	509	İlker SENCAN	Rize	Hemşin	1981-09-22	E	39
14	540	Şükriye DİNÇALTIN	Rize	Rize Merkez	1998-03-14	K	22
15	567	Şerafettin KAMIŞ	Rize	Kalkandere	1940-07-25	E	80
16	714	Kerim HASPAL	Rize	Çamlıhemşin	1970-10-03	E	50
17	717	Türkan EVİRGEN	Rize	Güneysu	1951-05-29	K	69

BETWEEN

--21 ve 23 yas araligindaki musteriler.

```
SELECT * FROM CUSTOMER
WHERE AGE BETWEEN 20 AND 23
--AGE>=21 AND AGE <=23 ile ayni sonucu verir.
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	1	Volkan ÇEKİP	Bartın	Kuruçâle	1997-10-24	E	23
2	17	Hazal ÜREGİL	Tunceli	Tunceli Merkez	1999-10-19	K	21
3	75	Mehmet Emir SERÇE	Manisa	Sanuhanlı	1997-08-29	E	23
4	118	Abdurrahman ALTİNGÖZ	Düzce	Gölyaka	1999-08-07	E	21
5	134	Rümeysa İNCEDAL	İstanbul	Beylikdüzü	1998-03-18	K	22
6	167	Muzaffer MURT	Yozgat	Yerköy	1997-06-25	E	23
7	174	Cihan KARADELİ	Iğdır	Karakoyunlu	1997-03-26	E	23
8	180	Batuhan KARAGÜNEY	Gaziantep	Oğuzeli	1999-01-25	E	21

```
--1998 yilinda dogan musteriler
SELECT * FROM CUSTOMER
WHERE BIRTHDATE BETWEEN '19980101' AND '19981231'
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	134	Rümeysa İNCEDAL	İstanbul	Beylikdüzü	1998-03-18	K	22
2	199	Güllü SALUR	Kütahya	Gediz	1998-01-24	K	22
3	247	Güler BEŞKAYA	Ankara	Pursaklar	1998-07-21	K	22
4	250	Eren KIRBASOGLU	Konya	Akşehir	1998-08-01	E	22

LIKE

-----LIKE-----

```
SELECT * FROM CUSTOMER
WHERE CUSTOMERNAME LIKE 'AHMET'
-- WHERE CUSTOMERNAME = 'AHMET' ile aynı anlamda.
```

--ismi Ahmet ile baslayan musteriler

```
SELECT * FROM CUSTOMER
WHERE CUSTOMERNAME LIKE 'Ahmet%'
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	64	Ahmet İNCİKAPI	Muğla	Fethiye	1991-05-28	E	29
2	641	Ahmet EĞİT	Niğde	Ulukışla	1964-12-28	E	56
3	809	Ahmet ÖZTOKLU	Muğla	Kavaklıdere	1970-08-18	E	50

--isminin icerisinden 'ince' gecenler.

```
SELECT * FROM CUSTOMER
WHERE CUSTOMERNAME LIKE '%ince%'
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	44	Ali İNCEKİ	Samsun	Atakum	1949-10-12	E	71
2	134	Rümeysa İNCEDAL	İstanbul	Beylikdüzü	1998-03-18	K	22
3	393	Yunus DİNCELİR	Giresun	Piraziz	1957-04-13	E	63
4	842	Batuhan İNCEDAYI	Adana	Seyhan	1983-03-19	E	37

--isminin sonunda 'Ornek' olan musteriler.

```
SELECT * FROM CUSTOMER  
WHERE CUSTOMERNAME LIKE '%Örnek'
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	42	Nazlı ÖRNEK	Rize	Derepazar	1949-08-24	K	71

IN

-----IN-----

```
SELECT*FROM CUSTOMER  
WHERE CITY='ISPARTA' AND DISTRICT IN ('ULUBORLU', 'YALVAÇ')
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	165	Çağla BEĞEN	Isparta	Yalvaç	1991-12-22	K	29
2	396	Yasin AKÇAKOCA	Isparta	Uluborlu	1959-10-26	E	61
3	779	Tülin AKTAŞDOĞAN	Isparta	Yalvaç	1962-04-17	K	58

NOT IN

-----NOT IN-----

```
SELECT*FROM CUSTOMER  
WHERE CITY='ISPARTA' AND DISTRICT NOT IN ('ULUBORLU', 'YALVAÇ')
```

161 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	4	Yaşar SAVURGAN	Isparta	Gelendost	1991-08-07	E	29
2	26	Nimet HAYDARV	Isparta	Gelendost	1963-09-04	K	57
3	37	Tülin GÖKSUN	Isparta	Gelendost	1972-09-27	K	48
4	62	Tuğçe AKKOÇ	Isparta	Şarkikaraağaç	1958-06-08	K	62
5	277	İzzet CANKURU	Isparta	Gönen / Isparta	1978-12-11	E	42
6	468	Ekin KAYNAR	Isparta	Senirkent	1982-06-27	K	38
7	589	Hatice TEKE	Isparta	Şarkikaraağaç	1956-09-30	K	64
8	625	Yavuz ARALP	Isparta	Isparta Merkez	1969-07-03	E	51
9	674	Muhammet Ali GÜLGÖR	Isparta	Aksu / Isparta	1953-12-07	E	67
10	865	Ercan BÜR.	Isparta	Şarkikaraağaç	1999-11-02	E	21
11	984	Aras KORKMAN	Isparta	Eğirdir	1950-07-12	E	70

DELETE Komutu ile WHERE Şartı Kullanımı

```
----DELETE ve WHERE----  
  
DELETE FROM CUSTOMER  
WHERE CUSTOMERNAME LIKE 'Serhat%'  
--İsmi Serhat ile başlayanları sil  
161 %   
Messages  
  
(3 rows affected)  
  
Completion time: 2020-07-21T21:52:56.5085931+03:00
```

UPDATE Komutu ile WHERE Şartı Kullanımı

```
----UPDATE ve WHERE----  
  
UPDATE CUSTOMER SET GENDER='Erkek'  
WHERE GENDER='E'  
  
UPDATE CUSTOMER SET GENDER='Kadın'  
WHERE GENDER='K'  
  
SELECT*FROM CUSTOMER  
161 %   
Results Messages  


|   | ID | CUSTOMERNAME       | CITY     | DISTRICT      | BIRTHDATE  | GENDER | AGE |
|---|----|--------------------|----------|---------------|------------|--------|-----|
| 1 | 1  | Volkan ÇEKİP       | Bartın   | Kurucaşile    | 1997-10-24 | Erkek  | 23  |
| 2 | 3  | İmak TAHSİNÖĞLU    | Erzincan | Çayırlı       | 1940-06-23 | Kadın  | 80  |
| 3 | 4  | Yaşar SAVURGAN     | Isparta  | Gelendost     | 1991-08-07 | Erkek  | 29  |
| 4 | 5  | Remzi ELYİĞİT      | Giresun  | Çanakkale     | 1953-09-14 | Erkek  | 67  |
| 5 | 6  | Mehmet Akif POLAST | Bitlis   | Mutki         | 1992-12-25 | Erkek  | 28  |
| 6 | 7  | Salih FAYDALI      | Ordu     | Çatalpınar    | 1996-08-03 | Erkek  | 24  |
| 7 | 8  | Münevver AYAOKU    | Artvin   | Artvin Merkez | 1954-04-25 | Kadın  | 66  |


```

Distinct Komutu

DISTINCT komutunu SELECT içerisinde kullanırsınız, tekrar eden satırlar için tek bir değer döndürür.

SELECT CITY FROM CUSTOMER --997 Satır veri getirdi.

SELECT DISTINCT CITY FROM CUSTOMER --81 Satır veri getirdi.

**SELECT DISTINCT CITY, DISTRICT FROM CUSTOMER
WHERE CITY='İstanbul'**

160 %

Results Messages

	CITY	DISTRICT
1	İstanbul	Adalar
2	İstanbul	Ataşehir
3	İstanbul	Avcılar
4	İstanbul	Bayrampaşa
5	İstanbul	Beşiktaş
6	İstanbul	Beykoz
7	İstanbul	Beylikdüzü
8	İstanbul	Büyükçekmece
9	İstanbul	Esenler
10	İstanbul	Esenyurt
11	İstanbul	Fatih
12	İstanbul	Güngören
13	İstanbul	Kadıköy
14	İstanbul	Kağıthane
15	İstanbul	Maltepe
16	İstanbul	Pendik
17	İstanbul	Sancaktepe
18	İstanbul	Sarıyer
19	İstanbul	Silivri
20	İstanbul	Sultanbeyli
21	İstanbul	Tuzla
22	İstanbul	Ümraniye

Order By Komutu

ORDER BY bir sıralama komutudur.

```
SELECT * FROM CUSTOMER  
ORDER BY CUSTOMERNAME  
--Varsayılan olarak a-z sıralanır (ascending)
```

32 %

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	349	Abdulkadir AĞZIKÜÇİK	Afyonkarahisar	Başmakçı	1972-03-15	Erkek	48
2	325	Abdullah TEM.SAN.TİC.LTD.ŞTİ.	Erzincan	Otlukbeli	1949-06-28	Erkek	71
3	712	Abdulsamet EKBER	Yozgat	Çayralan	1943-06-02	Erkek	77
4	118	Abdurrahman ALTINGÖZ	Düzce	Gölyaka	1999-08-07	Erkek	21
5	40	Abdurrahman GÜNEŞDOĞDU	Sivas	Yıldızeli	1982-08-21	Erkek	38

```
SELECT*FROM CUSTOMER  
ORDER BY CUSTOMERNAME DESC  
--DESC: descending, z-a, büyükten küçeye sıralama
```

132 %

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	169	Zübeyde TATLICI	Kocaeli	Dilovası	1951-07-03	Kadın	69
2	758	Zübeyde İNAY	Yozgat	Şefaatli	1996-07-08	Kadın	24
3	270	Zihan SARIARSLAN	Afyonkarahisar	Evciler	1984-05-14	Kadın	36
4	405	Zeynep HAKOĞLU	Kars	Susuz	1968-11-05	Kadın	52
5	965	Zerda BİNNEOĞLU	Malatya	Kale / Malatya	1983-06-04	Kadın	37
6	469	Zeliha BADIL	Sivas	Suçehri	1971-11-03	Kadın	49
7	93	Zekiye HATAY	Balıkesir	Susurluk	1989-04-23	Kadın	31

```
SELECT * FROM CUSTOMER  
ORDER BY CITY, CUSTOMERNAME  
--Şehir içerisinde isme göre sıralama yapar.
```

132 %

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	98	Azad ÖNÜR	Adana	Karataş	1989-03-23	Erkek	31
2	842	Batuhan İNCEDAYI	Adana	Seyhan	1983-03-19	Erkek	37
3	506	Birsen İSFEN	Adana	Pozantı	1950-01-25	Kadın	70
4	404	Cemal KILAVUZ	Adana	Yüreğir	1999-10-12	Erkek	21
5	236	Çağla SALONU	Adana	Yüreğir	1999-04-04	Kadın	21
6	789	Çiğdem SEVENCAN	Adana	Karataş	1948-07-23	Kadın	72

```

SELECT * FROM CUSTOMER
ORDER BY CITY , CUSTOMERNAME desc
--Sehirler a-z, isimler z-a siralandi.

```

132 %

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	658	Yeşim KARABÖRKLÜ	Adana	Tufanbeyli	1982-11-23	Kadın	38
2	504	Yeşim GÖKÇÖL	Adana	Yumurtalık	1961-02-19	Kadın	59
3	492	Süleyman MOĞOLKANLI	Adana	Saimbeyli	1984-09-19	Erkek	36
4	526	Onur AKSARAY	Adana	Çukurova	1979-10-18	Erkek	41
5	196	Nisanur SARICAM	Adana	Saimbeyli	1950-06-11	Kadın	70
6	25	Muzaffer AĞAÇKESEN	Adana	Feke	1950-04-28	Erkek	70
7	203	Kezban TEM.MAD.SAN.TİC.LTD	Adana	Sançam	1946-11-23	Kadın	74
8	690	Helin ÖZMEN	Adana	İmamoğlu	1967-07-28	Kadın	53
9	627	Hayrettin İLTAR	Adana	Tufanbeyli	1959-12-23	Erkek	61
10	55	Gülsüm BİKEÇ	Adana	Seyhan	1970-07-22	Kadın	50
11	617	Emrah TOPALAN	Adana	Ceyhan	1944-09-25	Erkek	76
12	739	Emir SEZİK	Adana	Sançam	1975-01-18	Erkek	45
13	789	Çağdem SEVENCAN	Adana	Karataş	1948-07-23	Kadın	72
14	236	Çağla SALONU	Adana	Yüreğir	1999-04-04	Kadın	21
15	404	Cemal KILAVUZ	Adana	Yüreğir	1999-10-12	Erkek	21
16	506	Birsen İSFEN	Adana	Pozantı	1950-01-25	Kadın	70
17	842	Batuhan İNCEDAYI	Adana	Seyhan	1983-03-19	Erkek	37
18	98	Azad ÖNÜR	Adana	Karataş	1989-03-23	Erkek	31
19	623	Sudenur USAK	Adiyaman	Adiyaman Merkez	1995-03-10	Kadın	25
20	629	Sinem SÜEL	Adiyaman	Sincik	1967-12-10	Kadın	53
21	152	Sebahat CİLALITAŞ	Adiyaman	Sincik	1978-09-30	Kadın	42
22	918	Niyazi ERCİNS	Adiyaman	Gölbaşı / Adiyaman	1966-11-23	Erkek	54
23	449	iifan FAKOĞLU	Adiyaman	Gölbaşı / Adiyaman	1971-12-31	Erkek	49

```

SELECT * FROM CUSTOMER
ORDER BY 3 DESC
--3. KOLON CITY OLDUGUNDAN CITY'DE ISLEM YAPACAK.
--2 OLSAYDI CUSTOMERNAME'E GORE SIRALAMA YAPACAKTI.

```

132 %

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	54	Birsen AKKOR	Zonguldak	Çaycuma	1950-06-01	Kadın	70
2	125	Demet EVGİN	Zonguldak	Zonguldak Merkez	1947-08-07	Kadın	73
3	636	Saadet TİMOÇİN	Zonguldak	Kilimli	1980-03-20	Kadın	40
4	672	Tuğçe GÜNİNİ	Zonguldak	Ereğli / Zonguldak	1970-02-20	Kadın	50

Top Komutu

TOP Komutu veri setinde belirli bir sayı kadar ya da belirli bir yüzde kadar satır döndürmemizi sağlar.

--Verilen şartlara göre ilk 5 kayıt

```
SELECT TOP 5 * FROM CUSTOMER  
WHERE CITY = 'İstanbul'  
ORDER BY CUSTOMERNAME
```

132 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	427	Ali Osman ÖZELÇAM	İstanbul	Beykoz	1990-09-08	Erkek	30
2	493	Asmin TÜĞLU	İstanbul	Ümraniye	1992-10-31	Kadın	28
3	162	Can TAŞER	İstanbul	Bayrampaşa	1953-11-04	Erkek	67
4	691	Ceren ÇALIŞKAN)	İstanbul	Maltepe	1997-12-02	Kadın	23
5	549	Çınar BITGEN	İstanbul	Ümraniye	1981-01-01	Erkek	39

--Verilen şartlara göre ilk %10'luk kayıt.

```
SELECT TOP 10 PERCENT * FROM CUSTOMER  
WHERE AGE BETWEEN 22 AND 24  
ORDER BY AGE
```

132 %

Results Messages

	ID	CUSTOMERNAME	CITY	DISTRICT	BIRTHDATE	GENDER	AGE
1	134	Rümeysa İNCEDAL	İstanbul	Beylikdüzü	1998-03-18	Kadın	22
2	199	Güllü SALUR	Kütahya	Gediz	1998-01-24	Kadın	22
3	247	Güler BEŞKAYA	Ankara	Pursaklar	1998-07-21	Kadın	22
4	250	Eren KIRBASOGLU	Konya	Akşehir	1998-08-01	Erkek	22
5	354	Eray ŞAHLAN	Sakarya	Serdivan	1998-05-19	Erkek	22
6	357	Melek SÜRÜCÜ	Rize	Çamlıhemşin	1998-10-04	Kadın	22

Örnek Satış Datası

Gerçek bir veri seti üzerinde işlemler yapmak için örnek satış veritabanımızı SQL Server'a ekledik.

Aggregate Fonksiyonlar

AGGREGATE FUNCTIONS (SUM,COUNT,MIN,MAX,AVG)

```
SELECT
    SUM(PRICE),COUNT(ID),MIN(PRICE),MAX(PRICE),
    AVG(PRICE)
FROM TABLOADI
```

```
-- Ilcelerdeki toplam satislarini getirelim.
SELECT * FROM SALES
WHERE BRANCH='Kocaeli Subesi'
-- Kocaeli Subesi'de 14.292 satis var.

-- Kocaeli subesindeki satislarin toplam fiyati
SELECT SUM(LINENET) AS TOPLAMFIYAT FROM SALES
WHERE BRANCH='Kocaeli Subesi'
-- 53.151,02 TL Toplam Satis var.
```

132 %

	TOPLAMFIYAT
1	53151,02

```
SELECT SUM(LINENET) AS TOPLAM_SATIS,
       COUNT(*) AS SATIR_SAYISI,
       MIN(LINENET) AS MINIMUM_SATIS,
       MAX(LINENET) AS MAXIMUM_SATIS,
       AVG(LINENET) AS ORTALAMA_SATIS_FIYATI
FROM SALES
WHERE BRANCH='Kocaeli Subesi'
```

132 %

	TOPLAM_SATIS	SATIR_SAYISI	MINIMUM_SATIS	MAXIMUM_SATIS	ORTALAMA_SATIS_FIYATI
1	53151,02	14292	0,01	162,22	3,71893506856983

```

SELECT SUM(LINENET) AS TOPLAM_SATIS,
       COUNT(*) AS SATIR_SAYISI,
       MIN(LINENET) AS MINIMUM_SATIS,
       MAX(LINENET) AS MAXIMUM_SATIS,
       AVG(LINENET) AS ORTALAMA_SATIS_FIYATI
  FROM SALES
 WHERE BRANCH='istanbul Subesi'

```

132 %

Results Messages

	TOPLAM_SATIS	SATIR_SAYISI	MINIMUM_SATIS	MAXIMUM_SATIS	ORTALAMA_SATIS_FIYATI
1	422170,479999998	113622	0,01	1219,66	3,71556987203181

Group By Kullanımı

```

-----GROUP BY-----
--Subelere gore gruplayarak listeleyelim.
SELECT BRANCH,
       SUM(LINENET) AS TOPLAM_FIYAT,
       COUNT(*) AS SATIS_SAYISI,
       MIN(LINENET) AS MINIMUM_SATIS_FIYATI,
       MAX(LINENET) AS MAXIMUM_SATIS_FIYATI,
       AVG(LINENET) AS ORTALAMA_SATIS_FIYATI
  FROM SALES WHERE BRANCH IS NOT NULL --NULL SATIR GELMESIN
 GROUP BY BRANCH

```

132 %

Results Messages

	BRANCH	TOPLAM_FIYAT	SATIS_SAYISI	MINIMUM_SATIS_FIYATI	MAXIMUM_SATIS_FIYATI	ORTALAMA_SATIS_FIYATI
1	Artvin Subesi	5258,79	1387	0,01	284,98	3,79148521989906
2	Bolu Subesi	8027,89	2172	0,17	78,94	3,69608195211786
3	Düzce Subesi	8962,41	2521	0,09	79,21	3,55510115033717
4	Isparta Subesi	13408,83	3612	0,13	100	3,71230066445183
5	Kars Subesi	6999,05	1976	0,17	58,24	3,54202935222672
6	Mersin Subesi	46322,19	12514	0,01	250	3,70162937509989
7	Muğla Subesi	25909,44	6694	0,01	250	3,870546758291
8	Aydın Subesi	29750,69	8143	0,01	316,49	3,6535294117647
9	Batman Subesi	15619,77	4291	0,01	78,97	3,64012351433233
10	Edime Subesi	8837,73	2570	0,17	125,57	3,43880544747082

81 adet şube getirdi.

En Çok Satış Yapan Şubeler

```
--En çok satış yapan subeleri bulalım.  
SELECT TOP 5 BRANCH,  
    SUM(LINENET) AS TOPLAM_FIYAT,  
    COUNT(*) AS SATIS_SAYISI,  
    MIN(LINENET) AS MINIMUM_SATIS_FIYATI,  
    MAX(LINENET) AS MAXIMUM_SATIS_FIYATI,  
    AVG(LINENET) AS ORTALAMA_SATIS_FIYATI  
FROM SALES WHERE BRANCH IS NOT NULL  
GROUP BY BRANCH  
ORDER BY TOPLAM_FIYAT DESC  
--BUYUKTEN KUCUGE OLMASI GEREKTIGINDEN DESCENDING OLMALI
```

	BRANCH	TOPLAM_FIYAT	SATIS_SAYISI	MINIMUM_SATIS_FIYATI	MAXIMUM_SATIS_FIYATI	ORTALAMA_SATIS_FIYATI
1	Istanbul Subesi	422170,479999998	113622	0,01	1219,66	3,71556987203181
2	Ankara Subesi	153768,439999999	40695	0,01	1207,41	3,77855854527581
3	İzmir Subesi	121346,5	32328	0,01	251,83	3,75360368720612
4	Bursa Subesi	74875,23	21215	0,01	129,63	3,52935328776809
5	Antalya Subesi	69754,8400000001	17371	0,04	2761	4,01559150307985

Toplam Satış Fiyatı 50.000'den Büyük Olan Şubeler

```
--Toplam Satışı 50.000'den büyük subeler  
SELECT BRANCH,  
    SUM(LINENET) AS TOPLAM_FIYAT,  
    COUNT(*) AS SATIS_SAYISI,  
    MIN(LINENET) AS MINIMUM_SATIS_FIYATI,  
    MAX(LINENET) AS MAXIMUM_SATIS_FIYATI,  
    AVG(LINENET) AS ORTALAMA_SATIS_FIYATI  
FROM SALES WHERE BRANCH IS NOT NULL  
GROUP BY BRANCH  
HAVING SUM(LINENET)>50000 --SUM Gibi fonksiyonlar ile where kullanamayız.  
ORDER BY TOPLAM_FIYAT DESC
```

	BRANCH	TOPLAM_FIYAT	SATIS_SAYISI	MINIMUM_SATIS_FIYATI	MAXIMUM_SATIS_FIYATI	ORTALAMA_SATIS_FIYATI
1	Istanbul Subesi	422170,479999998	113622	0,01	1219,66	3,71556987203181
2	Ankara Subesi	153768,439999999	40695	0,01	1207,41	3,77855854527582
3	İzmir Subesi	121346,499999999	32328	0,01	251,83	3,75360368720612
4	Bursa Subesi	74875,2300000001	21215	0,01	129,63	3,52935328776809
5	Antalya Subesi	69754,8400000001	17371	0,04	2761	4,01559150307985
6	Adana Subesi	59889,1200000001	15862	0,01	230,93	3,77563485058631
7	Konya Subesi	58842,9600000001	15886	0,01	122,68	3,70407654538588
8	Şanlıurfa Subesi	53578,52	14824	0,01	137,29	3,61430922827847
9	Kocaeli Subesi	53151,02	14292	0,01	162,22	3,71893506856983
10	Zonguldak Subesi	53098,99	14387	0,17	444,06	3,69076179884618
11	Gaziantep Subesi	52746,3800000001	14445	0,09	137,5	3,65153201799931
12	Diyarbakır Subesi	52091,21	12385	0,01	1665,74	4,20599192571659

SUM Gibi fonksiyonlar ile koşul sağlamak istediğimizde **WHERE** yerine **HAVING** kullanmalıyız.

Bir Mağazanın Gün Bazlı Satışları

```
---GUN BAZLI SATIS SAYILARI---

SELECT BRANCH AS SUBE,
       DATE_ AS TARIH,
       SUM(LINENET) AS TOPLAM_SATIS,
       COUNT(*) AS SATISSAYISI
  FROM SALES
 WHERE BRANCH='Ankara Subesi' --AND DATE_='2017-01-05'
 GROUP BY BRANCH, DATE_
 ORDER BY DATE_
--89 günlük satis verisi verisi geldi. (3 aylık)
```

132 %

Results

Messages

	SUBE	TARIH	TOPLAM_SATIS	SATISSAYISI
1	Ankara Subesi	2017-01-02 00:00:00.000	100,88	36
2	Ankara Subesi	2017-01-03 00:00:00.000	262,85	71
3	Ankara Subesi	2017-01-04 00:00:00.000	188,22	56
4	Ankara Subesi	2017-01-05 00:00:00.000	908,96	216
5	Ankara Subesi	2017-01-06 00:00:00.000	881,36	225
6	Ankara Subesi	2017-01-07 00:00:00.000	1315,1	380
7	Ankara Subesi	2017-01-08 00:00:00.000	1443,05	389
8	Ankara Subesi	2017-01-09 00:00:00.000	1150,36	346
9	Ankara Subesi	2017-01-10 00:00:00.000	4983,96	596
10	Ankara Subesi	2017-01-11 00:00:00.000	1983,06	513
11	Ankara Subesi	2017-01-12 00:00:00.000	1450,19	387
12	Ankara Subesi	2017-01-13 00:00:00.000	1278,83	386
13	Ankara Subesi	2017-01-14 00:00:00.000	1954,4	502

Bir Gündeki Mağaza Bazlı Satışlar

```
----Bir gundeki magaza bazli satislar----  
SELECT DATE_, BRANCH, SUM(LINENET) FROM SALES  
WHERE DATE_='20170105' --Gormek istedigimiz tarih  
GROUP BY DATE_, BRANCH  
ORDER BY SUM(LINENET) DESC
```

132 %

Results Messages

	DATE_	BRANCH	(No column name)
1	2017-01-05 00:00:00.000	İstanbul Subesi	3764,82
2	2017-01-05 00:00:00.000	İzmir Subesi	1004,24
3	2017-01-05 00:00:00.000	Ankara Subesi	908,96
4	2017-01-05 00:00:00.000	Bursa Subesi	703,85
5	2017-01-05 00:00:00.000	Mersin Subesi	558,34
6	2017-01-05 00:00:00.000	Kocaeli Subesi	545,87
7	2017-01-05 00:00:00.000	Manisa Subesi	521,42
8	2017-01-05 00:00:00.000	Diyarbakır Subesi	507,36
9	2017-01-05 00:00:00.000	Adana Subesi	473,96
10	2017-01-05 00:00:00.000	Mardin Subesi	468,26
11	2017-01-05 00:00:00.000	Antalya Subesi	459,74
12	2017-01-05 00:00:00.000	Hatay Subesi	420,33
13	2017-01-05 00:00:00.000	Gaziantep Subesi	408,99
14	2017-01-05 00:00:00.000	Kahramanmaraş Subesi	405,63
15	2017-01-05 00:00:00.000	Zonguldak Subesi	392,73
16	2017-01-05 00:00:00.000	Aydın Subesi	332,28
17	2017-01-05 00:00:00.000	Isparta Subesi	305,73

```
----Bir gundeki magaza bazli satislar----  
SELECT DATE_, BRANCH, SUM(LINENET) FROM SALES  
--WHERE DATE_='20170105' --Gormek istedigimiz tarih  
GROUP BY DATE_, BRANCH  
ORDER BY DATE_, SUM(LINENET) DESC
```

132 %

Results Messages

	DATE_	BRANCH	(No column name)
1	NULL	NULL	NULL
2	2017-01-02 00:00:00.000	İstanbul Subesi	736,98
3	2017-01-02 00:00:00.000	İzmir Subesi	349,05
4	2017-01-02 00:00:00.000	Samsun Subesi	122,57
5	2017-01-02 00:00:00.000	Sakarya Subesi	111,02
6	2017-01-02 00:00:00.000	Ankara Subesi	100,88
7	2017-01-02 00:00:00.000	Konya Subesi	97,34
8	2017-01-02 00:00:00.000	Afyonkarahisar Subesi	91,35
9	2017-01-02 00:00:00.000	Hatay Subesi	88,42
10	2017-01-02 00:00:00.000	Adana Subesi	87,22

Ürün Kategorilerine Göre Toplam Satış Tutarları

----- Urun Kategorilerine Gore Satis Rakamlari-----

```
SELECT CATEGORY_NAME1, BRAND, SUM(LINENET) AS TOPLAM_SATIS_TUTARI FROM SALES
GROUP BY CATEGORY_NAME1, BRAND
ORDER BY SUM(LINENET) DESC
```

132 %

Results Messages

	CATEGORY_NAME1	BRAND	TOPLAM_SATIS_TUTARI
1	MEYVE SEBZE	HAL	234947,380000002
2	ET TAVUK	KASAP	107064,35
3	GIDA	BAKLIYAT	84623,5299999999
4	GIDA	ÜLKER	71247,7000000005
5	SİGARA	PHILIP MORIS	68565,93
6	ET TAVUK	SENPİLİÇ	54251,99
7	SÜT KAHVALTILIK	SÜTAŞ	54155,4800000002
8	İÇECEK	NULL	49122,9199999997
9	SÜT KAHVALTILIK	NULL	48512,39
10	GIDA	EMEK YAĞ	47985,2

-- 2 basamaklı

```
SELECT CATEGORY_NAME1, CATEGORY_NAME2, BRAND, SUM(LINENET) AS TOPLAM_SATIS_TUTARI
FROM SALES
GROUP BY CATEGORY_NAME1, CATEGORY_NAME2, BRAND
ORDER BY SUM(LINENET) DESC
```

132 %

Results Messages

	CATEGORY_NAME1	CATEGORY_NAME2	BRAND	TOPLAM_SATIS_TUTARI
1	MEYVE SEBZE	SEBZE	HAL	150670,22000003
2	ET TAVUK	KIRMIZI ET	KASAP	107064,35
3	GIDA	BAKLIYAT	BAKLIYAT	84623,5299999998
4	MEYVE SEBZE	MEYVE	HAL	84277,1600000008
5	SİGARA	NULL	PHILIP MORIS	68565,93
6	ET TAVUK	KÜMES HAYVANLARI	SENPİLİÇ	52720,1799999999
7	İÇECEK	ÇAY KAHVE	NULL	48872,01000002
8	GIDA	SIVI YAĞLAR	EMEK YAĞ	47985,2000000004
9	SİGARA	NULL	JTİ	45496,4499999999
10	İÇECEK	ÇAY KAHVE	ÇAYKUR	43100,5699999988

-- 3 basamaklı

```
SELECT CATEGORY_NAME1, CATEGORY_NAME2, CATEGORY_NAME3, BRAND,
       SUM(LINENET) AS TOPLAM_SATIS_TUTARI
FROM SALES
WHERE BRAND= 'ÜLKER'
GROUP BY CATEGORY_NAME1, CATEGORY_NAME2, CATEGORY_NAME3, BRAND
ORDER BY SUM(LINENET) DESC
```

132 %

Results Messages

	CATEGORY_NAME1	CATEGORY_NAME2	CATEGORY_NAME3	BRAND	TOPLAM_SATIS_TUTARI
1	GIDA	BÜSKİVİ ÇEREZ	BÜSKİVİ	ÜLKER	25955,4399999974
2	GIDA	ÇİKOLATA GOFRET	NULL	ÜLKER	25238,0099999984
3	SÜT KAHVALTILIK	TEREYAĞ MARGARİN	MARGARİNLER	ÜLKER	11231,9500000002
4	SÜT KAHVALTILIK	KAHVALTILIK	KREM ÇIKOLATA	ÜLKER	6975,53000000014
5	GIDA	SIVI YAĞLAR	AYÇİÇEK	ÜLKER	4232,66
6	GIDA	BÜSKİVİ ÇEREZ	KEK	ÜLKER	3509,6300000004
7	SÜT KAHVALTILIK	KAHVALTILIK	GEVREK MÜSLİ	ÜLKER	3132,9100000001
8	İÇECEK	ÇAY KAHVE	KAHVE	ÜLKER	2966,8400000001
9	GIDA	UNLU MAMÜLLER	PASTA MALZEMELERİ	ÜLKER	2074,7100000002
10	İÇECEK	GAZSIZ İÇECEK	MEYVE SUYU	ÜLKER	1988,9600000001

Mağazaların Müşteri Sayısını Hesaplama

----Magazalarin Musteri Sayisini Hesaplama----

```
SELECT BRANCH, COUNT(DISTINCT CLIENTNAME) FROM SALES  
GROUP BY BRANCH  
ORDER BY COUNT(DISTINCT CLIENTNAME) DESC  
--Her bir musteriyi bir kez saymasi icin DISTINCT fonksiyonunu kullandik.
```

132 %

Results Messages

	BRANCH	(No column name)
1	İstanbul Subesi	19196
2	Ankara Subesi	8167
3	İzmir Subesi	6638
4	Bursa Subesi	4677
5	Antalya Subesi	3720
6	Adana Subesi	3592
7	Konya Subesi	3501
8	Gaziantep Subesi	3265
9	Şanlıurfa Subesi	3169
10	Zonguldak Subesi	3113
11	Kocaeli Subesi	3032
12	Mersin Subesi	2767

--Bir musteri birden fazla magazadan alisveris yapmis olabilir mi?

```
SELECT CLIENTNAME, COUNT(DISTINCT BRANCH) FROM SALES  
WHERE CLIENTNAME IS NOT NULL  
GROUP BY CLIENTNAME  
HAVING COUNT(DISTINCT BRANCH)>5 --5'Den fazla magazaya giden musteriler  
ORDER BY COUNT(DISTINCT BRANCH) DESC
```

132 %

Results Messages

	CLIENTNAME	(No column name)
1	Arzu ALPER	11
2	Levent ALPEREN	11
3	İbrahim Halil ZANTUR	11
4	Nazar GÜLGÖR	10
5	Kardelen POLAST	9
6	Ezgi APARI	9
7	Baran SONBAHAR	9
8	Meliha GONCA	9
9	Masal VARUL	9
10	Bedirhan SEZGİN	9
11	Lütfiye MERTCAN	9
12	Ayaz ÖZBELLİ	9
13	Pınar İSİK	9
14	Hediye ÖRGÜ	9
15	Mira DADELEN	9

```
--Arzu ALPER'in gittiği magazalar  
SELECT DISTINCT BRANCH FROM SALES  
WHERE CLIENTNAME='Arzu ALPER'
```

132 %

Results Messages

	BRANCH
1	Afyonkarahisar Subesi
2	Amasya Subesi
3	Ankara Subesi
4	Bursa Subesi
5	İzmir Subesi
6	Kahramanmaraş Subesi
7	Kars Subesi
8	Kocaeli Subesi
9	Nevşehir Subesi
10	Niğde Subesi
11	Tokat Subesi

SQL Server Veri Tipleri

[Veri Tipleri Excel dosyası](#)ndan detayları görebiliriz.

smallint	Minimum: -2^15 (-32,768)	2 Byte
	Maksimum: 2^15-1 (32,767)	
tinyint	Minimum: 0	1 Byte
	Maksimum: 255	
bit	0 ya da 1 değerini alır.	Eğer tabloda 8 ya da daha az bit kolonu varsa 1 byte, 8'den fazla ise 2 byte yer kaplar.
decimal/ numeric	Minimum: -10^38 +1	Hassasiyetine göre diskte kapladığı alan değişir.
		1'den 9'a kadar Hassasiyet için: 5 byte
	Maksimum: 10^38 – 1.	10'dan 19'a kadar Hassasiyet için: 9 byte
		20'den 28'a kadar Hassasiyet için: 13 byte
		29'dan 38'e kadar Hassasiyet için: 17 byte
money	Minimum: -922,337,203,685,477.5808 Maksimum: 922,337,203,685,477.5807	8 Byte
smalldmoney	Minimum: -214,748.3648	4 Byte
	Maksimum: 214,748.3647	
float	-1.79308 ile -2.23308, 0	7 basamağa kadar 4 Byte
	2.23308 ile 1.79308	15 basamağa kadar 8 Byte
Real	-3.438 ile -1.1838, 0	4 Byte
	1.1838 ile 3.438	
date	Minimum: 0001-01-01	4 Byte
	Maksimum: 9999-12-31	
smalldate	Minimum: 1900-01-01	3 Byte
	Maksimum: 2079-06-06	
datetime	Minimum: 1753-01-01 00:00:00.000 Maksimum: 9999-12-31 23:59:59.997	8 Byte
datetime2	Minimum: 0001-01-01 00:00:00.0000000	1-2 Hassasiyet İçin = 6 Byte
	Maksimum: 9999-12-31 23:59:59.9999999	3-4 Hassasiyet İçin = 7 Byte
		5-7 Hassasiyet İçin = 8 Byte

datetimeoffset	Minimum: 0001-01-01 00:00:00.0000000 Maksimum: 9999-12-31 23:59:59.9999999 Time zone offset Aralığı: -14:00 / +14:00	1-2 Hassasiyet İçin = 8 Byte 3-4 Hassasiyet İçin = 9 Byte 5-7 Hassasiyet İçin = 10 Byte
time	Minimum: 00:00:00.0000000 Maksimum: 23:59:59.9999999	5 Byte(Default olarak kullanılırsa)
char	0 ile 8000 arasında	Tanımladığı değer kadar Byte. Char(10) -> 10 Byte
varchar	0 ile 8000 arasında	Tanımladığı değer + 2 Byte
varchar(MAX)	0 ile 2 147 483 647 arasında	Maksimum değeri: $2^{31}-1$ (2,147,483,647) Byte
text	0 ile 2,147,483,647 arasında	Maksimum değeri: $2^{31}-1$ (2,147,483,647) Byte
ntext	0 ile 1,073,741,823 arasında	Maksimum değeri: $2^{30}-1$ (1,073,741,823) Byte
image		Maksimum değeri: $2^{31}-1$ (2,147,483,647) Byte
binary	0 ile 8000 arasında	Tanımladığı değer kadar Byte. Binary(10) -> 10 Byte
varbinary	0 ile 8000 arasında	Tanımladığı değer + 2 Byte
varbinary(MAX)	0 ile 2 147 483 647 arasında	Tanımladığı değer + 2 Byte Maksimum değeri: $2^{31}-1$ (2,147,483,647) Byte
sql_variant		Bazı veri tiplerinin değerlerini saklamak için kullanılır. Aşağıdakiler hariç: varchar(max),varbinary(max),nvarchar(max),xml,text, ntext,image,rowversion(timestamp),sql_variant, geography,hierarchyid,geometry,User-defined types, datetimeoffset
Xml		Xml veriler için kullanılır.
Table		Sonradan kullanım amacıyla bir sonuç kümesini saklamak için kullanılır.

	<p>GUID(global olarak tekilliği garanti eder) veriyi tutar.</p> <p>select NEWID() script'ini çalıştırıldığınızda aşağıdaki gibi bir GUID veri oluşturur.</p>
uniqueidentifier	A4C5DB26-7F18-4B4F-A898-E7DE26A8446A
	<p>Bazen veritabanlarında tekilliği sağlamak için kullanılır.</p> <p>Ama bu amaçla kullanıldığından genelde performansı düşürür.</p>
hierarchyid	Hiyerarşik yapılarda, hiyerarşideki pozisyonları temsil etmek için kullanılır.
geography	Dünyadaki koordinat sistemini tutar. Dünya'nın eğimlerini de hesaba katarak.
geometry	Euclidean (flat) sistemi ile koordinat sistemini tutar. Sadece 2 düzlem üzerinden hesaplanır. Dünya'nın eğimlerini hesaba katmaz.

İlişkisel Veritabanı Sistemleri (RDMS)

ÖRNEK BİR E TİCARET SİSTEMİ

1.Kullanıcı sisteme login olur.

Adres Telefon Email gibi bilgileri sistemde kayıtlıdır.

2.Seçtiği ürün ya da ürünlerini sepete ekler

3.Sepete eklediği ürünlerin ödeme ekranına gider.

Adres listesinden adres seçer

Bu sırada siparişi oluşturulur.

4.Sepete eklediği ürünlerin ödeme ekranına gider.

Kredi kartı ödemesi gerçekleşir.

5.Urün sevk edilir.



Bu E-Ticaret sisteminin veritabanını nasıl oluşturabileceğimize [bu dosyadan](#) göz atalım.

SQL Server Tablo Oluşturma

Excel üzerinde oluşturulan tablo ve alanları SQL Server üzerinde oluşturalım.

USER TABLOSU

```
CREATE TABLE USER_(
    ID INT IDENTITY(1,1),
    USERNAME_ VARCHAR(50),
    PASSWORD_ VARCHAR(50),
    NAMESURNAME VARCHAR(100),
    EMAIL VARCHAR(100),
    GENDER VARCHAR(1),
    CREATEDDATE DATETIME,
    BIRTHDATE DATE,
    TELNR1 VARCHAR(15),
    TELNR2 VARCHAR(15)
    CONSTRAINT [PK_USER_] PRIMARY KEY CLUSTERED (ID ASC))
```

ADDRESS TABLOSU

```
CREATE TABLE ADDRESS(
    ID INT IDENTITY(1,1),
    COUNTRYID INT,
    CITYID INT,
    TOWNID INT,
    DISTRICTID INT,
    POSTALCODE VARCHAR(10),
    ADDRESSTEXT VARCHAR(250),
    USERID INT,
    CONSTRAINT [PK_ADDRESS] PRIMARY KEY CLUSTERED (ID ASC))
```

COUNTRY TABLOSU

```
CREATE TABLE COUNTRY(
    ID INT IDENTITY(1,1),
    COUNTRY VARCHAR(100),
    CONSTRAINT [PK_COUNTRY] PRIMARY KEY CLUSTERED (ID ASC))
```

TOWN TABLOSU

```
CREATE TABLE TOWN(
    ID INT IDENTITY(1,1),
    TOWN VARCHAR(100),
    CONSTRAINT [PK_TOWN] PRIMARY KEY CLUSTERED (ID ASC))
```

DISTRICT TABLOSU

```
CREATE TABLE DISTRICT(
    ID INT IDENTITY(1,1),
    DISTRICT VARCHAR(100),
    CONSTRAINT [PK_DISTRICT] PRIMARY KEY CLUSTERED (ID ASC))
```

CITY TABLOSU

```
CREATE TABLE CITY(
    ID INT IDENTITY(1,1),
    CITY VARCHAR(100),
    CONSTRAINT [PK_CITY] PRIMARY KEY CLUSTERED (ID ASC))
```

ITEM TABLOSU

```
CREATE TABLE ITEM(
    ID INT IDENTITY(1,1),
    ITEMCODE VARCHAR(20),
    ITEMNAME VARCHAR(100),
    PRICE FLOAT,
    CATEGORY1 VARCHAR(50),
    CATEGORY2 VARCHAR(50),
    CATEGORY3 VARCHAR(50),
    CONSTRAINT [PK_ITEM] PRIMARY KEY CLUSTERED (ID ASC))
```

BASKET TABLOSU

```
CREATE TABLE BASKET(
    ID INT IDENTITY(1,1),
    USERID INT,
    CREATEDDATE DATETIME,
    LASTMODIFIEDDATE DATETIME,
    ITEMCOUNT INT,
    TOTALPRICE FLOAT,
    STATUS_ INT,
    CONSTRAINT [PK_BASKET] PRIMARY KEY CLUSTERED (ID ASC))
```

BASKETDETAIL TABLOSU

```
CREATE TABLE BASKETDETAIL(
    ID INT IDENTITY(1,1),
    BASKETID INT,
    ITEMID INT,
    AMOUNT FLOAT,
    PRICE FLOAT,
    TOTALPRICE FLOAT,
    DATE_ DATETIME,
    CONSTRAINT [PK_BASKETDETAIL] PRIMARY KEY CLUSTERED (ID ASC))
```

PAYMENT TABLOSU

```
CREATE TABLE PAYMENT(
    ID INT IDENTITY(1,1),
    BASKETID INT,
    TOTALPRICE FLOAT,
    PAYMENTTYPE INT,
    DATE_ DATETIME,
    ISOK BIT,
    APPROVECODE VARCHAR(20),
    ERROR_ VARCHAR(1000),
    CONSTRAINT [PK_PAYMENT] PRIMARY KEY CLUSTERED (ID ASC))
```

ORDER TABLOSU

```
CREATE TABLE ORDER_(
    ID INT IDENTITY(1,1),
    USERID INT,
    BASKETID INT,
    CREATEDDATE DATETIME,
    ITEMCOUNT INT,
    TOTALPRICE FLOAT,
    STATUS_ INT,
    CONSTRAINT [PK_ORDER_] PRIMARY KEY CLUSTERED (ID ASC))
```

ORDERDETAIL TABLOSU

```
CREATE TABLE ORDERDETAIL(
    ID INT IDENTITY(1,1),
    ORDERID INT,
    BASKETDETAILID INT,
    ITEMID INT,
    AMOUNT FLOAT,
    PRICE FLOAT,
    TOTALPRICE FLOAT,
    CONSTRAINT [PK_ORDERDETAIL] PRIMARY KEY CLUSTERED (ID ASC))
```

INVOICE TABLOSU

```
CREATE TABLE INVOICE(
    ID INT IDENTITY(1,1),
    ORDERID INT,
    INVOICENO VARCHAR(50),
    DATE_ DATETIME,
    CARGOFICHENO VARCHAR(50),
    STATUS_ INT,
    CONSTRAINT [PK_INVOICE] PRIMARY KEY CLUSTERED (ID ASC))
```

INVOICEDETAIL TABLOSU

```
CREATE TABLE INVOICEDETAIL(
    ID INT IDENTITY(1,1),
    INVOICEID INT,
    ORDERDETAILID INT,
    ITEMID INT,
    PRICE FLOAT,
    AMOUNT FLOAT,
    CONSTRAINT [PK_INVOICEDETAIL] PRIMARY KEY CLUSTERED (ID ASC))
```

Veri Oluşturma

RDMS isimli klasördeki ETRADE2.BAK dosyasını restore database işlemi ile SQL Server'a ekledik.

Join İşlemleri

INNER JOIN

```
SELECT USER_.USERNAME_, USER_.NAMESURNAME,
       USER_.TELNR1, USER_.TELNR2, ADDRES.ADDRESSTEXT
  FROM USER_
 JOIN ADDRES ON USER_.ID=ADDRES.USERID
 WHERE USERID=1
```

Results

	USERNAME_	NAMESURNAME	TELNR1	TELNR2	ADDRESSTEXT
1	N_OZSIMITCI	Nazlıcan ÖZSİMITÇİ	(555)3852463	(532)5898448	İHSANIYE MERKEZ MAH.4375.. SOKAK NO:607 100231 GÖLC...
2	N_OZSIMITCI	Nazlıcan ÖZSİMITÇİ	(555)3852463	(532)5898448	FATİH MAH.2757. SOKAK NO:856 116027 KIRIKKALE MERKEZ...
3	N_OZSIMITCI	Nazlıcan ÖZSİMITÇİ	(555)3852463	(532)5898448	GAZİ MUSTAFA KEMALPAŞA MAH.KARACAN SOKAK NO:160 ...
4	N_OZSIMITCI	Nazlıcan ÖZSİMITÇİ	(555)3852463	(532)5898448	KONAK MAH.HAMİT ÇİNE CADDESİ SOKAK NO:673 85599 BU...

```
SELECT USER_.ID, USER_.USERNAME_, ADDRES.* FROM USER_
 INNER JOIN ADDRES ON USER_.ID=ADDRES.USERID
 WHERE USER_.ID=1
```

Results

ID	USERNAME_	ID	USERID	ADRESSTYPE	COUNTRYID	CITYID	TOWNID	DISTRICTID	POSTALCODE	ADDRESSTEXT
1	N_OZSIMITCI	1	1	0	1	41	252	100231	100231	İHSANIYE MERKEZ MAH.4375.. SOKAK NO:607 100231 GÖLC...
2	N_OZSIMITCI	2	1	1	1	71	363	116027	116027	FATİH MAH.2757. SOKAK NO:856 116027 KIRIKKALE MERKEZ...
3	N_OZSIMITCI	3	1	2	1	59	148	110896	110896	GAZİ MUSTAFA KEMALPAŞA MAH.KARACAN SOKAK NO:160 ...
4	N_OZSIMITCI	4	1	3	1	15	114	85599	85599	KONAK MAH.HAMİT ÇİNE CADDESİ SOKAK NO:673 85599 BU...

LEFT JOIN

Sol taraftaki tablonun tüm elemanlarını getirir. Sol taraftaki satırın karşılığı sağ taraftaki tabloda yoksa yerine NULL yazar.

```
-----LEFT JOIN-----
SELECT USER_.USERNAME_, ADDRES.* FROM USER_
 LEFT JOIN ADDRES ON ADDRES.USERID=USER_.ID
 WHERE USER_.ID IN (100,101,102)
--102 ID'li user'in adres kaydi yok.
```

Results

	USERNAME_	ID	USERID	ADRESSTYPE	COUNTRYID	CITYID	TOWNID	DISTRICTID	POSTALCODE	ADDRESSTEXT
1	B_OZCAGLI	249	100	0	1	16	725	86035	86035	MAKSEM MAH.3.MEHTAP SOKAK NO:598 86035 OSMANGAZI...
2	B_OZCAGLI	250	100	1	1	70	334	115902	115902	MESUDİYE KÖYÜ MAH.KOYUN KENDİSİ SOKAK NO:239 1159...
3	B_OZCAGLI	251	100	2	1	48	96	105204	105204	KONACIK-MERKEZ MAH.ŞAH CİHAN SOKAK NO:1 105204 BO...
4	I_BOLVK	252	101	0	1	6	471	81938	81938	ŞENTEPE MAH.KUTUP SOKAK NO:163 81938 POLATLI/ANKA...
5	I_BOLVK	253	101	1	1	57	520	109457	109457	YENİ MAH.AHMET YESEVI SOKAK NO:174 109457 SINOP ME...
6	I_BOLVK	254	101	2	1	38	173	99166	99166	REŞADIYE MAH.HAKAN SOKAK NO:127 99166 DEVELİ/KAYS...
7	A_VAPUR		NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

RIGHT JOIN

Sağ taraftaki tablonun tüm elemanlarını getirir. Sağ taraftaki satırın karşılığı sol taraftaki tabloda yoksa yerine NULL yazar.

```
----RIGHT JOIN----
SELECT USER_.USERNAME_, ADDRES.* FROM USER_
RIGHT JOIN ADDRES ON ADDRES.USERID=USER_.ID
WHERE USER_.ID IN (100,101,102)
--102 ID'li user'in adres kaydi yok.
```

USERNAME_	ID	USERID	ADDRESSType	COUNTRYID	CITYID	TOWNID	DISTRICTID	POSTALCODE	ADDRESSTEXT
B_OZCAGLI	249	100	0	1	16	725	86035	86035	MAKSEM MAH.3.MEHTAP SOKAK NO:598 86035 OSMANGAZI...
B_OZCAGLI	250	100	1	1	70	334	115902	115902	MESUDİYE KÖYÜ MAH.KÖYÜN KENDİSİ SOKAK NO:239 1159...
B_OZCAGLI	251	100	2	1	48	96	105204	105204	KONACIK-MERKEZ MAH.ŞAH CIHAN SOKAK NO:1 105204 BO...
I_BOLVK	252	101	0	1	6	471	81938	81938	ŞENTEPE MAH.KUTUP SOKAK NO:163 81938 POLATLI/ANKA...
I_BOLVK	253	101	1	1	57	520	109457	109457	YENİ MAH.AHMET YESEVİ SOKAK NO:174 109457 SINOP ME...
I_BOLVK	254	101	2	1	38	173	99166	99166	REŞADIYE MAH.HAKAN SOKAK NO:127 99166 DEVELİ/KAYS...

FULL JOIN

İki tablonun birleşimi.

```
----FULL JOIN----
SELECT USER_.USERNAME_, ADDRES.* FROM USER_
FULL JOIN ADDRES ON ADDRES.USERID=USER_.ID
WHERE USER_.ID IN (100,101,102)
--102 ID'li user'in adres kaydi yok.
```

USERNAME_	ID	USERID	ADDRESSType	COUNTRYID	CITYID	TOWNID	DISTRICTID	POSTALCODE	ADDRESSTEXT
B_OZCAGLI	249	100	0	1	16	725	86035	86035	MAKSEM MAH.3.MEHTAP SOKAK NO:598 86035 OSMANGAZI...
B_OZCAGLI	250	100	1	1	70	334	115902	115902	MESUDİYE KÖYÜ MAH.KÖYÜN KENDİSİ SOKAK NO:239 1159...
B_OZCAGLI	251	100	2	1	48	96	105204	105204	KONACIK-MERKEZ MAH.ŞAH CIHAN SOKAK NO:1 105204 BO...
I_BOLVK	252	101	0	1	6	471	81938	81938	ŞENTEPE MAH.KUTUP SOKAK NO:163 81938 POLATLI/ANKA...
I_BOLVK	253	101	1	1	57	520	109457	109457	YENİ MAH.AHMET YESEVİ SOKAK NO:174 109457 SINOP ME...
I_BOLVK	254	101	2	1	38	173	99166	99166	REŞADIYE MAH.HAKAN SOKAK NO:127 99166 DEVELİ/KAYS...
A_VAPUR	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ALIAS KULLANIMI

```
SELECT U.USERNAME_, C.CITY, T.TOWN , D.DISTRICT, A.ADDRESSTEXT FROM USER_ U
LEFT JOIN ADDRES A ON U.ID=A.USERID
LEFT JOIN CITY C ON A.CITYID=C.ID
LEFT JOIN TOWN T ON A.TOWNID=T.ID
LEFT JOIN DISTRICT D ON A.DISTRICTID=D.ID
WHERE U.ID IN (100, 101, 102)
```

USERNAME_	CITY	TOWN	DISTRICT	ADDRESSTEXT
B_OZCAGLI	BURSA	OSMANGAZİ	MAKSEM MAH.	MAKSEM MAH.3.MEHTAP SOKAK NO:598 86035 OSMANGAZİ...
B_OZCAGLI	KARAMAN	KARAMAN MERKEZ	MESUDİYE KÖYÜ	MESUDİYE KÖYÜ MAH.KÖYÜN KENDİSİ SOKAK NO:239 1159...
B_OZCAGLI	MUĞLA	BODRUM	KONACIK-MERKEZ MAH.	KONACIK-MERKEZ MAH.ŞAH CIHAN SOKAK NO:1 105204 BO...
I_BOLVK	ANKARA	POLATLI	ŞENTEPE MAH.	ŞENTEPE MAH.KUTUP SOKAK NO:163 81938 POLATLI/ANKA...
I_BOLVK	SİNOP	SİNOP MERKEZ	YENİ MAH.	YENİ MAH.AHMET YESEVİ SOKAK NO:174 109457 SİNOP ME...
I_BOLVK	KAYSERİ	DEVELİ	REŞADIYE MAH.	REŞADIYE MAH.HAKAN SOKAK NO:127 99166 DEVELİ/KAYS...
A_VAPUR	NULL	NULL	NULL	NULL

GROUP BY KULLANIMI

```
--GROUP BY KULLANIMI----  
--Her bir kullanıcının kaç adresi olduğunu listeleyelim  
SELECT U.ID ,U.USERNAME_ , COUNT(A.ID) ADRES SAYISI, COUNT(DISTINCT C.CITY) SEHIR SAYISI FROM USER_ U  
JOIN ADDRES A ON U.ID=A.USERID  
JOIN CITY C ON C.ID=A.CITYID  
GROUP BY U.ID ,U.USERNAME_
```

132 %

ID	USERNAME_	ADRES SAYISI	SEHIR SAYISI
1	N_OZSIMITCI	4	4
2	A_UNLUMAMULERI	1	1
3	D_GURBETOGLU	1	1
4	E_SELIM	2	2
5	S_VLGEN	2	2
6	C_BORKLV	3	3
7	E_IBUKVRTVNCV	4	4
8	H_VREGIL	2	2
9	A_SUYUR	2	2
10	O_KIRIT	1	1
11	S_TVKEZIM	1	1
12	B_PIRINCAL	3	3
13	A_GVLDV	4	4
14	P_SAN.TIC.A.S.	4	4
15	S_OKCE	4	4

--Hangi şehirde kaç kullanıcımız var?

```
SELECT C.CITY SEHIR, COUNT(U.ID) [MUSTERİ SAYISI] FROM CITY C  
JOIN ADDRES A ON A.CITYID=C.ID  
JOIN USER_ U ON U.ID=A.USERID  
GROUP BY C.CITY  
ORDER BY COUNT(U.ID) DESC
```

132 %

	SEHIR	MUSTERİ SAYISI
1	İSTANBUL	2547
2	İZMİR	1570
3	ANKARA	1042
4	KONYA	966
5	BURSA	882
6	ADANA	840
7	ANTALYA	816
8	KOCAELİ	686
9	DENİZLİ	632
10	BALIKESİR	624
11	MERSİN	609
12	MUĞLA	540
13	MANİSA	530
14	SAMSUN	483
15	AYDIN	479
16	HATAY	455

E-Ticaret Datası Sorgulama

Örnek-1

```
SELECT U.USERNAME_, B.CREATEDDATE, BD.AMOUNT, BD.PRICE,
BD.TOTAL, I.ITEMCODE, I.ITEMNAME, I.CATEGORY1,
I.CATEGORY2, I.CATEGORY3, C.CITY, T.TOWN,
D.DISTRICT ,A.ADDRESSTEXT, INV.INVOICENO,
INV.CARGOFICHENO, INV.DATE_ SHIPDATE
FROM USER_ U
INNER JOIN BASKET B ON B.USERID=U.ID
INNER JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
INNER JOIN ITEM I ON I.ID=BD.ID
INNER JOIN ORDER_ O ON O.BASKETID=B.ID
INNER JOIN ADDRES A ON A.ID=O.ADDRESSID
INNER JOIN CITY C ON C.ID=A.CITYID
INNER JOIN TOWN T ON T.ID=A.TOWNID
INNER JOIN DISTRICT D ON D.ID=A.DISTRICTID
INNER JOIN INVOICE INV ON INV.ORDERID=O.ID
```

132 %

Results Messages

	USERNAME_	CREATEDDATE	AMOUNT	PRICE	TOTAL	ITEMCODE	ITEMNAME
1	S_NAZ	2020-01-22 02:36:58.937	1	60	60	25655	Reis Gönen Baldo Pirinç 2,5 kg
2	S_NAZ	2020-01-22 02:36:58.937	2	14,5	29	25656	Reis Jasmine Pirinç 1 kg
3	S_NAZ	2020-01-22 02:36:58.937	1	2,95	2,95	25657	Reis Kepekli Diyet Pirinç 1 kg

CATEGORY1	CATEGORY2	CATEGORY3	CITY	TOWN	DISTRICT
gida-yemek-malzemeleri	makama-pirinc-ve-bakliyat	pirinc	AYDIN	DİDİM	AKBÜK-ATATÜRK MAH.
gida-yemek-malzemeleri	makama-pirinc-ve-bakliyat	pirinc	AYDIN	DİDİM	AKBÜK-ATATÜRK MAH.
gida-yemek-malzemeleri	makama-pirinc-ve-bakliyat	pirinc	AYDIN	DİDİM	AKBÜK-ATATÜRK MAH.

ADDRESSTEXT	INVOICENO	CARGOFICHENO	SHIPDATE
AKBÜK-ATATÜRK MAH.3161. SOKAK NO:714 83153 DİDİM...	24072020-023658	CRG24072020-023658	2020-01-25 05:54:19.937
AKBÜK-ATATÜRK MAH.3161. SOKAK NO:714 83153 DİDİM...	24072020-023658	CRG24072020-023658	2020-01-25 05:54:19.937
AKBÜK-ATATÜRK MAH.3161. SOKAK NO:714 83153 DİDİM...	24072020-023658	CRG24072020-023658	2020-01-25 05:54:19.937

Örnek-2

```
--Bir kullanıcının yaptığı alışveriş sayisi, adedi ve toplamı
SELECT U.USERNAME_ ADSOYAD, COUNT(DISTINCT BD.BASKETID) ALISVERISSAYISI,
       COUNT(BD.ID) ALDIGI_URUN_SAYISI,
       SUM(BD.AMOUNT) ALDIGI_URUN_MIKTARI, SUM(TOTAL) ODEDIGI_UCRET
FROM USER_ U
INNER JOIN BASKET B ON B.USERID=U.ID
INNER JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
INNER JOIN ITEM I ON I.ID=BD.ID
INNER JOIN ORDER_ O ON O.BASKETID=B.ID
INNER JOIN ADDRES A ON A.ID=O.ADDRESSID
INNER JOIN CITY C ON C.ID=A.CITYID
INNER JOIN TOWN T ON T.ID=A.TOWNID
INNER JOIN DISTRICT D ON D.ID=A.DISTRICTID
INNER JOIN INVOICE INV ON INV.ORDERID=O.ID
GROUP BY U.USERNAME_
```

132 %

Results Messages

	ADSOYAD	ALISVERISSAYISI	ALDIGI_URUN_SAYISI	ALDIGI_URUN_MIKTARI	ODEDIGI_UCRET
1	S_DABAK	1	1	1	0,8
2	E_KVNUTKUM	1	1	1	20,9
3	Z_OVGV	1	1	3	8,85
4	S_GOZVAKCA	1	3	4	56,8
5	S_NAZ	1	3	4	91,95
6	S_KODAMAN	1	4	15	121,2
7	N_ABUSKAN	1	2	5	188,23
8	E_KISAC	1	9	25	283,22
9	A_CATAKLI	1	2	4	378,25
10	R_OZSOYLU	1	9	20	227,95
11	S_ALICIK	1	1	1	2,8
12	H_KATKAYA	1	7	19	256,5

Örnek-3

```
--Hangi sehirde ne kadar alisveris yapildi
SELECT C.CITY SEHIR, COUNT(DISTINCT BD.BASKETID) ALISVERISSAYISI,
       COUNT(BD.ID) ALDIGI_URUN_SAYISI,
       SUM(BD.AMOUNT) ALDIGI_URUN_MIKTARI, SUM(TOTAL) ODEDIGI_UCRET
FROM USER_U
INNER JOIN BASKET B ON B.USERID=U.ID
INNER JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
INNER JOIN ITEM I ON I.ID=BD.ID
INNER JOIN ORDER_O ON O.BASKETID=B.ID
INNER JOIN ADDRES A ON A.ID=O.ADDRESSID
INNER JOIN CITY C ON C.ID=A.CITYID
INNER JOIN TOWN T ON T.ID=A.TOWNID
INNER JOIN DISTRICT D ON D.ID=A.DISTRICTID
INNER JOIN INVOICE INV ON INV.ORDERID=O.ID
GROUP BY C.CITY
```

132 %

	SEHIR	ALISVERISSAYISI	ALDIGI_URUN_SAYISI	ALDIGI_URUN_MIKTARI	ODEDIGI_UCRET
1	ANKARA	3	10	26	643,6
2	AYDIN	1	3	4	91,95
3	BALIKESİR	2	10	26	286,02
4	ÇORUM	1	4	15	121,2
5	DİYARBAKIR	1	1	1	0,8
6	İSTANBUL	1	1	1	20,9
7	KAHRAMANMARAŞ	1	3	4	56,8
8	MANISA	1	9	20	227,95
9	SAMSUN	1	2	5	188,23

Örnek-4

```
--En cok satis yapılan 10 musteri
SELECT TOP 10 U.USERNAME_, U.NAMESURNAME, SUM(BD.TOTAL) AS TOTAL
FROM USER_U
INNER JOIN BASKET B ON B.USERID=U.ID
INNER JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
GROUP BY U.USERNAME_, U.NAMESURNAME
ORDER BY SUM(BD.TOTAL) DESC
```

132 %

	USERNAME_	NAMESURNAME	TOTAL
1	S_TIMOCİN	Saadet TİMÖÇİN	27016
2	K_SAMLI	Kadir ŞAMLI	12507,5
3	A_SENSAN	Ayten ŞENSAN	11671,1
4	Y_ESENGİN	Yüksel ESENGİN	10633,7
5	I_CETINSOY	İhsan ÇETİNSOY	9581,66
6	Z_UCER	Zehra UÇER	8709,2
7	A_BOYLU	Azad BOYLU	8003,65
8	S_DOGULUGIL	Sinem DOĞULUGİL	7795,44
9	C_ATALAR	Cem ATALAR	7790,4
10	D_DEMIRCILER	Demet DEMİRCİLER	7712,55

Örnek-5

```
--En çok satış yapılan ürün kategorileri
SELECT I.CATEGORY1, I.CATEGORY2, I.CATEGORY3, SUM(OD.TOTALPRICE) AS TOTAL
FROM ITEM I
INNER JOIN ORDERDETAIL OD ON OD.ITEMID=I.ID
INNER JOIN ORDER_ O ON O.ID=OD.ORDERID
GROUP BY I.CATEGORY1, I.CATEGORY2, I.CATEGORY3
ORDER BY SUM(OD.TOTALPRICE) DESC
```

132 %

Results Messages

	CATEGORY1	CATEGORY2	CATEGORY3	TOTAL
1	elektronik-gida-disi	elektronik		97647,32
2	elektronik-gida-disi	beyaz-esya		83343,1
3	elektronik-gida-disi	elektronik	bilgisayar	37325,59
4	elektronik-gida-disi	kucuk-ev-aletleri		32858,51
5	elektronik-gida-disi	beyaz-esya	buzdolabi	28396
6	elektronik-gida-disi	kucuk-ev-aletleri	elektrikli-supurgeler	19626,93
7	elektronik-gida-disi	kucuk-ev-aletleri	mutfak-aletleri	17656,46
8	elektronik-gida-disi	beyaz-esya	camasir-ve-kurutma-makinaları	16112,7
9	elektronik-gida-disi	beyaz-esya	klimalar	15291
10	elektronik-gida-disi	kucuk-ev-aletleri	kisisel-bakım	8613,51
11	kisisel-bakım	kadin-urunleri		5610,05
12	kisisel-bakım	sac-bakım-ve-aksesuar		4819,18
13	elektronik-gida-disi	beyaz-esya	ocak-ve-firinlar	4616,8
14	kisisel-bakım	kadin-urunleri	yuz-bakım-urunleri	4186,83
15	elektronik-gida-disi	kucuk-ev-aletleri	utuler	4129,98

Örnek-6

```
-- 2018-2020'de alışveriş yapan kadın ve erkek müşteriler
SELECT U.GENDER, SUM(BD.TOTAL) AS TOTAL
FROM USER_ U
INNER JOIN BASKET B ON B.USERID=U.ID
INNER JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
WHERE B.CREATEDDATE BETWEEN '20180101' AND '20201231'
GROUP BY U.GENDER
```

132 %

Results Messages

	GENDER	TOTAL
1	K	257612,01
2	E	234462,12

Sub Query

SubQuery Giriş

```
--Bir musterinin kaç adet alisveris yaptigini bulalim
SELECT U.ID, U.USERNAME_, COUNT(B.ID)
FROM USER_ U
JOIN BASKET B ON B.USERID=U.ID
GROUP BY U.ID, U.USERNAME_

132 %

Results Messages


|    | ID | USERNAME_    | (No column name) |
|----|----|--------------|------------------|
| 1  | 1  | N_OZSIMITCI  | 2                |
| 2  | 8  | H_VREGIL     | 1                |
| 3  | 16 | M_AGACKESEN  | 1                |
| 4  | 24 | F_DENIZALP   | 1                |
| 5  | 27 | S_ILIS       | 2                |
| 6  | 35 | A_INCECIK    | 1                |
| 7  | 45 | B_AKKOR      | 2                |
| 8  | 53 | T_AKKOC      | 1                |
| 9  | 57 | E_OZCELIKBAS | 1                |
| 10 | 61 | C_KAZIKLI    | 1                |



--Aynı soruyu SUBQUERY ile yapmak istersek;
SELECT U.ID, U.USERNAME_,
(SELECT COUNT(B.ID) FROM BASKET B WHERE USERID=U.ID)
FROM USER_ U
WHERE (SELECT COUNT(B.ID) FROM BASKET B WHERE USERID=U.ID)>0

132 %

Results Messages


|    | ID | USERNAME_    | (No column name) |
|----|----|--------------|------------------|
| 1  | 1  | N_OZSIMITCI  | 2                |
| 2  | 8  | H_VREGIL     | 1                |
| 3  | 16 | M_AGACKESEN  | 1                |
| 4  | 24 | F_DENIZALP   | 1                |
| 5  | 27 | S_ILIS       | 2                |
| 6  | 35 | A_INCECIK    | 1                |
| 7  | 45 | B_AKKOR      | 2                |
| 8  | 53 | T_AKKOC      | 1                |
| 9  | 57 | E_OZCELIKBAS | 1                |
| 10 | 61 | C_KAZIKLI    | 1                |


```

Hız olarak Join işlemi ve SubQuery işlemi neredeyse aynıdır. Kod yazımı olarak ise join biraz daha kolay diyebiliriz. Ancak bazı örnekler var ki sadece Sub Query kullanarak yapabiliriz.

Örnek

```
--Sub Query ile musteri bilgisi getirme
SELECT U.ID, U.USERNAME_,
       (SELECT COUNT(*) FROM BASKET WHERE USERID=U.ID) AS BASKETCOUNT,
       (SELECT MIN(CREATEDDATE) FROM BASKET WHERE USERID=U.ID) AS FIRSTBASKETDATE,
       (SELECT MAX(CREATEDDATE) FROM BASKET WHERE USERID=U.ID) AS LASTBASKETDATE,
       (SELECT SUM(TOTAL) FROM BASKETDETAIL WHERE BASKETID
           IN (SELECT ID FROM BASKET WHERE USERID=U.ID)) AS TOTAL,
       (SELECT COUNT(*) FROM BASKETDETAIL WHERE BASKETID
           IN (SELECT ID FROM BASKET WHERE USERID=U.ID)) AS COUNT_
FROM USER_ U
WHERE (SELECT SUM(TOTAL) FROM BASKETDETAIL WHERE BASKETID IN
       (SELECT ID FROM BASKET WHERE USERID=U.ID))>0
```

132 %

Results Messages

ID	USERNAME_	BASKETCOUNT	FIRSTBASKETDATE	LASTBASKETDATE	TOTAL	COUNT_
1	53 T_AKKOC	1	2019-09-22 01:25:10.217	2019-09-22 01:25:10.217	462,14	7
2	117 S_ALEMDAROGLU	1	2019-11-17 01:25:04.697	2019-11-17 01:25:04.697	33,2	2
3	121 T_KAFE	1	2020-02-10 01:25:08.310	2020-02-10 01:25:08.310	368,7	9
4	330 M_ACLAN	2	2019-08-07 01:25:04.253	2020-04-08 01:25:06.010	41,03	5
5	385 N_UCUK	1	2020-05-03 01:25:04.690	2020-05-03 01:25:04.690	83,78	4
6	394 B_DIZDAR	1	2020-03-26 01:25:09.167	2020-03-26 01:25:09.167	114,6	6
7	396 Z_HAKOGLU	1	2020-05-22 01:25:14.937	2020-05-22 01:25:14.937	281,6	9
8	455 F_DONAT	1	2020-01-16 01:14:47.920	2020-01-16 01:14:47.920	144,36	4
9	564 N_PIROGLU	1	2020-03-06 01:25:07.557	2020-03-06 01:25:07.557	307,04	8
10	621 B_SEKRETER	1	2020-03-26 01:25:09.907	2020-03-26 01:25:09.907	89,49	3

132 %

```
--Aynı soruyu JOIN ile yaparsak;
SELECT U.ID, U.USERNAME_, COUNT(DISTINCT B.ID) BASKETCOUNT, MIN(B.CREATEDDATE) FIRSTBASKETDATE,
       MAX(B.CREATEDDATE) LASTBASKETDATE, SUM(BD.TOTAL) TOTAL, COUNT(DISTINCT BD.ID) COUNT_
FROM USER_ U
JOIN BASKET B ON B.USERID=U.ID
JOIN BASKETDETAIL BD ON BD.BASKETID=B.ID
GROUP BY U.ID, U.USERNAME_
```

132 %

Results Messages

ID	USERNAME_	BASKETCOUNT	FIRSTBASKETDATE	LASTBASKETDATE	TOTAL	COUNT_
1	N_OZSIMITCI	2	2020-01-31 01:25:04.627	2020-06-19 01:25:14.440	211,65	7
2	H_VREGIL	1	2019-12-06 01:25:13.367	2019-12-06 01:25:13.367	1546,2	4
3	M_AGACKESEN	1	2019-09-24 01:25:10.573	2019-09-24 01:25:10.573	119,95	5
4	F_DENIZALP	1	2020-06-06 01:25:10.143	2020-06-06 01:25:10.143	91,15	6
5	S_ILIS	2	2019-08-24 01:25:08.387	2020-04-24 01:25:08.850	1272,8	15
6	A_INCECIK	1	2020-02-28 01:25:06.633	2020-02-28 01:25:06.633	319,64	5
7	B_AKKOR	1	2020-04-20 01:25:04.777	2020-04-20 01:25:04.777	7,9	1
8	53 T_AKKOC	1	2019-09-22 01:25:10.217	2019-09-22 01:25:10.217	462,14	7
9	E_OZCELIKBAS	1	2019-12-30 01:25:13.927	2019-12-30 01:25:13.927	133	8
10	C_KAZIKLI	1	2019-08-04 01:25:12.893	2019-08-04 01:25:12.893	89,55	2

Örnek: Müşterinin Sepete Eklediği Son Ürün

```
--Musterinin Sepetine Ekleceği Son Ürün
SELECT U.ID, U.USERNAME_,
       (SELECT COUNT(*) FROM BASKET WHERE USERID=U.ID) AS BASKETCOUNT,
       (SELECT MIN(CREATEDDATE) FROM BASKET WHERE USERID=U.ID) AS FIRSTBASKETDATE,
       (SELECT MAX(CREATEDDATE) FROM BASKET WHERE USERID=U.ID) AS LASTBASKETDATE,
       (SELECT SUM(TOTAL) FROM BASKETDETAIL WHERE BASKETID
          IN (SELECT ID FROM BASKET WHERE USERID=U.ID)) AS TOTAL,
       (SELECT COUNT(*) FROM BASKETDETAIL WHERE BASKETID
          IN (SELECT ID FROM BASKET WHERE USERID=U.ID)) AS COUNT_,
       (SELECT ITEMNAME FROM ITEM WHERE ID IN
          (SELECT TOP 1 ITEMID FROM BASKETDETAIL WHERE BASKETID IN
             (SELECT ID FROM BASKET WHERE USERID=U.ID))
          ORDER BY DATE_ DESC
       )) AS LASTITEMNAME
FROM USER_ U
WHERE (SELECT SUM(TOTAL) FROM BASKETDETAIL WHERE BASKETID IN
       (SELECT ID FROM BASKET WHERE USERID=U.ID))>0
```

132 %

Results Messages

ID	USERNAME_	BASKETCOUNT	FIRSTBASKETDATE	LASTBASKETDATE	TOTAL	COUNT_	LASTITEMNAME
1	N_OZSIMITCI	2	2020-01-31 01:25:04.627	2020-06-19 01:25:14.440	211,65	7	Elit Bitter Fındık Draje 250 g
2	A_INCECIK	1	2020-02-28 01:25:06.633	2020-02-28 01:25:06.633	319,64	5	Carrefour Enİte Makama 500 g
3	E_BVYKTVLV	1	2020-01-24 01:25:07.903	2020-01-24 01:25:07.903	464	2	Exper Easypad T7Q Tablet
4	L_BIRINCI	2	2019-10-05 01:25:06.767	2020-04-20 01:25:09.770	1118,15	10	Le Petit Marseillais Mandalina Limon Duş Jeli 2...
5	E_IN.TUR.OTO.LTD.	1	2020-04-11 01:25:13.217	2020-04-11 01:25:13.217	41,15	3	Pınar Ton Balık 3'80 g
6	D_HACMALZEMELERI	1	2020-04-23 01:25:08.643	2020-04-23 01:25:08.643	17,3	2	Parex Canlı Renkler Oluklu Sünger 3lü
7	B_AZGIT	1	2020-01-04 01:25:10.920	2020-01-04 01:25:10.920	34,3	3	Hünkar Pilavlık Bulgur 2,5 kg
8	C_CAYCI	1	2020-03-15 01:25:07.630	2020-03-15 01:25:07.630	1709,9	9	Komili Bebek Şampuan Kremlı 750 ml
9	M_ENES	1	2020-02-09 01:25:06.853	2020-02-09 01:25:06.853	167,1	7	Elit Truffle Çikolata 135 g
10	Z_KACAN	1	2020-06-03 01:25:07.010	2020-06-03 01:25:07.010	151,9	6	Bingo Mutfak Temizleyici 750 ml + Banyo Spreyi

String İşlemleri

ASCII VE CHAR

ASCII fonksiyonu içerisinde aldığı karakterin ASCII türünden değerini verir. (Byte olarak karşılığını verir.)

CHAR fonksiyonu ise ASCII fonksiyonunun tersidir.

```
--ASCII VE CHAR FONKSIYONLARI
SELECT ASCII ('A') --64
-- A KARAKTERININ ASCII KARSILIGI

--CHAR'DA ASCII'NIN TERS FONKSIYONUDUR
SELECT CHAR (65) --A
-- 65 ASCII DEGERININ CHAR KARSILIGI

132 %
Results Messages
(No column name)
1 65

(No column name)
1 A

SELECT ASCII(CHAR(65)) --65
SELECT CHAR(ASCII('A')) --A

132 %
Results Messages
(No column name)
1 65

(No column name)
1 A
```

SUBSTRING

SUBSTRING fonksiyonu bir string içerisinde belirli bir yerden bir yere kadar olan bölümü almamızı sağlar.

```
-----SUBSTRING FONKSIYONU-----
SELECT SUBSTRING('RECEP AYDOGDU', 3, 5)
--3. KARakterden itibaren 5 KARakter alır.

132 %
Results Messages
(No column name)
1 CEP A
```

CHAR INDEX

Bir stringin içerisinde başka bir stringi buldurup onun pozisyonunu öğrenmemizi sağlar.

```

-----CHAR INDEX FONKSIYONU-----
SELECT CHARINDEX('A', 'RECEP AYDOGDU', 1)
132 %
Results Messages
(No column name)
1 7

```

CONCAT, CONCAT_WS

CONCAT, iki veya daha fazla stringin yan yana birleştirilmesini sağlar.

```

-----CONCAT, CONCAT_WS-----
SELECT CONCAT('RECEP', 'AYDOGDU')
132 %
Results Messages
(No column name)
1 RECEPAYDOGDU

SELECT CONCAT_WS('**', 'A', 'B', 'C')
132 %
Results Messages
(No column name)
1 A**B**C

SELECT
    USERNAME_ + ' ' + PASSWORD_ AS NORMAL,
    CONCAT(USERNAME_, ' ', PASSWORD_) AS CONCAT_,
    CONCAT_WS(' ', USERNAME_, PASSWORD_) AS CONCAT_WS_
    FROM USER_

```

	NORMAL	CONCAT_	CONCAT_WS_
1	N_OZSIMITCI doropipa	N_OZSIMITCI doropipa	N_OZSIMITCI doropipa
2	A_UNLUMAMULERI boerke80	A_UNLUMAMULERI boerke80	A_UNLUMAMULERI boerke80
3	D_GURBETOGLU theebryantbitch	D_GURBETOGLU theebryantbitch	D_GURBETOGLU theebryantbitch
4	E_SELIM xpais	E_SELIM xpais	E_SELIM xpais
5	S_VLGEN 4042521	S_VLGEN 4042521	S_VLGEN 4042521
6	C_BORKLV audraia	C_BORKLV audraia	C_BORKLV audraia
7	E_IBUKVRTVNCV translatlantic	E_IBUKVRTVNCV translatlantic	E_IBUKVRTVNCV translatlantic
8	H_VREGIL happy2shoes	H_VREGIL happy2shoes	H_VREGIL happy2shoes
9	A_SUYUR im_nthelordsarmy	A_SUYUR im_nthelordsarmy	A_SUYUR im_nthelordsarmy
10	O_KIRIT sugarandspice39	O_KIRIT sugarandspice39	O_KIRIT sugarandspice39

FORMAT

Sayı yada tarih türündeki değerleri local olarak istediğimiz formatta, istediğimiz dile uygun şekilde yazdırımızı sağlar.

```
----FORMAT FONKSIYONU----  
SELECT FORMAT(GETDATE(), 'D', 'tr')  
SELECT FORMAT(GETDATE(), 'D', 'EN-US')  
SELECT FORMAT(GETDATE(), 'd', 'TR')  
SELECT FORMAT(GETDATE(), 'd', 'EN-US')  
132 %  
Results Messages  


|                  |                      |
|------------------|----------------------|
| (No column name) |                      |
| 1                | 26 Temmuz 2020 Pazar |


|                  |                       |
|------------------|-----------------------|
| (No column name) |                       |
| 1                | Sunday, July 26, 2020 |


|                  |            |
|------------------|------------|
| (No column name) |            |
| 1                | 26.07.2020 |


|                  |           |
|------------------|-----------|
| (No column name) |           |
| 1                | 7/26/2020 |


```

LEFT, RIGHT, LEN

```
----LEFT, RIGHT, LEN FONKSIYONLARI----  
SELECT LEFT('RECEP AYDOGDU', 5)  
--SOLDAN 5 KARAKTER ALIR.  
132 %  
Results Messages  


|                  |       |
|------------------|-------|
| (No column name) |       |
| 1                | RECEP |


```
SELECT RIGHT('RECEP AYDOGDU',7)
--SAGDAN 7 KARAKTER ALIR.
```

  
132 %  
Results Messages  


|                  |         |
|------------------|---------|
| (No column name) |         |
| 1                | AYDOGDU |


```
SELECT LEFT('RECEP AYDOGDU',CHARINDEX(' ', 'RECEP AYDOGDU')),
RIGHT('RECEP AYDOGDU',LEN('RECEP AYDOGDU')-CHARINDEX(' ', 'RECEP AYDOGDU'))
```

  
132 %  
Results Messages  


|                  |                  |
|------------------|------------------|
| (No column name) | (No column name) |
| 1                | RECEP AYDOGDU    |


```

```
-- USER_ TABLOSUNDA DENEYELIM
SELECT LEFT(NAMESURNAME,CHARINDEX(' ',NAMESURNAME)) AS NAME,
       RIGHT(NAMESURNAME,LEN(NAMESURNAME)-CHARINDEX(' ',NAMESURNAME)) AS SURNAME
FROM USER_

```

132 %

	NAME	SURNAME
1	Nazlican	ÖZSİMITÇİ
2	Arya	UNLUMAMULERİ
3	Döne	GURBETOĞLU
4	Emirhan	SELİM
5	Soner	ULGEN
6	Çetin	BÖRKLO
7	Ezgi	İBUKÜRTÖNCÜ
8	Hazal	ÜREGİL
9	Aykut	SUYUR
10	Onur	KIRIT

TRIM, LTRIM, RTRIM

Boslukları temizlemek için kullanılır.

```
----TRIM, LTRIM, RTRIM FONKSİYONLARI----
SELECT TRIM(' RECEP AYDOGDU ')
--BASINDAKI VE SONUNDAKI BOSLUGU TEMİZLER

SELECT LTRIM(' RECEP AYDOGDU ')
--SOLUNDAKI BOSLUGU TEMİZLER

SELECT RTRIM(' RECEP AYDOGDU ')
--SAGINDAKI BOSLUGU TEMİZLER

SELECT LTRIM(RTRIM(' RECEP AYDOGDU '))
-- SOL VE SAGDAKİ BOSLUGU TEMİZLER
```

LOWER, UPPER, REVERSE, REPLICATE

```
----LOWER, UPPER, REVERSE, REPLICATE| FONKSIYONLARI----  
  
SELECT LOWER('RECEP')  
-- KUCUK HARFE CEVIRIR  
  
SELECT UPPER('recep')  
-- BUYUK HARFE CEVIRIR  
  
SELECT REVERSE('RECEP AYDOGDU')  
-- TERS YAZDIRIR.  
  
132 % < Results Messages  


|                  |
|------------------|
| (No column name) |
|------------------|

| 1 recep |
  
  


|                  |
|------------------|
| (No column name) |
|------------------|

| 1 RECEP |
  
  


|                  |
|------------------|
| (No column name) |
|------------------|

| 1 UDGODYA PECER |

```

```
SQLQuery21.sql - (L...\\Administrator (98)) * -> SQLQuery20.sql - (L...\\Administrator (95)) * SQLQuery19.sql - (L...\\Administrator (97)) * SQLQuery18.sql - (L...\\Ad  
[ ] SELECT *,REPLICATE('0',8-LEN(SIRANO))+CONVERT(VARCHAR,SIRANO) FROM TEST  
[ ] UPDATE TEST SET SIRANO2=REPLICATE('0',8-LEN(SIRANO))+CONVERT(VARCHAR,SIRANO)  
  
159 % < Results Messages  


| SIRANO | SIRANO2  | (No column name) |
|--------|----------|------------------|
| 1      | 00000001 | 00000001         |
| 2      | 00000002 | 00000002         |
| 3      | 00000003 | 00000003         |
| 4      | 00000004 | 00000004         |
| 5      | 00000005 | 00000005         |
| 6      | 00000006 | 00000006         |
| 7      | 00000007 | 00000007         |
| 8      | 00000008 | 00000008         |
| 9      | 00000009 | 00000009         |
| 10     | 00000010 | 00000010         |
| 11     | 00000011 | 00000011         |


Activate Windows  
Query executed successfully. (local) (14.0 RTM) \SQLLEGITIM\Administrato...


```

REPLACE

Bir şey ile başka bir şeyi yer değiştiren fonksiyondur.

-----REPLACE FONKSİYONU-----

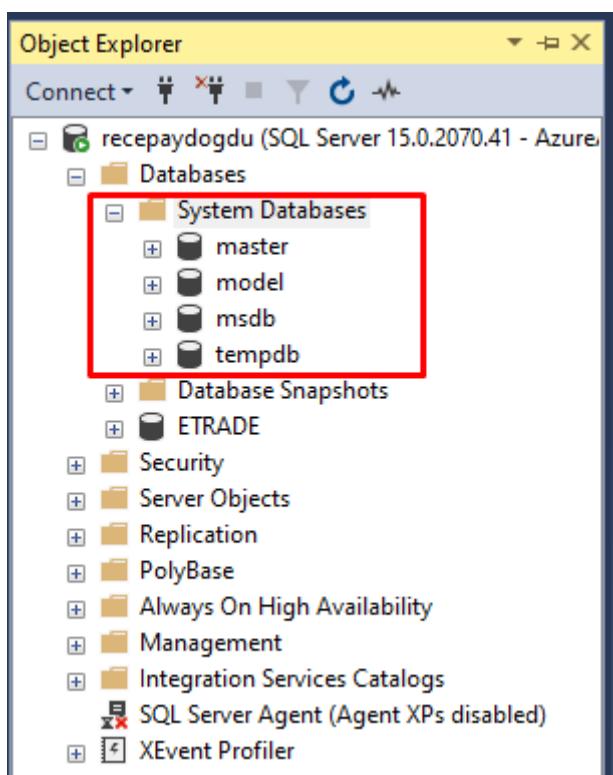
```
SELECT REPLACE('DEGISECEK AYDOĞDU', 'DEGISECEK', 'DEGİSTİ')
```

Uçtan Uca SQL Server

Sistem Database'leri

SQL Server kendisini yönetirken bazı databaseler ile yönetir, biz bunlara Sistem Database'leri deriz.

SİSTEM DATABASE'LERİ



Master DB

MASTER database'inin dosyası bizim için en önemlidir.

Master DB olmazsa SQL Server çalışmaz.

MASTER DB

- **Sistem Konfigürasyonu**
- **Kullanıcılar**
- **Veritabanları**
- **Sistem Dosyaları**
- **Collation Bilgisi**
- **Gibi SQL Server sisteminin temel konfigürasyon bilgilerini tutar.**

Model DB

MODEL database'i ise yeni bir database oluştururken baz alınan database'dır.

Model DB olmazsa SQL Server çalışmaz.

MODEL DB

- **Şablon veritabanıdır.**
- **Her bir oluşturulacak veritabanı ModelDB'nin bir kopyası olarak oluşturulur.**
- **Her veritabanında otomatik olarak olmasını istediğimiz tipler, fonksiyonlar, tablolar vs varsa bu veritabanının içine konulabilir.**

MSDB

MSDB

- **SQL Server Agent servisinin kullandığı veritabanıdır.**
- **Periyodik olarak çalıştırılan her türlü işlem (Joblar, schedule lar, alertler) burada tutulur.**

Temp DB

TEMPDB'de ise geçici tablolar tutulur. Geçici tablo oluşturmak için tablo oluşturulurken başına # işaret konulur. Tek # işaretini kullanarak oluşturulan tablolara sadece oluşturulduğu sessiondan ulaşılabilir. Çift # yani ## işaretini kullanarak oluşturulan tablolara ise farklı sessionlardan da ulaşılabilir.

TEMP DB

- **Geçici tabloların oluşturulduğu işlemler burada gerçekleşir.**
- **Kullanıcı veritabanlarından çekilendataları group by,order by, sum, count,max,min gibi komutlarla özetlemek amacıyla kullandığımız aggregation işlemleri için de temp dB kullanılır.**

T-SQL Kodları

Data Definition Language (DDL) Kodları

```
CREATE DATABASE TEST
--TABLO OLUSTURMA
CREATE TABLE TEST (ID INT IDENTITY(1,1), ISIM VARCHAR(20))

--TABLOYA COLUMN EKLEME
ALTER TABLE TEST ADD ADRES VARCHAR(500)

--TABLODAN COLUMN SILME
ALTER TABLE TEST DROP COLUMN ADRES

--TABLODA COLUMN GUNCELLEME
ALTER TABLE TEST ALTER COLUMN ISIM VARCHAR(50)
```

Değişken Kullanımı

```
DECLARE @ISIM AS VARCHAR(50)
SET @ISIM='RECEP'

DECLARE @SAYI AS INTEGER
SET @SAYI=15

DECLARE @SAYI2 AS INTEGER
SET @SAYI2=20

DECLARE @TOPLAM AS INTEGER
SET @TOPLAM=@SAYI+@SAYI2

SELECT @ISIM AS ISIM, @SAYI AS SAYI, @SAYI2 AS SAYI2, @TOPLAM AS TOPLAM
```

	ISIM	SAYI	SAYI2	TOPLAM
1	RECEP	15	20	35

Tablodan Gelen Değeri Değişkene Atama

```
--tablodan gelen degeri degiskene atama
DECLARE @ISIM AS VARCHAR(50),
        @TELEFON AS INT

SELECT
    @ISIM=ISIMLER, @TELEFON=TEL
FROM TEST
WHERE ID=1

SELECT @ISIM, @TELEFON --ATANAN DEGERLERİ GORELİM
```

132 %

Results Messages

	(No column name)	(No column name)
1	OMER	12345678

Date Time fonksiyonları

DATEADD

Belirlediğimiz tarihe ekleme yapmamızı sağlar.

```
----DATE TIME-FONKSIYONLARI----
--DATEADD
DECLARE @TARIH AS DATETIME
SET @TARIH='2019-01-01 16:21:37'
DECLARE @TARIH2 AS DATETIME
SET @TARIH2=DATEADD(MONTH,6,@TARIH)
SELECT @TARIH, @TARIH2
```

132 %

Results Messages

	(No column name)	(No column name)
1	2019-01-01 16:21:37.000	2019-07-01 16:21:37.000

DATEDIFF

İki tarih arasındaki farkı hesaplar.

```
--DATEDIFF
--İki tarih arasındaki farkı hesaplar
DECLARE @TARIH AS DATETIME
SET @TARIH='2019-01-01 16:21:37'
DECLARE @TARIH2 AS DATETIME
SET @TARIH2='2019-07-01 16:21:37'
SELECT DATEDIFF(MONTH, @TARIH, @TARIH2) AS FARK
```

132 %

Results Messages

FARK
1 6

DATEFROMPARTS

İçine girdiğimiz değerler ile tarih oluşturur.

```
--DATEFROMPARTS
DECLARE @TARIH AS DATE
SELECT @TARIH=DATEFROMPARTS(2020,07,20)
SELECT @TARIH
```

132 %

Results Messages

(No column name)
1 2020-07-20

DATEPART

İçine girdiğimiz tarihin istediğimiz bölümünü verir.

```
--DATEPART
DECLARE @TARIH AS DATE
SELECT @TARIH=DATEFROMPARTS(2020,07,20)
SELECT DATEPART(YEAR, @TARIH)
```

132 %

Results Messages

(No column name)
1 2020

T-SQL'de Döngüler

```
Loops.sql - recepa...\\recepaydo\u0111du (51)* ✎ X
CREATE TABLE TARIHLER (ID INT IDENTITY(1,1), TARIH DATETIME2(7))

DECLARE @I AS INT=0
WHILE @I<10
BEGIN
    INSERT INTO TARIHLER (TARIH) VALUES (GETDATE())
    SET @I=@I+1
END

SELECT * FROM TARIHLER
```

132 %

ID	TARIH
1	2020-07-29 00:32:25.2333333
2	2020-07-29 00:32:25.2333333
3	2020-07-29 00:32:25.2333333
4	2020-07-29 00:32:25.2333333
5	2020-07-29 00:32:25.2333333
6	2020-07-29 00:32:25.2366667
7	2020-07-29 00:32:25.2366667
8	2020-07-29 00:32:25.2366667
9	2020-07-29 00:32:25.2366667
10	2020-07-29 00:32:25.2366667

```
Loops.sql - recepa...\\recepaydo\u0111du (51)* ✎ X
CREATE TABLE TARIHLER (ID INT IDENTITY(1,1), TARIH DATETIME2(7))

DECLARE @I AS INT=0
WHILE @I<10
BEGIN
    INSERT INTO TARIHLER (TARIH) VALUES (GETDATE())
    WAITFOR DELAY '00:00:02' --2 saniye bekleyerek g\u00f6r\u00fclmek istenir.
    SET @I=@I+1
END

SELECT * FROM TARIHLER
```

132 %

ID	TARIH
1	2020-07-29 00:34:31.2266667
2	2020-07-29 00:34:33.2300000
3	2020-07-29 00:34:35.2300000
4	2020-07-29 00:34:37.2333333
5	2020-07-29 00:34:39.2366667
6	2020-07-29 00:34:41.2400000
7	2020-07-29 00:34:43.8833333
8	2020-07-29 00:34:45.8833333
9	2020-07-29 00:34:47.8866667
10	2020-07-29 00:34:49.8900000

Rastgele Kişi Oluşturma

```
--RASTGELE KİŞİ OLUSTURMA
DECLARE @I AS INT=0
DECLARE @AD AS VARCHAR(50)
DECLARE @CINSIYET AS VARCHAR(1)
DECLARE @SOYAD AS VARCHAR(50)
DECLARE @DOGUMTARIHI AS DATE
DECLARE @YAS AS INT
DECLARE @YASGRUBU AS VARCHAR(30)

WHILE @I<1000
BEGIN
    SELECT @AD=AD, @CINSIYET=CINSIYET FROM ISIMLER WHERE ID=ROUND(RAND()*611,0)
    SELECT @SOYAD=SOYISIM FROM SOYISIMLER WHERE ID=ROUND(RAND()*1000,0)
    SET @DOGUMTARIHI=DATEADD(DAY,ROUND(RAND()*18250,0), '1950-01-01')
    SET @YAS=DATEDIFF(YEAR,@DOGUMTARIHI,GETDATE())

    IF @YAS BETWEEN 10 AND 20
        SET @YASGRUBU='20 YAS ALTI'
    IF @YAS BETWEEN 21 AND 30
        SET @YASGRUBU='20-30 ARASI'
    IF @YAS BETWEEN 31 AND 40
        SET @YASGRUBU='30-40 ARASI'
    IF @YAS BETWEEN 41 AND 50
        SET @YASGRUBU='40-50 ARASI'
    IF @YAS BETWEEN 51 AND 60
        SET @YASGRUBU='50-60 ARASI'
    IF @YAS>60
        SET @YASGRUBU='60' 'DAN BUYUK'

    INSERT INTO KISILER (AD,SOYAD,CINSIYET,DOGUMTARIHI,YAS,YASGRUBU)
    VALUES (@AD, @SOYAD, @CINSIYET, @DOGUMTARIHI, @YAS, @YASGRUBU)

    SET @I=@I+1
END

SELECT * FROM KISILER
```

	ID	AD	SOYAD	CINSIYET	DOGUMTARIHI	YAS	YASGRUBU
1	1	Ramazan	ÇAVDAROĞLU	E	1985-12-10	35	30-40 ARASI
2	2	Şenol	KIYAR	E	1996-06-13	24	20-30 ARASI
3	3	Esra	KELAMLI	K	1955-02-17	65	60'DAN BUYUK
4	4	Yunus Emre	KAZDAĞ	E	1963-12-30	57	50-60 ARASI
5	5	Nihat	ÖZGÖNCÜ	E	1971-03-16	49	40-50 ARASI
6	6	Serhat	ANIK	E	1992-02-28	28	20-30 ARASI
7	7	Yakup	ŞANLITÜRK	E	1970-01-30	50	40-50 ARASI
8	8	Yakup	KÖKER	E	1989-06-13	31	30-40 ARASI
9	9	Semih	MERTCAN	E	1954-08-29	66	60'DAN BUYUK
10	10	Polat	GÜRSU	E	1967-03-20	53	50-60 ARASI

Index

SQL SERVER'DA INDEX KAVRAMI

Veritabanı yönetim sistemlerinin temel misyonu veriyi yönetmektir.



SQL SERVER'DA INDEX KAVRAMI

Genel ortalamaya baktığımızda Veritabanı Yönetim Sistemlerinin %95 oranında okuma, %5 oranında yazma yaptığını görüyoruz.

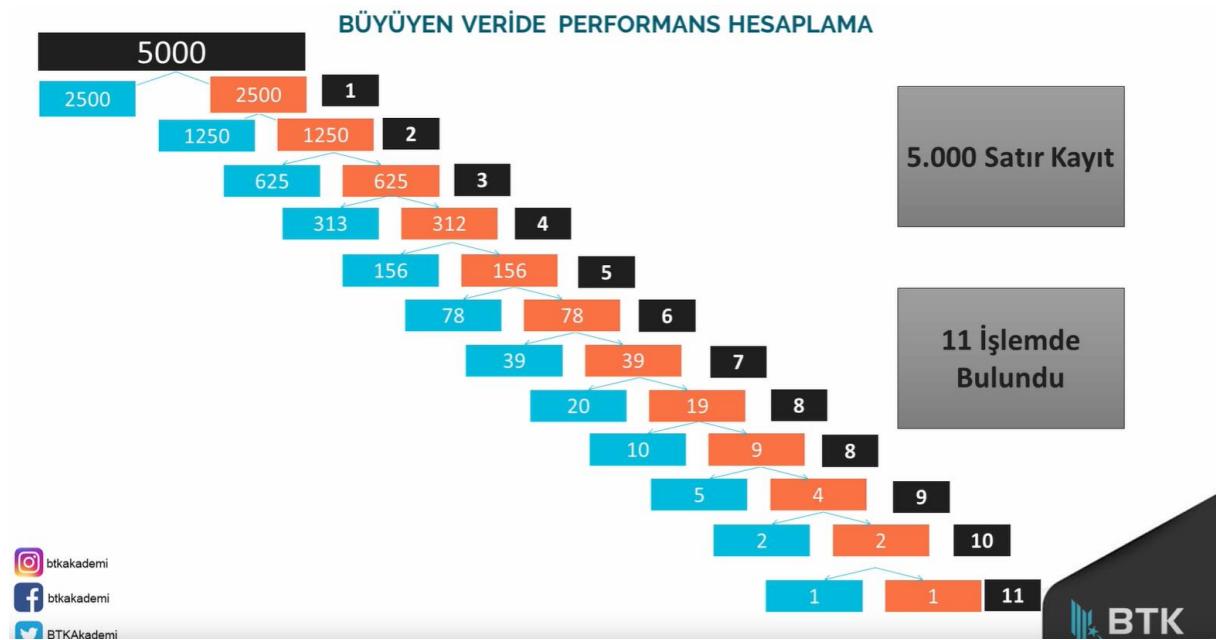


SQL SERVER'DA INDEX KAVRAMI

Bu durumda okuma performansını artıracak her türlü işlem veritabanı sunucunun performansının artırılmasında doğrudan etkili olacaktır.



Büyük Veride Performans Hesaplama



BÜYÜKEN VERİDE PERFORMANS HESAPLAMA

Kayıt Sayısı	İşlem Sayısı (Indexsiz)	İşlem Sayısı (Indexli)
5.000	5.000	11
10.000	10.000	12
20.000	20.000	13
40.000	40.000	14
80.000	80.000	15
160.000	160.000	16
320.000	320.000	17
640.000	640.000	18

BÜYÜKEN VERİDE PERFORMANS HESAPLAMA

Kayıt Sayısı	İşlem Sayısı (Indexsiz)	İşlem Sayısı (Indexli)
1.000.000	1.000.000	19
2.000.000	2.000.000	20
4.000.000	4.000.000	21
8.000.000	8.000.000	22
16.000.000	16.000.000	23
32.000.000	32.000.000	24
64.000.000	64.000.000	25
128.000.000	128.000.000	26

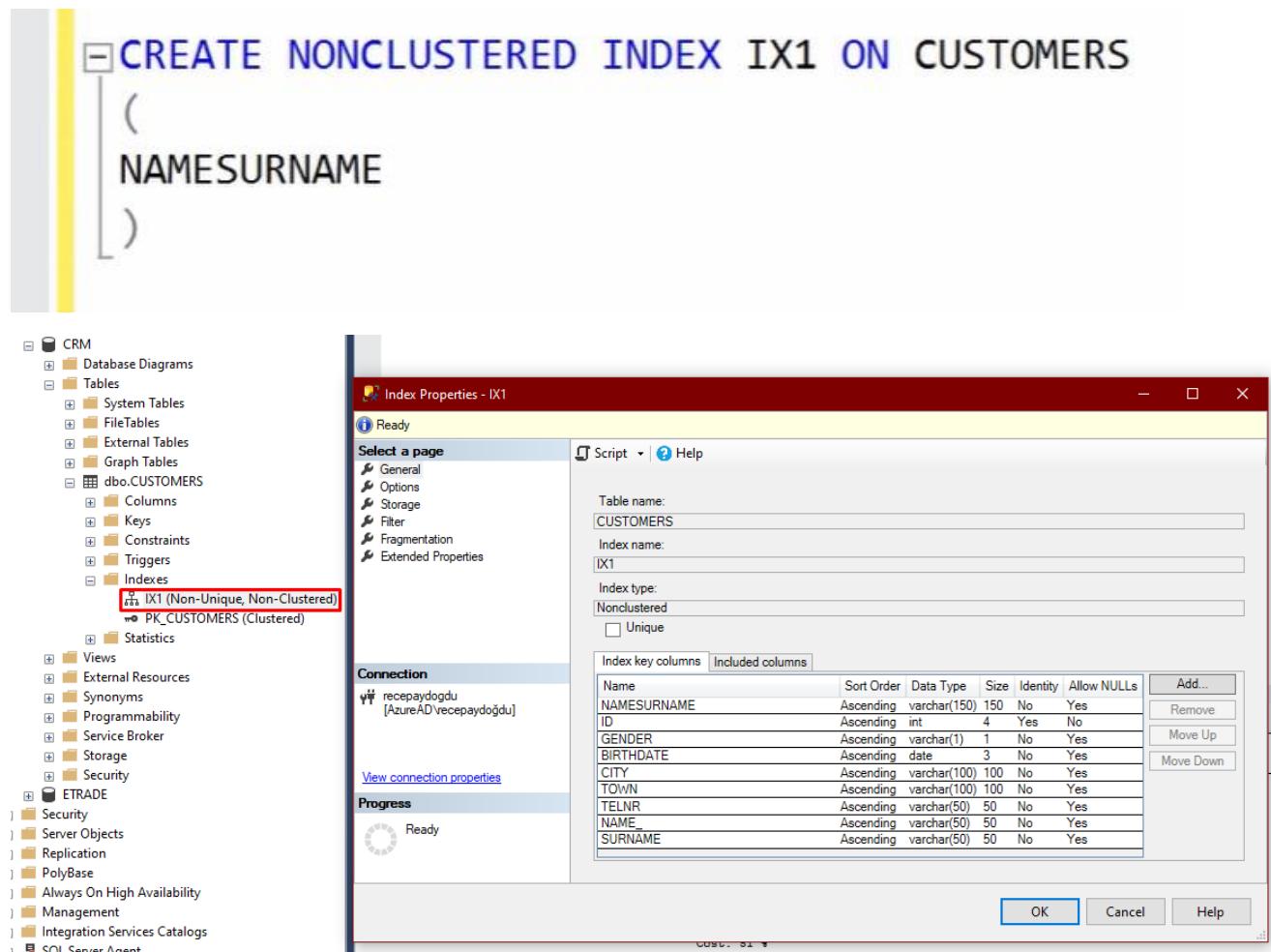
BÜYÜKEN VERİDE PERFORMANS HESAPLAMA



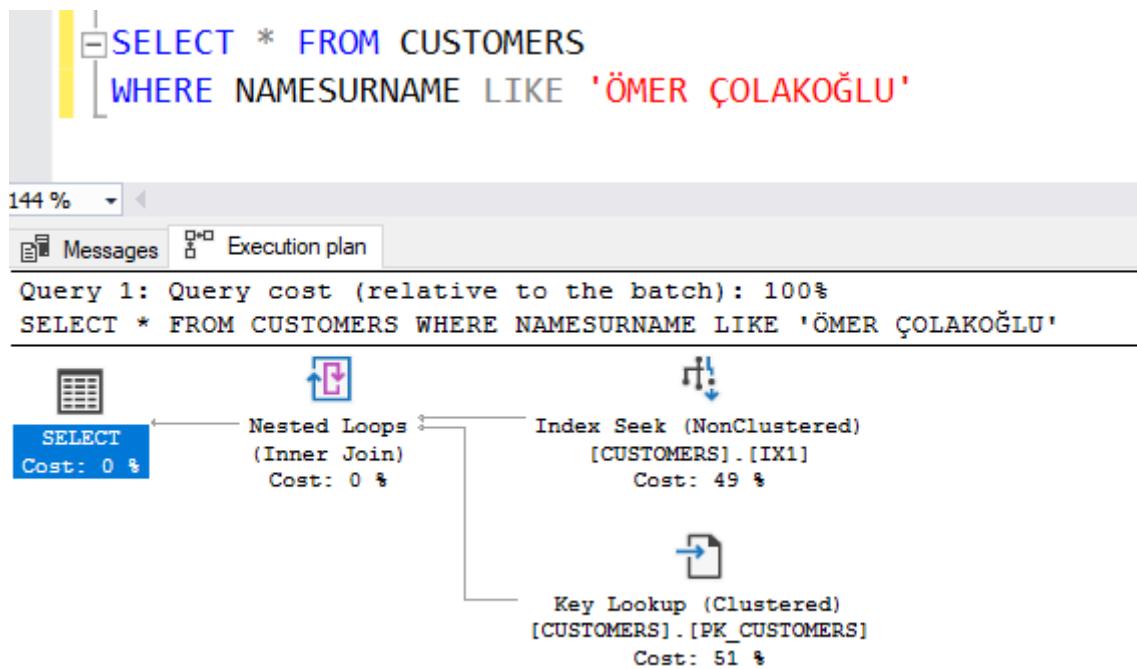
Index Uygulama

Eğer tabloda primary key yoksa Table Scan yöntemi ile arama yapar, bu en kötü yöntemdir.

Sorgu sürelerini hızlandırmak için tablomuza primary key ve index ekleyip sıralamayı ona göre yaptırımlıyız.



Non-Clustered index ekleyerek sorgu işlemini binlerce kat hızlandırdık.



Buradaki INDEX SEEK işlemi ile ismi Ömer Çolakoğlu olan kişiyi buldu, sonrasında ise o kişinin tüm bilgilerini getirmek için ID numarasını kullanarak bir de KEY LOOKUP işlemi yaptı.

Bu işlemi de daha hızlı yapmanın yolu INCLUDED COLUMN kullanmaktadır.

INCLUDED COLUMN

Index Properties ekranından Included Columns sekmesinde tüm kolonları ekledikten sonra sorguyu tekrar çalıştırıralım.

```

index.sql - recepa...\\recepaydogdu (54)* 143 %
SET STATISTICS IO ON
SELECT * FROM CUSTOMERS
WHERE NAMESURNAME LIKE 'ÖMER ÇOLAKOĞLU'

(1 row affected)
Table 'CUSTOMERS'. Scan count 1, logical reads 4, physical reads 0, page server
Completion time: 2020-07-30T21:40:26.1593489+03:00
| 

```

Sadece 4 adımda sorgunun sonucunu buldu.

Ve sadece Index Seek işlemini gerçekleştirdi. Key Lookup ile tekrar aramasına gerek kalmadı. Çünkü tüm istediklerimiz tek bir yerde.

Messages Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM CUSTOMERS WHERE NAMESURNAME LIKE 'ÖMER ÇOLAKOĞLU'
```

```
SELECT [CUSTOMERS].[IX1]
Cost: 0 %
Cost: 100 %
```

Ancak bu işlemlerin bize bir maliyeti var.

Index oluşturmadan önce tablomuzun diskte harcadığı alana bakalım:

SP_SPACEUSED 'CUSTOMERS'

143 %

	name	rows	reserved	data	index_size	unused
1	CUSTOMERS	2511833	430888 KB	429712 KB	1096 KB	80 KB

Gördüğümüz üzere reserved alanı 430888 kb ve index_size boyutu 1096 kb

Index oluşturuktan sonra kapladığı alan ise;

SP_SPACEUSED 'CUSTOMERS' --index olusturulunca

143 %

	name	rows	reserved	data	index_size	unused
1	CUSTOMERS	2511833	500528 KB	429712 KB	70320 KB	496 KB

Bir de INCLUDED COLUMN işlemi sonrasında diskte kapladığı alana bakalım;

SP_SPACEUSED 'CUSTOMERS' --included column sonrasında

143 %

	name	rows	reserved	data	index_size	unused
1	CUSTOMERS	2511833	674864 KB	429712 KB	244520 KB	632 KB

Birden Fazla Index Oluşturma

The screenshot shows the SQL Server Management Studio interface with the following content:

Execution plan:

Query 1: Query cost (relative to the batch): 50%
SELECT * FROM CUSTOMERS WHERE NAMESURNAME='ÖMER ÇOLAKOĞLU'

Execution Plan Details:

- Icon: Table
- Icon: Clustered Index Seek (Clustered)
- Text: [CUSTOMERS].[PK_CUSTOMERS]
- Text: Cost: 100 %
- Text: SELECT Cost: 0 %

Query 2: Query cost (relative to the batch): 50%
SELECT * FROM CUSTOMERS WHERE ID=168483

Execution Plan Details:

- Icon: Table
- Icon: NonClustered Index Seek (NonClustered)
- Text: [CUSTOMERS].[IX1]
- Text: Cost: 100 %
- Text: SELECT Cost: 0 %

SET STATISTICS IO ON

SELECT * FROM CUSTOMERS

WHERE TELNR=03423344747

143 %

Results Messages

```
(1 row affected)
Table 'CUSTOMERS'. Scan count 9, logical reads 32475, ph
```

Completion time: 2020-07-30T22:06:07.1639855+03:00

Önceki örneklerde yaptığımız işlemlerde ID ve NAMESURNAME alanlarına göre arama yapmıştık. TELNR'ye göre arama yaptığımızda ise buna özel bir index olmadığı için işlem uzun sürdü ve 32475 işlem sonucunda gerçekleşti.

```
CREATE NONCLUSTERED INDEX IX2
ON CUSTOMERS (TELNR)
```

```

SELECT * FROM CUSTOMERS
WHERE TELNR='03423344747'

```

143 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT * FROM CUSTOMERS WHERE TELNR='03423344747'

```

graph TD
    A[SELECT  
Cost: 0 %] --> B[Nested Loops  
(Inner Join)  
Cost: 0 %]
    B --> C[Index Seek  
[CUSTOMERS].[IX2]  
Cost: 50 %]
    C --> D[Key Lookup  
[CUSTOMERS].[PK_CUSTOMERS]  
Cost: 50 %]

```

```

SELECT * FROM CUSTOMERS
WHERE BIRTHDATE='1964-03-30'

```

143 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT * FROM CUSTOMERS WHERE BIRTHDATE='1964-03-30'

Missing Index (Impact 99.541): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[CUSTOMERS] ([BIRTHDATE])

```

graph LR
    A[SELECT  
Cost: 0 %] --> B[Parallelism  
(Gather Streams)]
    B --> C[Index Scan  
[CUSTOMERS].[IX1]  
Cost: 1 %]

```

BIRTHDATE alanına göre soru yaptığımızda ise bize 99.541 kat daha hızlı işlem yapabileceğim index önerisinde bulundu.

```

CREATE NONCLUSTERED INDEX IX3
ON [dbo].[CUSTOMERS] ([BIRTHDATE])
INCLUDE ([NAMESURNAME],[GENDER],[CITY],[TOWN],[TELNR],[NAME_],[SURNAME],[TCNO])

```

Önerdiği index'i ekledik.

```

SELECT * FROM CUSTOMERS
WHERE BIRTHDATE='1964-03-30'

```

143 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT * FROM CUSTOMERS WHERE BIRTHDATE='1964-03-

```

graph LR
    A[SELECT  
Cost: 0 %] --> B[Index Seek  
[CUSTOMERS].[IX3]  
Cost: 100 %]

```

Unique Index

```
SELECT * FROM CUSTOMERS  
WHERE NAMESURNAME='ÖMER ÇOLAKOĞLU'
```

Biz SQL Server'a bu soruyu gönderdiğimizde SQL Server istediğimiz satırı bulsa bile, tekrar varsa diye sonraki satırları da aramaya devam eder. Oysa ki biz SQL Server'a aradığımızın unique bir değer olduğunu belirtseydik sorgu işlemi çok daha hızlı şekilde bitecekti.

Örneğin TC kimlik numarası hiçbir zaman tekrar etmez.

```
SELECT * FROM CUSTOMERS  
WHERE TCNO='10019865823'
```

143 %

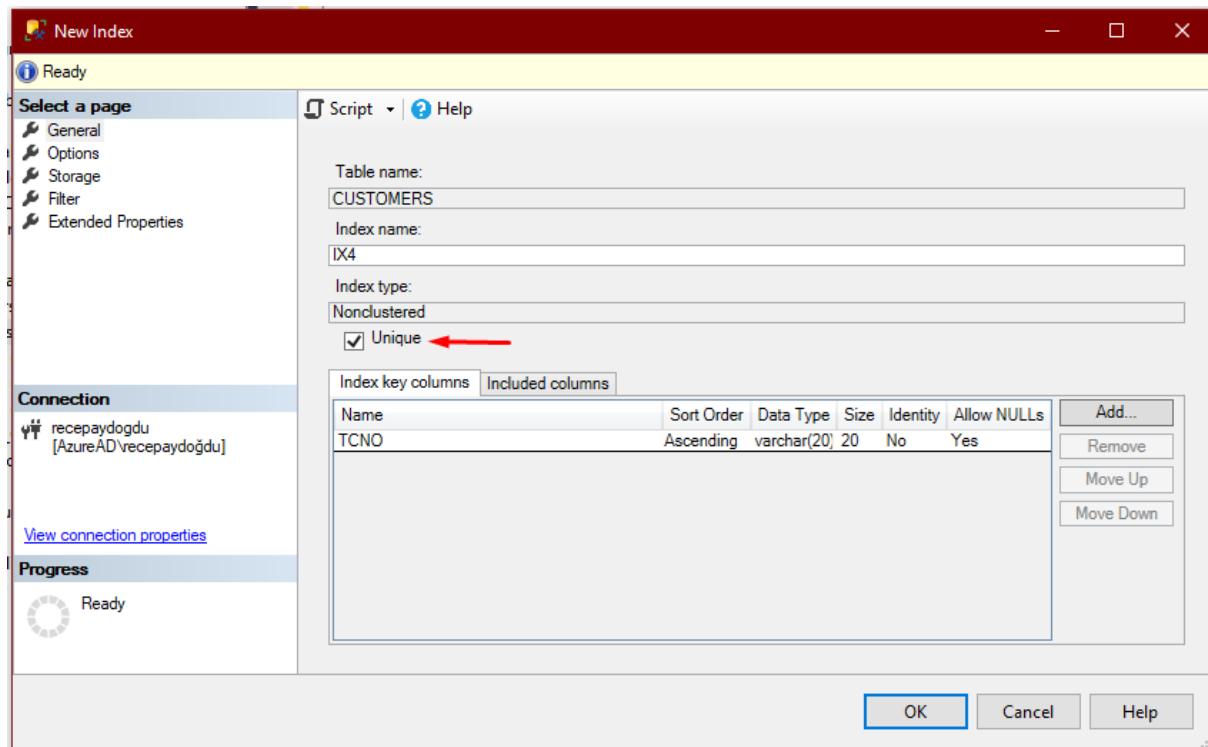
Messages Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT * FROM CUSTOMERS WHERE TCNO='10019865823'
Missing Index (Impact 99.8606): CREATE NONCLUSTERED INDEX

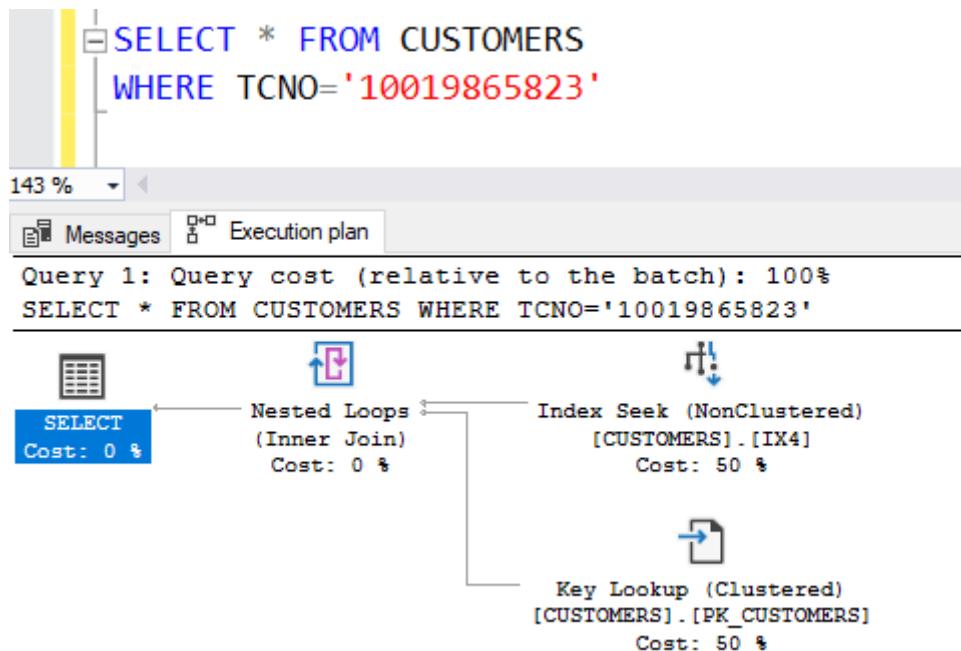
Diagram of the execution plan:

```
graph LR; A[SELECT Cost: 0 %] --> B[Parallelism (Gather Streams) Cost: 1 %]; B --> C[Index Scan (NonClustered) [CUSTOMERS].[IX3] Cost: 99 %]
```

Burada alakasız bir index kullanıp soruyu çalıştırdı. Yeni bir index oluşturalım.



Buradaki kritik nokta Unique kutucuğunu seçiyor olmamız.



Bu sayede unique olması gereken verilerin tekrar etmesinin de önüne geçmiş oluruz.

The screenshot shows a series of statements: SET STATISTICS IO ON, SELECT * FROM CUSTOMERS WHERE ID=35578, and UPDATE CUSTOMERS SET TCNO='10019865823' WHERE ID=35578. The update statement is highlighted in pink. In the message pane, an error message is displayed: Msg 2601, Level 14, State 1, Line 7: Cannot insert duplicate key row in object 'dbo.CUSTOMERS' with unique index 'IX4'. The statement has been terminated. The completion time is listed as 2020-04-08T02:53:01.1924839+03:00.

```
SET STATISTICS IO ON
SELECT * FROM CUSTOMERS WHERE
ID=35578

WHERE TCNO='10019865823'

UPDATE CUSTOMERS SET TCNO='10019865823' WHERE ID=35578

Msg 2601, Level 14, State 1, Line 7
Cannot insert duplicate key row in object 'dbo.CUSTOMERS' with unique index 'IX4'.
The statement has been terminated.

Completion time: 2020-04-08T02:53:01.1924839+03:00
```

Var olan bir TCNO'yu başka bir satıra update etmek istediğimizde bu hataya karşılaştık.

Index Bozulmaları (Fragmantation)

Gerçek hayat verilerinde veritabanına dinamik olarak veriler eklendikçe index bozulmaları yaşanabiliyor.

INDEX BOZULMALARI

DATA		INDEX	
ID	İSİM	ID	İSİM
1	OSMAN	2	AHMET
2	AHMET	11	ALİ
3	MUSTAFA	19	AYTEN
4	BEKİR	21	BANU
5	ZEYNEP	4	BEKİR
6	ELA	20	CANAN
7	CENGİZ	7	CENGİZ
8	MUSTAFA	6	ELA
9	EMİR	9	EMİR
10	ÖMER	12	GAMZE
11	ALİ	24	HANDAN
12	GAMZE	17	İSMAİL
13	YUSUF	23	LALE
14	MURAT	14	MURAT
15	ÖNDER	3	MUSTAFA
16	SEDAT	8	MUSTAFA
17	İSMAİL	1	OSMAN
18	SAMET	10	ÖMER
19	AYTEN	15	ÖNDER
20	CANAN	18	SAMET
21	BANU	16	SEDAT
22	ZUHAL	13	YUSUF
23	LALE	5	ZEYNEP
24	HANDAN	22	ZUHAL

btkakademi btkakademi

Veritabanının en küçük yapısı page'ler idi. Bir veritabanı 8 kb'lık pagelerden oluşur. Örnek göstermek adına 1 page'e 8 kayıt gelecek şekilde gösterilmiştir ancak bu gerçekte değişebilir.

DATA PAGE LER

PAGE 1		PAGE 2		PAGE 3	
ID	İSİM	ID	İSİM	ID	İSİM
1	OSMAN	9	EMİR	17	İSMAİL
2	AHMET	10	ÖMER	18	SAMET
3	MUSTAFA	11	ALİ	19	AYTEN
4	BEKİR	12	GAMZE	20	CANAN
5	ZEYNEP	13	YUSUF	21	BANU
6	ELA	14	MURAT	22	ZUHAL
7	CENGİZ	15	ÖNDER	23	LALE
8	MUSTAFA	16	SEDAT	24	HANDAN

INDEX BOZULMALARI



Yeni eklenen kayıtlar index page'lerinde sona eklenir.

Indexler belirli zamanlarda yeni kayıtlar eklendikçe bozulur ve yeniden düzenlenmesi gereklidir.

Bu düzeltme işleminin adı **rebuild**'dir ve database maintenance planlarından birisidir. (Haftada bir, ayda bir düzenlenir.)

Bu index düzenleme işlemleri sistem kapalıken gerçekleşmesi gereklidir. 7/24 çalışan sistemlerde bu zor bir durumdur.

Bu durumla başa çıkmak için indexleri daha zor bozulur hale getirmemiz gerekiyor.



Örneğin index page'lerimizin her birinde bir miktar boş alan bırakıksak ve yeni eklenen veriler bu boş alanlara eklendiğinde index bozulma süremiz uzayacaktır.

PAGE 1

ID	İSİM
2	AHMET
11	ALİ
25	AYDIN
19	AYTEN
21	BANU
4	BEKİR

Yeni kayıt Aydın geldiğinde indexin sonuna değil Page 1'e eklenecek.

INDEX PAGE LER

PAGE 1

ID	İSİM
2	AHMET
11	ALİ
26	ARZU
25	AYDIN
19	AYTEN
21	BANU
4	BEKİR

PAGE 2

ID	İSİM
20	CANAN
7	CENGİZ
6	ELA
9	EMİR
12	GAMZE

PAGE 3

ID	İSİM
24	HANDAN
17	İSMAİL
27	KAAN
23	LALE
14	MURAT
3	MUSTAFA

PAGE 4

ID	İSİM
8	MUSTAFA
28	NAZIM
1	OSMAN
10	ÖMER
15	ÖNDER
18	SAMET

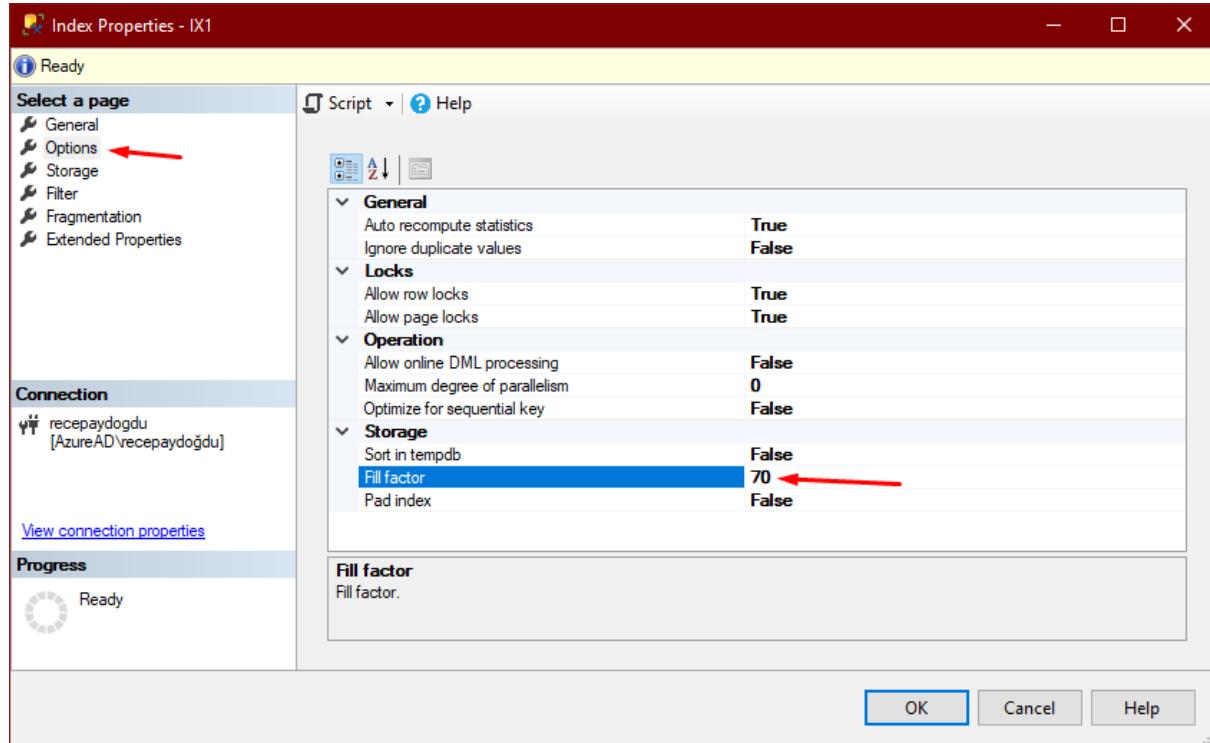
Burada turkuaz renkle işaretli alanlara Fill Factor adı verilir. Biz eğer indexi oluştururken ona belirli bir Fill Factor oranında olmasını söyleseysek indexlerimizin bozulma süresi uzayacaktır.

Index Fragmentation Uygulaması (Fill Factor)

Bozulan indexi düzeltmenin 2 yolu var. **Rebuild** ve **Reorganize**.

Rebuild daha iyi sonuç verir.

Bozulmanın önüne geçmek için ise **Fill Factor** kullanınız.



Fill Factor için 70 değer verdiğimizde index page'lerimizin %70'i dolu olacak şekilde düzenlenir.

Script ile oluşturulması ise aşağıdaki gibidir;

```
CREATE NONCLUSTERED INDEX [IX1] ON [dbo].[CUSTOMERS]
(
    [NAMESURNAME] ASC
)WITH (FILLFACTOR = 70)
```

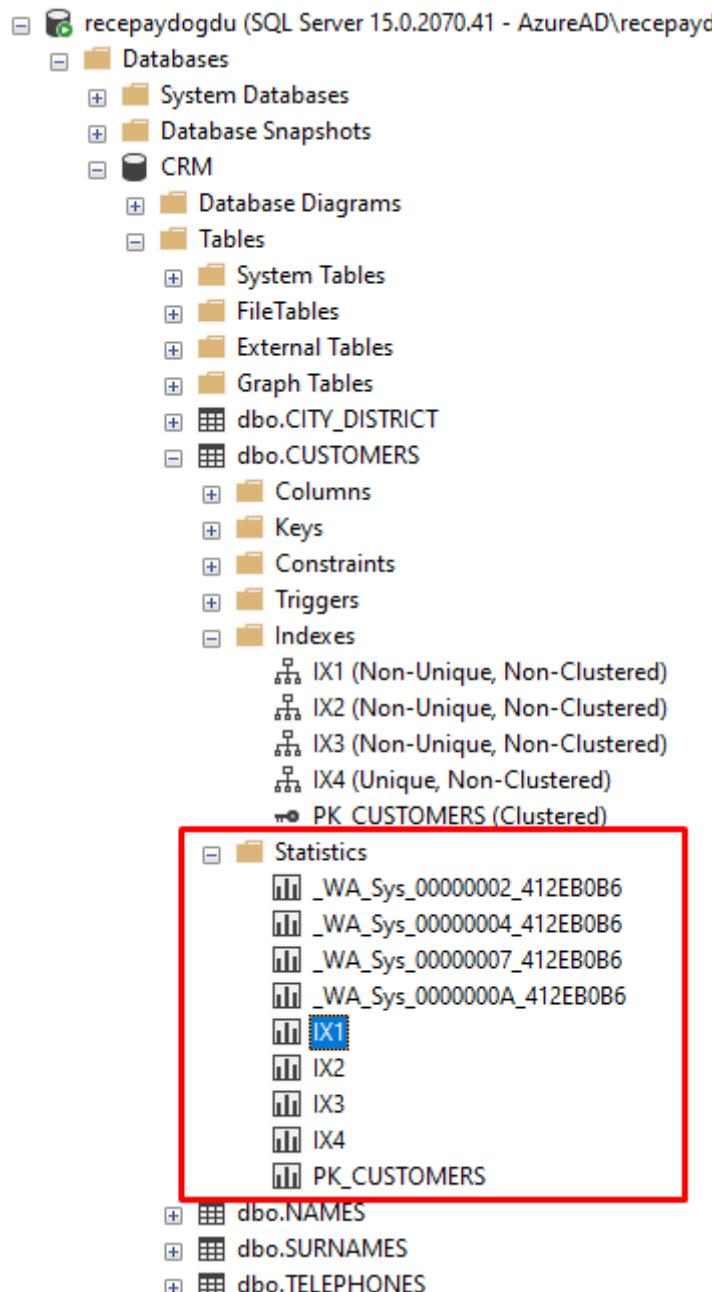
Her seferin Fill Factor ayarlamakla uğraşmamalı, server bazında default olarak Fill Factor değeri oluşturulmalıdır. Her index oluşturduğumuzda default Fill Factor değeri geçerli olsun dersek Server Properties sekmesinden Database Settings içerisinde default index fill factor değerini değiştirebiliriz.

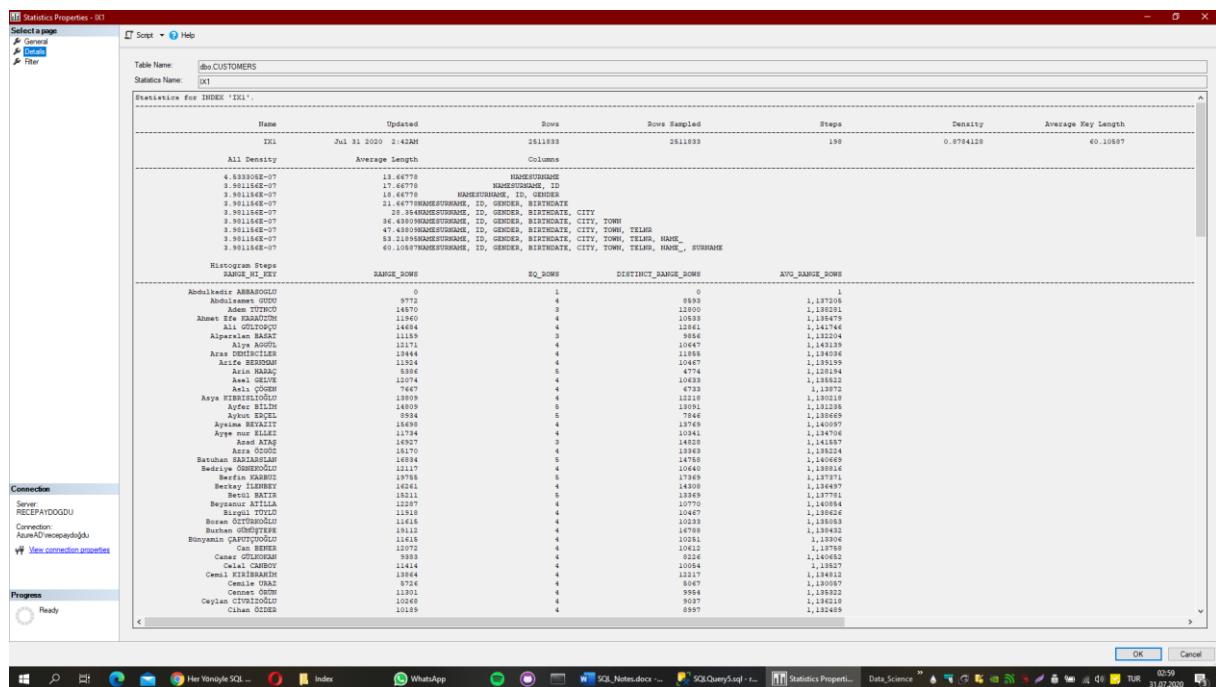
The screenshot shows two windows from SQL Server Management Studio (SSMS).
The top window is a context menu for a database named 'recepaydogdu'. The 'Properties' option is highlighted with a red arrow. The menu includes options like Connect..., Disconnect, Register..., New Query, Activity Monitor, Start, Stop, Pause, Resume, Restart, Policies, Facets, Start PowerShell, Azure Data Studio, Reports, Refresh, and Properties.
The bottom window is the 'Server Properties - recepaydogdu' dialog. The 'Select a page' sidebar has 'Database Settings' selected, indicated by a red arrow. The main pane shows 'Default index fill factor' set to 70, which is also highlighted with a red arrow. Other settings include 'Backup and restore' options (Wait indefinitely, Try once, Try for 0 minutes), 'Default backup media retention (in days)' set to 0, and 'Recovery' settings. The 'Connection' section shows the server is 'RECEPAYDOGDU' and the connection is 'AzureAD\recepaydogdu'. The 'Progress' section shows the status as 'Ready'. At the bottom are 'OK' and 'Cancel' buttons.

Bu işlemden sonra oluşturduğumuz her indexin Fill Factor değeri 70 olarak olacaktır.

Istatistikler

Bizim her indeximiz için Statistics bölümünde istatistikler oluşur.





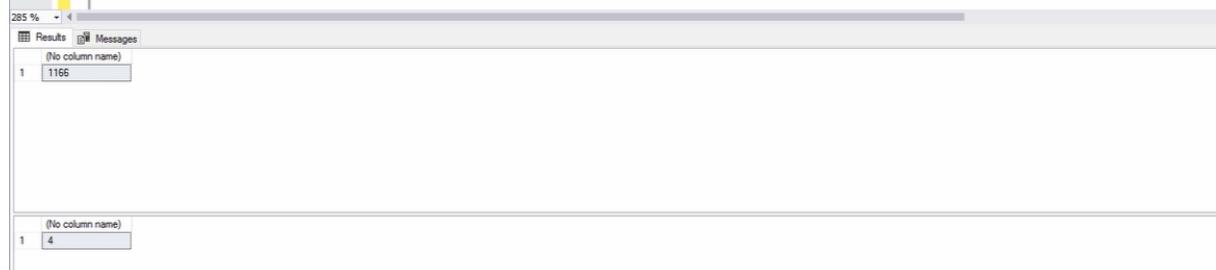
Örneğin IX1 indexinin istatistik detaylarına baktığımızda her bir değişkenin kaç kez var olduğu gibi değerler görürüz.

Bizim sorgularımızda SQL Server hangi index'i kullanacağına bu istatistik değerlerine göre karar verir.

Örneğin BIRTHDATE ve USERNAME alanlarına göre sorgulama yapacağımızı varsayıyalım;

```
SELECT * FROM CUSTOMERS
WHERE BIRTHDATE='1996-03-09'
AND NAMESURNAME='ADA SÖZÜER'

SELECT COUNT(*) FROM CUSTOMERS WHERE BIRTHDATE='1996-03-09'
SELECT COUNT(*) FROM CUSTOMERS WHERE NAMESURNAME='ADA SÖZÜER'
```



O doğum tarihinde doğan 1166 kişi var, ADA SÖZÜER isminde ise 4 kişi var. Biz yukarıdaki sorguyu yaptığımızda 4 seçenek arasından bulup getirmek daha hızlı olacağından SQL Server IX1 indexini çalıştırır.

Bu istatistik değerleri her index oluşturulduğunda güncellenir, ya da elle güncellememiz gereklidir. SQL Server'ımızın doğru performansta çalışması için düzenli olarak istatistiklerin güncellenmesi gereklidir.

ADA SÖZÜER isimli müşteriden yaklaşık 5000 adet eklendiğini düşünelim ve istatistiklerin güncellenmediğini düşünelim.

Biz aynı sorguyu tekrar yaptığımda istatistiklerimiz güncel olmadığından SQL Server yine IX1 index'ini kullanarak sorguyu gerçekleştirecektir.

IX1 indexi ile yapılan page okuma sayısını aşağıda görebilirsiniz. (22437)

WITH (INDEX=IX3) komutu ile sorguda kullanmak istediğimiz indexi belirttiğimizde bu sorguda okunan page sayısı 3593 olarak karşımıza çıktı.

```
SET STATISTICS IO ON
SELECT * FROM CUSTOMERS WITH (INDEX=IX3)
WHERE BIRTHDATE='1996-03-09'
AND NAMESURNAME='ADA SÖZÜER'
-- 22437 IX1
-- 3593
```

Istatistik Güncellemenin 3 Yolu Var

- 1- Index'i rebuild etmek veya reorganize etmek.
Bu adım sistem kapalı yapılması gerekiğinden her zaman mümkün olmayabilir.
- 2- SP_UPDATERSTATS komutu ile tüm tablolar için istatistikleri güncelleyebiliriz.
- 3- UPDATE STATISTICS CUSTOMERS komutu ile de belirttiğimiz tablo için istatistiklerimizi güncelleyebiliriz.

```
SP_UPDATERSTATS
UPDATE STATISTICS CUSTOMERS
```

Istatistiklerimiz güncellendikten sonra yukarıdaki sorguyu çalıştırduğumda ise okunan page sayısı 35 oldu.

Istatistiklerimin her gün güncellenmesi gereklidir!!

View

Bir yahut birden fazla tabloyu join ile birleştirip, where ile koşul belirttikten sonra içerisinde group by deyiminin kullanılabildeği sanal tabloya view denir. View in amaçlarından bir tanesi ise birden fazla tablodan çekilen verilerin tek tablo içerisinde toplanması ve bu sanal tablo içerisinde gerekli filtreler verilerek sorgu ile raporlama yapılabilmesidir, bunlar sql view avantajları arasında sayılabilir.

```
SELECT
    U.USERNAME_ KULLANICIADI, U.NAMESURNAME ADSOYAD, C.COUNTRY ULKE, CT.CITY SEHIR,
    T.TOWN ILCE, O.ID, ITM.ITEMCODE, ITM.ITEMNAME, ITM.BRAND MARKA, OD.AMOUNT MIKTAR,
    OD.UNITPRICE BIRIMFIYAT, OD.LINETOTAL SATIRTOPLAMI
FROM
    ORDERDETAILS OD
JOIN ORDERS O ON OD.ORDERID=O.ID
JOIN ITEMS ITM ON ITM.ID=OD.ITEMID
JOIN USERS U ON U.ID=O.USERID
JOIN ADDRESS A ON A.ID=O.ADDRESSID
JOIN COUNTRIES C ON C.ID=A.COUNTRYID
JOIN CITIES CT ON CT.ID=A.CITYID
JOIN TOWNS T ON T.ID=A.TOWNID
```

143 % -

Results Messages

KULLANICIADI	ADSOYAD	ULKE	SEHIR	ILCE	ID	ITEMCODE	ITEMNAME	MARKA	MIKTAR	BIRIMFIYAT	SATIRTOPLAMI
5 E_TOPGVLER	Emine TOPGÜLER	TÜRKİYE	GAZİANTEP	İSLAHİYE	23	8290	KITAP HİC KİMSE SIRADAN DEĞİLDİR	KİTAPLAR	2	11.9040	23.8080
6 G_TUGUTLU	Gülcen TUĞUTLU	TÜRKİYE	BALIKESİR	BALIKESİR MERKEZ	38	13491	JİBER 3 LU PİRİNA ÇOCUK SLİP 753	JİBER	8	9.2384	73.9072
7 H_GÖZTEPE	Hira GÖZTEPE	TÜRKİYE	İZMİR	BÖRNOVA	55	29152	OYUNCAK KRT. 2-Lİ SILAHLI POLIS SETİ	OYUNCAK	7	12.3731	86.6117
8 S_TORT	Sudetur TORT	TÜRKİYE	UŞAK	EŞME	56	41469	SOLEN BISCOLATA STARZ BIT.CIK.50 GR'24*	SOLEN	4	1.2564	5.0255
9 N_CUHA	Nuriye CUHA	TÜRKİYE	SAKARYA	SERDİVAN	73	37736	KENT BORINGER KARAMEL TOPING SOS 750GR '12*	KENT	4	14.3426	57.3702
10 N_CUHA	Nuriye CUHA	TÜRKİYE	SAKARYA	SERDİVAN	73	11148	VENUS MAKINA+3 BİCAK	VENUS	7	27.0497	189.3477
11 A_ENGVZEL	Aya ENGVZEL	TÜRKİYE	VAN	VAN MERKEZ	88	41967	ICIM PEYNİR 250 GR SUZME "12"	ICIM	2	7.5904	15.1808
12 C_ZAIM	Canan ZAIM	TÜRKİYE	SİVAS	SİVAS MERKEZ	91	36133	PATIS-TODIS-YUMIS-HOPIS BOYAMA	KİTAPLAR	7	2.3733	16.6134
13 B_ALPAR	Baş Alpar	TÜRKİYE	MARDİN	MARDİN MERKEZ	105	39679	DERBY TRAS KOPUGU BOSS 200ML	DERBY	8	3.8405	30.7237
14 B_ALPAR	Baş Alpar	TÜRKİYE	MARDİN	MARDİN MERKEZ	105	10175	KARTELA OK YAY SETİ	OYUNCAK	3	28.1362	84.4085
15 B_ALPAR	Baş Alpar	TÜRKİYE	MARDİN	MARDİN MERKEZ	105	14954	MOLPED MAXI DRY&PLS SW 10'LU"24"	MOLPED	9	7.8004	70.2037
16 N_BALK	Naz BALK	TÜRKİYE	KAYSERİ	BÜNYAN	106	25724	KENT TAVUK BULYON 20 GR '12*	KENT	4	0.8386	3.3545
17 S_KARADAG	Simge KARADAG	TÜRKİYE	AFYONKARAHİSAR	BOLVADIN	123	10133	NT EL KREMI PARFÜMSÜZ	NEUTROGENA	9	20.2665	182.3986
18 S_KARADAG	Simge KARADAG	TÜRKİYE	AFYONKARAHİSAR	BOLVADIN	123	8018	OLUM PEYGAMBERİ	KİTAPLAR	7	12.6572	88.6004
19 M_TALU	Münnevver TALU	TÜRKİYE	DÜZCE	DÜZCE MERKEZ	124	37621	PRT.LUX YUMURTA SAKLAMA KABI	LUX	10	2.9587	29.5875
20 G_MEHEL	Gülen MEHEL	TÜRKİYE	BURSA	MUDANYA	138	27760	NERGIS PİRİNÇ PILAVLIK 5 KG "4"	NERGIS	7	56.9209	398.4464

Burada oluşturduğumuz gibi joinler ve alias'lar içeren karmaşık bir sorguyu bazen tablo olarak kullanmamız gerekebilir. Bu yapıya View denir.

```
CREATE VIEW VWORDERS
AS
SELECT
    U.USERNAME_ KULLANICIADI, U.NAMESURNAME ADSOYAD, C.COUNTRY ULKE, CT.CITY SEHIR,
    T.TOWN ILCE, O.ID, ITM.ITEMCODE, ITM.ITEMNAME, ITM.BRAND MARKA, OD.AMOUNT MIKTAR,
    OD.UNITPRICE BIRIMFIYAT, OD.LINETOTAL SATIRTOPLAMI
FROM
    ORDERDETAILS OD
JOIN ORDERS O ON OD.ORDERID=O.ID
JOIN ITEMS ITM ON ITM.ID=OD.ITEMID
JOIN USERS U ON U.ID=O.USERID
JOIN ADDRESS A ON A.ID=O.ADDRESSID
JOIN COUNTRIES C ON C.ID=A.COUNTRYID
JOIN CITIES CT ON CT.ID=A.CITYID
JOIN TOWNS T ON T.ID=A.TOWNID
```

Sorgumuzun başına bu kod satırlarını ekleyerek VWORDERS adında View oluşturduk.

```

SELECT * FROM VWORDERS --View'lar bu sekilde cagirilir.
WHERE KULLANICIADI='M_ALI' --View'larda kolon isimlerini kullanabiliriz.

```

143 %

Results Messages

KULLANICIADI	ADSOYAD	ULKE	SEHIR	ILCE	ID	ITEMCODE	ITEMNAME	MARKA	MIKTAR	BIRIMFIYAT	SATIRTOPLAMI
1 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	40010	14720	BAY.SEK.KENT 1 KG ELEGAN CILEK '1"	KENT	9	22.2442	200.1974
2 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	63082	8832	ETI TADINDA 140 GR PEYNIRLI'16"	ETI	8	1.5422	12.3376
3 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	70764	21645	RINSO MATIK 10 KG KIR BAHCESI'1"	RINSO	1	65.7913	65.7913
4 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	45399	6791	FERSAN LIMON SUyu 330 ML '24"	FERSAN	6	2.9074	17.4445
5 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	70764	33637	ETI HOSBES VANILYALI 160 GR '19"	ETI	2	1.7876	3.5751
6 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	22661	18880	KENT JELIBON 80 GR TOPIK '24"	KENT	7	4.6222	32.3552
7 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	45399	42796	TIGRA SERIT DAKSIL	KIRTASIYELER	9	1.3262	11.9360
8 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	22661	40650	GEZER TERLIK FILET TRASLI 6455	GEZER	7	7.5183	52.6282
9 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	22661	1284	GOLBASI MANDALINA	MANDALINA	6	1.1407	6.8442
10 M_ALI	Muhammed Ali YALGIN	TURKIYE	BURSA	OSMANGAZI	22661	15191	BORA 0325 YENI PASTA VE KEK KABI	BORA	2	11.2556	22.5112

User Defined Functions

```

CREATE FUNCTION DBO.TOPLA(@SAYI1 AS INT, @SAYI2 AS INT)
RETURNS INT
AS
BEGIN
    DECLARE @SONUC AS INT
    SET @SONUC=@SAYI1+@SAYI2
    RETURN @SONUC
END

SELECT DBO.TOPLA(8,10) --Fonksiyon calistirma

```

143 %

Results Messages

(No column name)
1 18

```

--Dogum tarihini alip yasini bulan fonksiyon
CREATE FUNCTION DBO.FINDAGE(@TARIH AS DATE)
RETURNS INT
AS
BEGIN
    DECLARE @YAS AS INT
    SET @YAS=DATEDIFF(YEAR,@TARIH,GETDATE())
    RETURN @YAS
END

SELECT DBO.FINDAGE('1999-11-13') AS YAS
--Bu fonksiyonu db üzerinde kullanalım
SELECT TOP 1000 USERNAME_, DBO.FINDAGE(BIRTHDATE) AS AGE
FROM USERS

```

143 %

	USERNAME_	AGE
1	N_OZSIMITCI	69
2	E_SELIM	27
3	S_VLGEN	80
4	C_BORKLV	74
5	E_IBUKVRTVNCV	35
6	H_VREGIL	21
7	A_SUYUR	56
8	O_KIRIT	48
9	S_TVKEZIM	37
10	B_PIRINCAL	57

Gün Adını Getiren Fonksiyon

```
CREATE FUNCTION DBO.DAYOFWEEK_(@DATE AS DATETIME)
RETURNS VARCHAR(10)
AS
BEGIN
DECLARE @RESULT AS VARCHAR(10)

IF DATEPART(DW,@DATE)=2 SET @RESULT='1.PZT'
IF DATEPART(DW,@DATE)=3 SET @RESULT='2.SAL'
IF DATEPART(DW,@DATE)=4 SET @RESULT='3.CAR'
IF DATEPART(DW,@DATE)=5 SET @RESULT='4.PER'
IF DATEPART(DW,@DATE)=6 SET @RESULT='5.CUM'
IF DATEPART(DW,@DATE)=7 SET @RESULT='6.CMT'
IF DATEPART(DW,@DATE)=1 SET @RESULT='7.PAZ'

RETURN @RESULT
END
```

```
--Urun Analizi
SELECT
ITM.ID,ITM.ITEMCODE URUNKODU, ITM.ITEMNAME URUNADI, MIN(OD.UNITPRICE) ENDUSUKFIYAT,
MAX(OD.UNITPRICE) ENYUKSEKFIYAT, AVG(OD.UNITPRICE) ORTALAMAFIYAT,
SUM(OD.LINETOTAL) TOPLAMSATISTUTAR,SUM(OD.AMOUNT) AS TOPLAMMIKTAR
FROM ORDERDETAILS OD
JOIN ITEMS ITM ON ITM.ID=OD.ITEMID
JOIN ORDERS O ON O.ID=OD.ORDERID
GROUP BY ITM.ID,ITM.ITEMCODE, ITM.ITEMNAME

--Bu soruyu fonksiyon haline getirelim
CREATE FUNCTION DBO.GETITEM_MIN_PRICE(@ITEMID AS INT)
RETURNS FLOAT
AS
BEGIN
DECLARE @RESULT AS FLOAT
SELECT @RESULT=MIN(UNITPRICE) FROM ORDERDETAILS OD WHERE OD.ITEMID=@ITEMID
RETURN @RESULT
END

SELECT DBO.GETITEM_MIN_PRICE(152)
```

143 %	
	Results
1	(No column name) 8,9409

```

SELECT DBO.GETITEM_MIN_PRICE(152)
SELECT DBO.GETITEM_MAX_PRICE(152)
SELECT DBO.GETITEM_AVG_PRICE(152)
SELECT DBO.GETITEM_TOTAL_SALE(152)

```

143 %

Results Messages

	(No column name)
1	8,9409
	(No column name)
1	10,4709
	(No column name)
1	9,711755
	(No column name)
1	943,4082

Fonksiyonların en büyük dezavantajı satır satır işlem yaptıkları için çok yavaş çalışmalarıdır.

TOP 100 sorgumuz bile 4 saniye sürdü.

--Simdi yukarıdaki sorguyu fonksiyonlar ile yapalım

```

SELECT TOP 100
    ITM.ID, ITM.ITEMCODE, ITM.ITEMNAME,
    DBO.GETITEM_MIN_PRICE(ITM.ID), DBO.GETITEM_MAX_PRICE(ITM.ID),
    DBO.GETITEM_AVG_PRICE(ITM.ID), DBO.GETITEM_TOTAL_SALE(ITM.ID)
    FROM ORDERDETAILS OD
    JOIN ITEMS ITM ON ITM.ID=OD.ITEMID
    JOIN ORDERS O ON O.ID=OD.ORDERID
    GROUP BY ITM.ID,ITM.ITEMCODE,ITM.ITEMNAME
    --FONKSIYONLARIN EN BUYUK DEZAVANTAJI COK YAVAS CALISMALARIDIR!

```

143 %

Results Messages

ID	ITEMCODE	ITEMNAME	(No column name)	(No column name)	(No column name)	(No column name)
1	660	DOKME KIMYON 10 KG.	12,8775	15,2828	13,963925	1886,0103
2	1673	SILA TRUVA 1 NO CAYDANLIK	56,3285	65,9262	61,324539	6996,0234
3	675	OB.D.FIR.NATURAL FRESH MED 40(2+1)*96*	31,3143	37,2573	34,033476	4013,3436
4	4217	BREF SOLID STICK PINE 40 GR*24*	3,0571	3,5209	3,256324	445,8179
5	19206	OBA MAK.TEL SEHRIYE 500 GR *20*	1,161	1,3777	1,266075	176,2005
6	2305	OYUNCAK METAL ARAB.SETI 5 PCS(DIE CAST)	20,3207	24,201	21,872768	3192,3888
7	14011	YOBO F405 CICITLI KORUKLU DOSYA	7,0286	8,2923	7,691725	1152,2585
8	1375	DALIN SAMP.125 ML KUCUK *48*	7,1884	8,5097	7,760443	726,4685
9	15401	OLUMU OZLEMEN ASKI ANLAYAMAZ	3,8077	4,3794	4,135609	484,083
10	21633	SEK M.SUYU 1 LT KARISIK *12*	2,9201	3,4678	3,21493	342,8148

Bu kadar yavaş olmasının sebebi index olmamasıdır.

ITEMID için index oluşturursak sorgumuz hızlanacaktır.

Included Columns'lara ise AMOUNT, UNITPRICE ve LINETOTAL sütunlarını ekleyelim.

İstatistikleri de güncelleyelim.

Bu işlemler sonucunda 3 4 dakika süren tüm verileri çağırma süresi 1 saniyeye düştü.

İki Parametreli Fonksiyonlar

```
--Az önce yazdigimiz fonksiyonlari tek bir fonksiyon haline getirelim.  
CREATE FUNCTION DBO.GETITEM_PRICE(@ITEMID AS INT, @PRICETYPE AS VARCHAR(10))  
RETURNS FLOAT  
AS  
BEGIN  
    DECLARE @RESULT AS FLOAT  
  
    IF @PRICETYPE='MIN'  
    BEGIN  
        SELECT @RESULT=MIN(UNITPRICE) FROM ORDERDETAILS OD WHERE OD.ITEMID=@ITEMID  
    END  
  
    IF @PRICETYPE='MAX'  
    BEGIN  
        SELECT @RESULT=MAX(UNITPRICE) FROM ORDERDETAILS OD WHERE OD.ITEMID=@ITEMID  
    END  
  
    IF @PRICETYPE='AVG'  
    BEGIN  
        SELECT @RESULT=AVG(UNITPRICE) FROM ORDERDETAILS OD WHERE OD.ITEMID=@ITEMID  
    END  
  
    IF @PRICETYPE='TOTAL'  
    BEGIN  
        SELECT @RESULT=SUM(LINETOTAL) FROM ORDERDETAILS OD WHERE OD.ITEMID=@ITEMID  
    END  
  
    RETURN @RESULT  
END  
  
-----  
SELECT  
    ITM.ID, ITM.ITEMCODE, ITM.ITEMNAME,  
    DBO.GETITEM_PRICE(ITM.ID, 'MIN') ENDUSUKFIYAT, DBO.GETITEM_PRICE(ITM.ID, 'MAX') ENYUKSEKFIYAT,  
    DBO.GETITEM_PRICE(ITM.ID, 'AVG') ORTALAMAFIYAT, DBO.GETITEM_PRICE(ITM.ID, 'TOTAL') TOPLAMSATISTUTARI  
FROM ORDERDETAILS OD  
JOIN ITEMS ITM ON ITM.ID=OD.ITEMID  
JOIN ORDERS O ON O.ID=OD.ORDERID  
GROUP BY ITM.ID, ITM.ITEMCODE, ITM.ITEMNAME
```

ID	ITEMCODE	ITEMNAME	ENDUSUKFIYAT	ENYUKSEKFIYAT	ORTALAMAFIYAT	TOPLAMSATISTUTARI
1	5	PIL KODAK XTRA HEAVY 9 V	5,1725	6,1203	5,726971	545,8972
2	6	PIL KODAK AA'2 MAX ALKALIN KALEM	8,3444	9,6654	9,162638	635,6018
3	9	PILLI SESLİ UCAK	53,3183	63,6549	56,774324	7926,3332
4	11	VAKKUMLU TASİYICI TIR	21,0391	24,8522	22,986933	1864,0023
5	15	OYUNCAK KUT.4LU METAL CEK BIRAK	31,5345	37,0897	34,563779	4984,6109
6	16	OYUNCAK BEZ BEBEK	19,579	23,4404	22,01575	2256,9652
7	17	OFICA SERİT DAKSIL	1,028	1,199	1,121944	184,3315
8	18	OFFICA MAKET BİCAGI UCU	1,6888	2,0129	1,85882	265,2704
9	19	OFFICA FH4550 HESAP MAKİNASI	32,0204	35,7313	33,752707	2649,3165
10	21	PRT.UDAG WA0030 SAKLAMA KABI	15,7903	18,6048	16,025744	1486,5168

Table Valued Functions

Tek seferde birden fazla veri döndüren fonksiyonlar table valued functions olarak isimlendirilir.

```
--Table Valued Funtions

CREATE FUNCTION DBO.GETITEM_INFO(@ITEMID INT)
RETURNS TABLE
AS
RETURN
(
    --BURAYA YAZACAGIMIZ SQL SORGUSUNU TABLO OLARAK DONDURECEKTIR.
)

--Table Valued Funtions

CREATE FUNCTION DBO.GETITEM_INFO(@ITEMID INT)
RETURNS TABLE
AS
RETURN
(
    --BURAYA YAZACAGIMIZ SQL SORGUSUNU TABLO OLARAK DONDURECEKTIR.
    SELECT
        MIN(UNITPRICE) AS MINPRICE,
        MAX(UNITPRICE) AS MAXPRICE,
        AVG(UNITPRICE) AS AVGPRICE,
        SUM(LINETOTAL) AS TOTALSALE,
        SUM(AMOUNT) AS TOTALAMOUNT
    FROM ORDERDETAILS
    WHERE ITEMID=@ITEMID
)
```

143 %

Results Messages

	MINPRICE	MAXPRICE	AVGPRICE	TOTALSALE	TOTALAMOUNT
1	8.9409	10.4709	9.711755	943.4082	97

```

--Bu fonksiyonu artık bizim tablomuzla beraber kullanabiliriz.
--Bu işlemi join yerine CROSS APPLY kullanarak yaparız.
SELECT
    ITM.ID, ITM.ITEMCODE, ITM.ITEMNAME,
    ITEMINFO.MINPRICE AS ENDUSUKFIYAT,
    ITEMINFO.MAXPRICE AS ENYUKSEKFIYAT,
    ITEMINFO.AVGPRICE AS ORTALAMAFIYAT,
    ITEMINFO.TOTALAMOUNT AS SATISMIKTARI,
    ITEMINFO.TOTALSALE AS TOPLAMSATISTUTARI
FROM ITEMS ITM
CROSS APPLY DBO.GETITEM_INFO(ITM.ID) AS ITEMINFO

```

143 %

Results Messages

ID	ITEMCODE	ITEMNAME	ENDUSUKFIYAT	ENYUKSEKFIYAT	ORTALAMAFIYAT	SATISMIKTARI	TOPLAMSATISTUTARI
1	1	PIL KODAK XTRA HEAVY 9 V	5.1725	6.1203	5.726971	96	545.8972
2	2	PIL KODAK AA'2 MAX ALKALIN KALEM	8.3444	9.6654	9.162638	69	635.6018
3	3	PILLI SESLİ UCAK	53.3183	63.6549	56.774324	140	7926.3332
4	4	VAKKUMLU TASİYICI TIR	21.0391	24.8522	22.986933	81	1864.0023
5	5	OYUNCAK KUT.4LU METAL CEK BIRAK	31.5345	37.0897	34.563779	143	4984.6109
6	6	OYUNCAK BEZ BEBEK	19.5790	23.4404	22.015750	102	2256.9652
7	7	OFICA SERİT DAKSIL	1.0280	1.1990	1.121944	164	184.3315
8	8	OFFICA MAKET BİCAGI UCU	1.6888	2.0129	1.858820	143	265.2704
9	9	OFICA FH4550 HESAP MAKİNASI	32.0204	35.7313	33.752707	79	2649.3165
10	10	PRT.UDAG WA0030 SAKLAMA KABI	15.7903	18.6048	16.825744	89	1486.5168

Bu yöntemle çok çok daha hızlı şekilde sorgularımızı gerçekleştirdik.

Stored Procedure

T-SQL cümlelerinin SQL Server'ın hafızasına kaydedilerek derlendiği ve derlenmiş hallerinin çalıştırıldığı yapılardır.

Stored Procedure



Turnike Geçiş Sistemi Örneği

Stored Procedure Turnike Kartlı Geçiş Sistemi Örneği



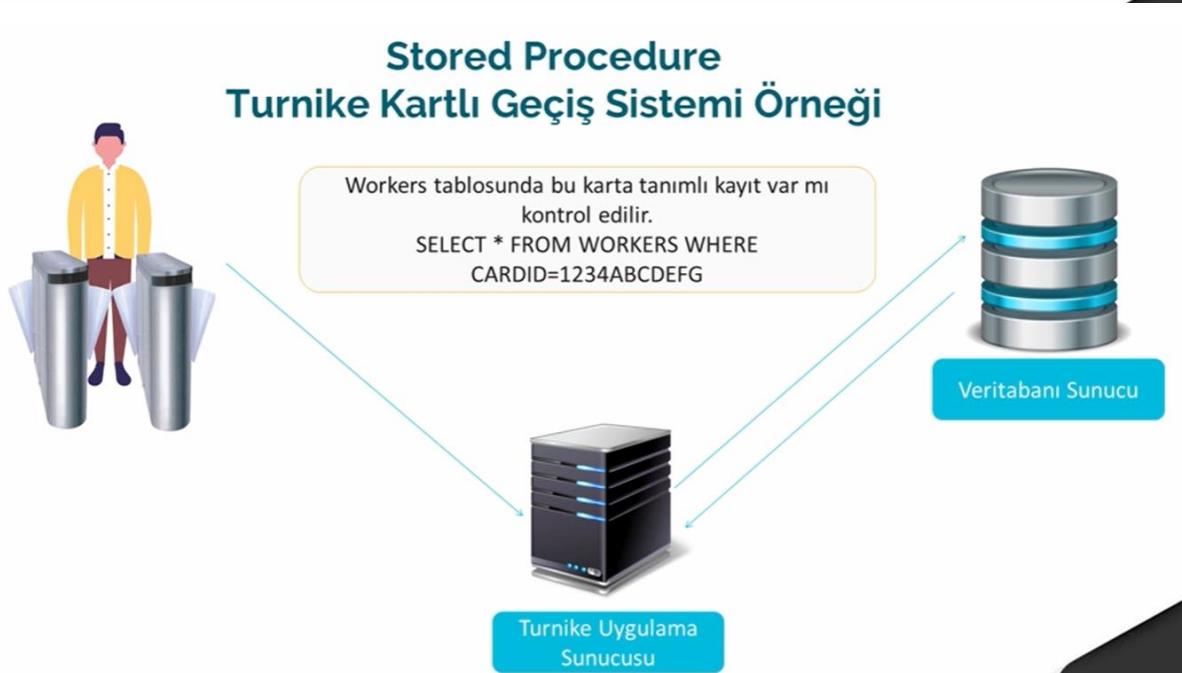
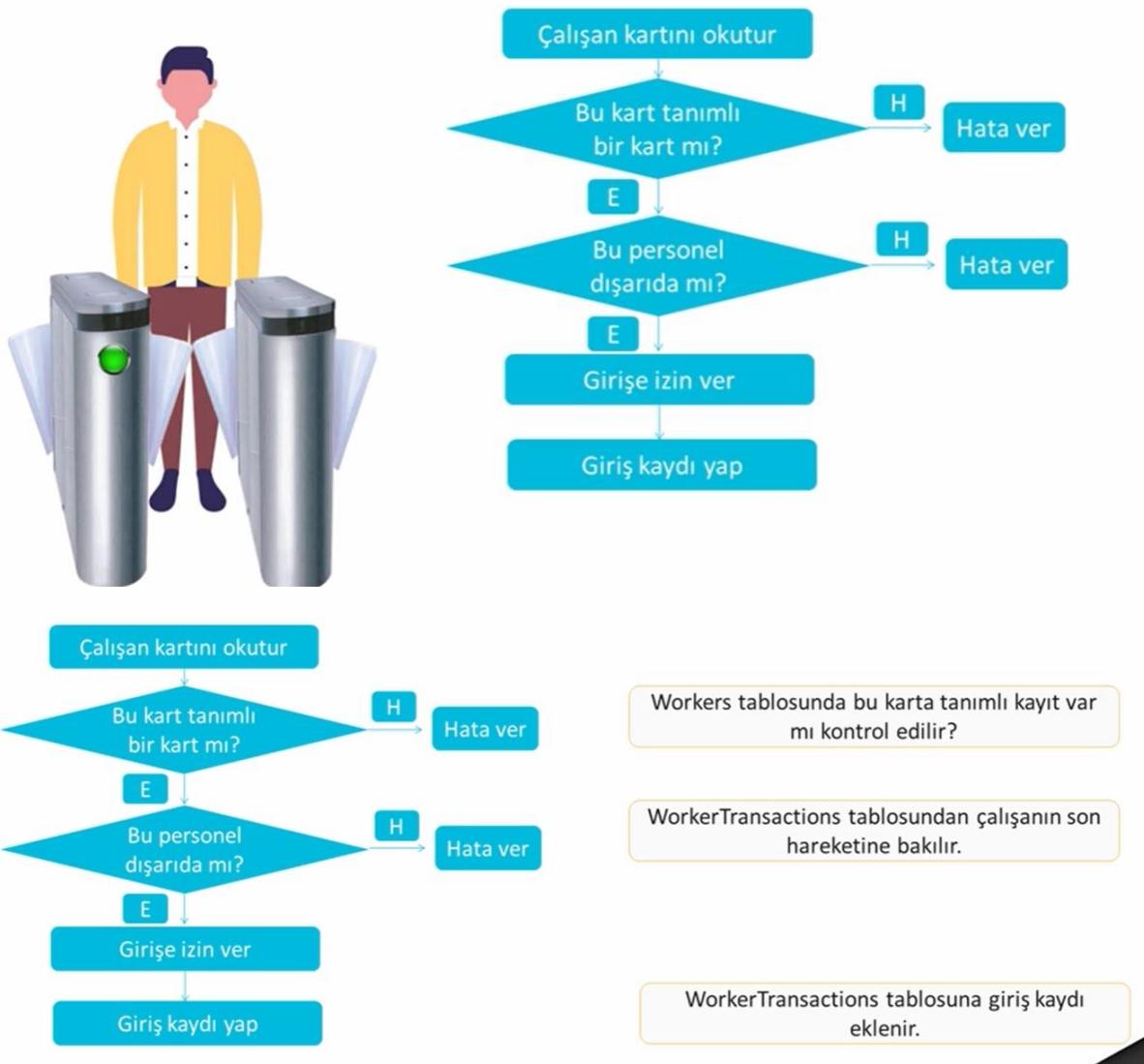
2.000 çalışan

Gün içerisinde ortalama 8 giriş-çıkış

Günlük $2.000 \times 8 = 16.000$ giriş-çıkış hareketi

Aylık $22 \times 16.000 = 352.000$ satır hareket

Yıllık $4.224.000$ hareket



Stored Procedure Turnike Kartlı Geçiş Sistemi Örneği



Stored Procedure Turnike Kartlı Geçiş Sistemi Örneği



Stored Procedure Oluşturma

```
CREATE PROCEDURE SP_CARD_CONTROL
@WORKERBARCODE AS VARCHAR(50)
AS
BEGIN
    SELECT * FROM WORKERS WHERE WORKERBARCODE=@WORKERBARCODE
END
```

Barkod numarasının geçerli olup olmadığını kontrol etmek için IF EXISTS ve RAISERROR yapısını kullanabiliriz.

```
CREATE PROCEDURE SP_CARD_CONTROL_DENEME
@WORKERBARCODE AS VARCHAR(50)
AS
BEGIN
    DECLARE @WORKERNAME AS VARCHAR(100)
    SELECT @WORKERNAME=WORKERNAME FROM WORKERS WHERE WORKERBARCODE=@WORKERBARCODE

    IF @WORKERNAME IS NULL
    BEGIN
        RAISERROR('KART GECERSIZ',16,1)
        RETURN
    END
    ELSE
    BEGIN
        SELECT @WORKERNAME
    END
END
```

EXEC SP_CARD_CONTROL_DENEME '7D53DB7F-2B73-41CB-93AB-98080D3109AC'
--Dogru kart okutunca

143 %

	(No column name)
1	Yalın TANRIKULU

```

ALTER PROC SP_WORKER_INOUT_DENEME
@WORKERBARCODE AS VARCHAR(50),
@DATE AS DATETIME,
@IOTYPE AS VARCHAR(1),
@GATEID AS INT
AS
BEGIN
    DECLARE @WORKERNAME AS VARCHAR(100)
    DECLARE @WORKERID AS INT
    SELECT @WORKERNAME=WORKERNAME, @WORKERID=ID FROM WORKERS WHERE WORKERBARCODE=@WORKERBARCODE
    IF @WORKERID IS NULL
    BEGIN
        RAISERROR('OKUTULAN KART GECERSIZ',16,1)
        RETURN
    END

    DECLARE @LASTIO AS VARCHAR(1)

    SELECT TOP 1 @LASTIO=IOTYPE FROM WORKERTRANSACTIONS
    WHERE WORKERID=1 AND DATE_>=CONVERT(DATE,GETDATE())
    ORDER BY DATE_ DESC

    IF @LASTIO=@IOTYPE
    BEGIN
        IF @IOTYPE='G'
        BEGIN
            RAISERROR('ZATEN ICERIDE GOZUKUYORSUNUZ',16,1)
            RETURN
        END
        IF @IOTYPE='C'
        BEGIN
            RAISERROR('ZATEN DISARIDA GOZUKUYORSUNUZ',16,1)
            RETURN
        END
        RETURN
    END
    INSERT INTO WORKERTRANSACTIONS (WORKERID,DATE_,IOTYPE,GATEID)
    VALUES (@WORKERID, @DATE, @IOTYPE, @GATEID)

    SELECT @WORKERID AS WORKERID, @DATE AS DATE_, @IOTYPE AS IOTYPE, @GATEID AS GATEID
END

```

```

EXEC SP_WORKER_INOUT_DENEME
@WORKERBARCODE='7D53DB7F-2B73-41CB-93AB-98080D3109AC',
@DATE='2020-01-01',
@IOTYPE='G',
@GATEID=1

```

	WORKERID	DATE_	IOTYPE	GATEID
1	1	2020-01-01 00:00:00.000	G	1

Stored Procedure'lerimizin sonuçlarını INSERT INTO EXEC komutu ile tabloya atabiliyoruz. Bu tablo ile view'da yaptığımız gibi filtreleme işlemleri yapabiliyoruz.

Farklı kullanıcıların aynı anda bu işlemi yapabileceği durumlarda #RAPOR gibi # simbolünü kullanarak sadece bir session'a özel olan geçici tablo oluşturmamız faydalı olacaktır.

Stored Procedure Kullanmanın Faydaları

Stored Procedure Kullanmanın Faydaları

Daha hızlıdır

Execution plan çıkarıldığı ve derlendiği için ilk 4 aşamayı atlar ve daha hızlı çalışır.



Daha hızlıdır

Client Server mimarisinin aksine toplu işlemler kendi içerisinde çalıştığı için sorgular network hızında değil ram hızında çalışır.



Kolay güncelleme

Yazılım güncellemeden değişiklikler yapılabılır.



Güvenlidir

SQL Injection saldırılarına karşı kesin çözümüdür.



Güvenlidir

Güvenlidir. Kritik raporlar için stored procedure bazında yetki verilebilir.



Yeteneklidir

Herhangi bir programlama dilinde yazılabilecek hemen hemen her türlü komut burada yazılp çalıştırılabilir.



Esnektiler

Stored procedure ler birbiri içerisindeinden çağrırlabilir.



Ölçülebilirdir

Performansı ölçülebilir.
Kaç kez çalıştırılmış en son ne zaman çalıştırılmış gibi bilgiler görülebilir.



Trigger

- İngilizce anlamı «Tetikleyici» demektir.
- Veritabanı tablosunda bir işlem gerçekleştiğinde başka işlemin otomatik olarak gerçekleşmesi anlamına gelir.
- Burada işlem olarak kastedilen data manipülasyonudur.
- Data manipülyonları ise Insert,Update ve Delete işlemleridir.
- Yazılan triggerlar Insert,Update ve Delete işlemlerinden sonra otomatik çalışan yapılardır.
- Trigger'ların içinde sanal olarak oluşan Inserted ve Deleted tabloları vardır.
- Inserted tablosu yeni eklenen kaydın ya da update edildiğinde değişen kaydın yeni değerini tutar.
- Deleted tablosu ise silinen kaydı ya da değiştirilen kaydın eski değerini tutar.
- Trigger'lar genelde otomatik toplam hesaplama, son değeri alma ya da loglama amacı ile kullanılır.

Toplam Tablosu Oluşturma

```
DECLARE @I AS INT=0

WHILE @I<1000
BEGIN
    DECLARE @ITEMID AS INT
    DECLARE @DATE AS DATETIME
    DECLARE @AMOUNT AS INT
    DECLARE @IOTYPE AS SMALLINT

    SET @ITEMID=ROUND(RAND())*500,0)
    IF @ITEMID=0
        SET @ITEMID=1
    SET @DATE=DATEADD(DAY,-ROUND(RAND())*365,0), GETDATE())
    SET @AMOUNT=ROUND(RAND())*9,0)+1
    SET @IOTYPE=ROUND(RAND())*1,0)+1
    INSERT INTO ITEMTRANSACTIONS
    (ITEMID,DATE_,AMOUNT,IOTYPE)
    VALUES (@ITEMID,@DATE,@AMOUNT,@IOTYPE)
    SET @I=@I+1
END

SELECT * FROM ITEMS

SELECT IOTYPE,SUM(AMOUNT),COUNT(IOTYPE) FROM ITEMTRANSACTIONS WHERE ITEMID=1
GROUP BY IOTYPE

SELECT *,
    (SELECT SUM(AMOUNT) FROM ITEMTRANSACTIONS WHERE ITEMID=ITM.ID AND IOTYPE=1)
    -
    (SELECT SUM(AMOUNT) FROM ITEMTRANSACTIONS WHERE ITEMID=ITM.ID AND IOTYPE=2) AS STOK
FROM ITEMS ITM
```

I

ID	ITEMCODE	ITEMNAME	STOK
1	URUN001	ORON 1	-16
2	URUN002	ORON 2	-447
3	URUN003	ORON 3	-658
4	URUN004	ORON 4	285
5	URUN005	ORON 5	258
6	URUN006	ORON 6	171
7	URUN007	ORON 7	476
8	URUN008	ORON 8	-97
9	URUN009	ORON 9	222
10	URUN010	ORON 10	-226
11	URUN011	ORON 11	356
12	URUN012	ORON 12	-93
13	URUN013	ORON 13	552
14	URUN014	ORON 14	-193
15	URUN015	ORON 15	-440
16	URUN016	ORON 16	-636
17	URUN017	ORON 17	210
18	URUN018	ORON 18	173
19	URUN019	ORON 19	242

Bir STOCK tablosu oluşturup bu verileri tabloya atarsak, sonraki sorgularımızda bu tablodan verileri çekersek çok daha az maliyet ile işlemi gerçekleştirmiş oluruz.

Toplam Tablosu Insert Trigger

The screenshot shows the SQL Server Management Studio interface. On the left, there's a tree view of database objects under 'TRG'. The 'Triggers' node is expanded, and the 'TRG_TRANSACTION_INSERT' trigger is selected. On the right, the script pane displays the T-SQL code for the trigger:

```
CREATE TRIGGER TRG_TRANSACTION_INSERT
ON ITEMTRANSACTIONS
AFTER INSERT
AS
BEGIN
    DECLARE @ITEMID AS INT
    DECLARE @AMOUNT AS INT
    DECLARE @IOTYPE AS SMALLINT

    SELECT @ITEMID=ITEMID,@AMOUNT=AMOUNT,@IOTYPE=IOTYPE FROM INSERTED

    IF @IOTYPE=1
        UPDATE STOCK SET STOCK=STOCK+@AMOUNT WHERE ITEMID=@ITEMID
    IF @IOTYPE=2
        UPDATE STOCK SET STOCK=STOCK-@AMOUNT WHERE ITEMID=@ITEMID
END
```

Below the trigger definition, there is a script pane containing the following T-SQL command:

```
INSERT INTO ITEMTRANSACTIONS (ITEMID,AMOUNT,IOTYPE,DATE_)
VALUES (1,3,2,GETDATE())
```

Bu trigger sayesinde sonraki insert işlemlerimiz sonucunda Stock tablomuza da ekleme yapacak.

Toplam Tablosu Delete Trigger

```
CREATE TRIGGER TRG_TRANSACTION_DELETE
ON ITEMTRANSACTIONS
AFTER DELETE
AS
BEGIN
DECLARE @ITEMID AS INT
DECLARE @AMOUNT AS INT
DECLARE @IOTYPE AS SMALLINT

SELECT @ITEMID=ITEMID,@AMOUNT=AMOUNT,@IOTYPE=IOTYPE FROM DELETED

IF @IOTYPE=1
    UPDATE STOCK SET STOCK=STOCK-@AMOUNT WHERE ITEMID=@ITEMID
IF @IOTYPE=2
    UPDATE STOCK SET STOCK=STOCK+@AMOUNT WHERE ITEMID=@ITEMID
END
```

Toplam Tablosu Update Trigger

```
CREATE TRIGGER TRG_TRANSACTION_UPDATE
ON ITEMTRANSACTIONS
AFTER UPDATE
AS
BEGIN
DECLARE @ITEMID AS INT
DECLARE @IOTYPE AS INT
DECLARE @OLDAMOUNT AS INT
DECLARE @NEWAMOUNT AS INT
DECLARE @AMOUNT AS INT

SELECT @ITEMID=ITEMID,@IOTYPE=IOTYPE,@OLDAMOUNT=AMOUNT FROM DELETED

SELECT @NEWAMOUNT=AMOUNT FROM INSERTED
SELECT @AMOUNT=@OLDAMOUNT-@NEWAMOUNT

IF @IOTYPE=1
    UPDATE STOCK SET STOCK=STOCK-@AMOUNT WHERE ITEMID=@ITEMID
IF @IOTYPE=2
    UPDATE STOCK SET STOCK=STOCK+@AMOUNT WHERE ITEMID=@ITEMID
END
```

Trigger ile Son Kaydı Loglama

--INSERT INTO SELECT komutu ile WORKERS tablomuzu dolduracagiz.

```
INSERT INTO WORKERS
([WORKERCODE], [WORKERNAME], [GENDER], [BIRTHDATE],
[TCNO], [WORKERBARCODE])
SELECT TOP 1000
TCNO AS WORKERCODE, NAMESURNAME AS WORKERNAME, GENDER,
BIRTHDATE, TCNO, NEWID() AS WORKERBARCODE
FROM CRM.DBO.CUSTOMERS

SELECT * FROM CRM.DBO.CUSTOMERS
```

156 %

Results Messages

ID	NAMESURNAME	GENDER	BIRTHDATE	CITY	TOWN	TELNR	NAME_	SURNAME	TCNO
1	Yalçın TANRIKULU	E	1958-07-26	Tokat	Erbaa	03563355976	Yalçın	TANRIKULU	97952403152
2	Tülin AKMAN	K	1961-08-16	Bitlis	Ahlat	04342374611	Tülin	AKMAN	82864312043
3	Nuri ENİS	E	1952-09-03	Erzincan	Refahiye	04462346157	Nuri	ENİS	00961869075
4	Gamze AÇICI	K	1998-03-18	Mardin	Dargeçit	04823340868	Gamze	AÇICI	82129355628
5	Muzaffer CANSOY	E	1975-08-29	Erzincan	Üzümlü	04463388810	Muzaffer	CANSOY	55884482980
6	Sude CÜLÜ	K	1991-02-14	Diyarbakır	Hazro	04122339756	Sude	CÜLÜ	30526658079
7	Selma CÜLÜ	K	1970-12-04	İstanbul	Küçükçekmece	02122365878	Selma	CÜLÜ	87342951107
8	Diyar KARPAK	E	1951-12-15	Konya	Derbent	03323349018	Diyar	KARPAK	46595418699
9	Ece ÖNLÜSAVURAN	K	1975-10-18	Siirt	Tillo	04842365859	Ece	ÖNLÜSAVURAN	24433947884
10	Bedirhan HOCAOGLU	E	1977-11-11	Gaziantep	Karkamış	03423365812	Bedirhan	HOCAOGLU	54529670938

--Calisanin son hareket turu ve zamanini bulan script

```
SET STATISTICS IO ON
SELECT *,
(SELECT TOP 1 IOTYPE FROM WORKERTRANSACTIONS WHERE WORKERID=W.ID ORDER BY DATE_ DESC) SONHAREKETTURU,
(SELECT MAX(DATE_) FROM WORKERTRANSACTIONS WHERE WORKERID=W.ID) SONHAREKETZAMANI
FROM WORKERS W
```

129 %

Results Messages

ID	WORKERCODE	WORKERNAME	GENDER	BIRTHDATE	TCNO	WORKERBARCODE	SONHAREKETTURU	SONHAREKETZAMANI	
1	97952403152	Yalçın TANRIKULU	E	1958-07-26	97952403152	321E66A6-3A02-49D5-A6F6-275D2B87D535	C	2019-12-28 17:09:00.000	
2	3	00961869075	E	1952-09-03	00961869075	6C6F3A11-A998-49F6-BCCC-98ECA8215384	C	2019-12-31 16:54:00.000	
3	10	54529670938	Bedirhan HOCAOGLU	E	1977-11-11	54529670938	5DCF353F-DDA0-4F3C-8E37-5811A761A50E	C	2019-12-31 16:53:00.000
4	12	86924314173	Oykun KARABAĞ	K	1970-04-08	86924314173	1CE7EC53-8168-4416-AE5-180A769P9ECE	C	2019-12-30 17:00:00.000
5	35	09046444034	Muhammed Ali MOROĞLU	E	1973-12-19	09046444034	3EDA70D9-1BD6-4921-97CB-3D1E159C2310	C	2019-12-28 16:55:00.000
6	37	33224628380	Halime ALPDAĞ	K	1958-05-19	33224628380	FC3CE376-9608-465C-B6CA-8D333A5CE508	C	2019-12-31 16:54:00.000
7	42	88194168421	Eşile KITAY	K	1995-03-10	88194168421	35C9E51F-2462-4C0E-8746-BA238F8F65F5	C	2019-12-31 16:54:00.000
8	44	26513227152	Meryem YALINCAN	K	1962-08-03	26513227152	2D5BE98F-4529-4F02-86B5-19FE7422441A	C	2019-12-31 17:03:00.000
9	67	25696331037	Derin ULUBEY	K	1970-10-07	25696331037	36ECAG6A-0572-4E11-A264-EC9AAB88D323	C	2019-12-31 16:54:00.000
10	69	08688996906	Mustafa FRIDAY	E	1994-11-12	08688996906	9326EC45-4FF9-4405-9B4B-6B907E4AB9AC	C	2019-12-31 16:56:00.000
11	71	31588192151	İkra PAKZAD	K	1972-10-08	31588192151	33DE91AF-93BD-468B-AE68-5D65A6665BFC	C	2019-12-31 17:08:00.000
12	76	72360412941	Zehra ALPDAĞ	K	1987-11-10	72360412941	06D35A87-6A87-471D-A7C8-E89D67316317	C	2019-12-31 17:09:00.000
13	78	63469362069	Onur MATA	E	1969-04-25	63469362069	945E2A26-ACC6-4D24-B929-C6AE48CC5989	C	2019-12-28 17:06:00.000

Bu işlemi bu şekilde yapmak çok maliyetlidir. GB'larda veri okuması gereklidir.

Her çalışanın son işlemini yeni bir tabloya aktararak bu sorunu çözebiliriz.

```

--Son islemi alan trigger
ALTER TRIGGER TRG_TRANSACTION_INSERT
ON WORKERTRANSACTIONS
AFTER INSERT
AS
BEGIN
    DECLARE @WORKERID AS INT
    DECLARE @DATE AS DATETIME
    DECLARE @IOTYPE AS VARCHAR(1)

    SELECT @WORKERID=WORKERID, @DATE=DATE_, @IOTYPE=IOTYPE FROM INSERTED

    UPDATE WORKER_LAST_TRANSACTIONS SET LASTDATE=@DATE, LASTIOTYPE=@IOTYPE
    WHERE WORKERID=@WORKERID
END

EXEC GENERATE_WORKER_TRANSACTION 1

SELECT * FROM WORKER_LAST_TRANSACTIONS

```

129 % ▾

	ID	WORKERID	LASTDATE	LASTIOTYPE
1	1	1	2019-12-31 16:54:00.000	C
2	2	2	2019-12-31 17:00:00.000	C
3	3	3	2019-12-31 16:54:00.000	C
4	4	4	2019-12-31 17:00:00.000	C
5	5	5	2019-12-31 16:55:00.000	C

Trigger ile Loglama

E-Trade veritabanında yeni bir ürün eklendiğinde, güncellendiğinde ya da silindiğinde o kaydı loglayacak bir trigger yazacağımız.

Öncelikle log'ları tutacağımız bir tablo oluşturalım.

	Column Name	Data Type	Allow Nulls
▶	ID	int	<input type="checkbox"/>
	ITEMCODE	varchar(50)	<input checked="" type="checkbox"/>
	ITEMNAME	varchar(100)	<input checked="" type="checkbox"/>
	UNITPRICE	float	<input checked="" type="checkbox"/>
	CATEGORY1	varchar(50)	<input checked="" type="checkbox"/>
	CATEGORY2	varchar(50)	<input checked="" type="checkbox"/>
	CATEGORY3	varchar(50)	<input checked="" type="checkbox"/>
	CATEGORY4	varchar(50)	<input checked="" type="checkbox"/>
	BRAND	varchar(50)	<input checked="" type="checkbox"/>
	LOG_ACTIONTYPE	varchar(20)	<input checked="" type="checkbox"/>
	LOG_DATE	datetime	<input checked="" type="checkbox"/>
	LOG_USERNAME	varchar(50)	<input checked="" type="checkbox"/>
	LOG_PROGRAMNAME	varchar(50)	<input checked="" type="checkbox"/>
	LOG_HOSTNAME	varchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

```
CREATE TRIGGER TRG_ITEMS_UPDATE
ON ITEMS
AFTER UPDATE
AS
BEGIN

    INSERT INTO ITEMS_LOG
        ([ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
        [CATEGORY3], [CATEGORY4], [BRAND], [LOG_ACTIONTYPE], [LOG_DATE],
        [LOG_USERNAME], [LOG_PROGRAMNAME], [LOG_HOSTNAME])

    SELECT [ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
        [CATEGORY3], [CATEGORY4], [BRAND], 'UPDATE', GETDATE(), SUSER_NAME(),
        PROGRAM_NAME(), HOST_NAME()
    FROM DELETED
END
```

128 % < Messages
Commands completed successfully.
Completion time: 2020-08-12T00:32:52 9731880+03:00

```
--UPDATE TRIGGER TEST
SELECT * FROM ITEMS_LOG
UPDATE ITEMS SET UNITPRICE=80 WHERE ID=3
```

128 %

	ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4	BRAND	LOG_ACTIONTYPE
1	3	9	PILLI SESLİ UCAK	53,2	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK	UPDATE

LOG_ACTIONTYPE	LOG_DATE	LOG_USERNAME	LOG_PROGRAMNAME	LOG_HOSTNAME
UPDATE	2020-08-12 00:53:18.093	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU

```
--DELETE TRIGGER
CREATE TRIGGER TRG_ITEMS_DELETE
ON ITEMS
AFTER DELETE
AS
BEGIN
    INSERT INTO ITEMS_LOG
        ([ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
        [CATEGORY3], [CATEGORY4], [BRAND], [LOG_ACTIONTYPE], [LOG_DATE],
        [LOG_USERNAME], [LOG_PROGRAMNAME], [LOG_HOSTNAME])
    SELECT [ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
        [CATEGORY3], [CATEGORY4], [BRAND], 'DELETE', GETDATE(), SUSER_NAME(),
        PROGRAM_NAME(), HOST_NAME()
    FROM DELETED
END
-- DELETE TRIGGER TEST
DELETE FROM ITEMS WHERE ID=3
SELECT * FROM ITEMS_LOG
```

128 %

	ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4	BRAND	LOG_ACTIONTYPE
1	3	9	PILLI SESLİ UCAK	53,2	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK	UPDATE
2	3	9	PILLI SESLİ UCAK	80	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK	DELETE

LOG_DATE	LOG_USERNAME	LOG_PROGRAMNAME	LOG_HOSTNAME
2020-08-12 00:53:18.093	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU
2020-08-12 01:00:31.080	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU

```

--Tek Trigger ile iki işlem
CREATE TRIGGER TRG_ITEMS_DELETE_UPDATE
ON ITEMS
AFTER DELETE, UPDATE
AS
BEGIN
--Eğer ben update yapıyorsam inserted ve deleted tablolarım dolu
--Eğer ben delete yapıyorsam inserted boş, deleted tablom dolu
DECLARE @DELETEDCOUNT AS INT
DECLARE @INSERTEDCOUNT AS INT

SELECT @DELETEDCOUNT=COUNT(*) FROM DELETED
SELECT @INSERTEDCOUNT=COUNT(*) FROM INSERTED

DECLARE @LOG_ACTIONTYPE AS VARCHAR(20)

IF @DELETEDCOUNT>0 AND @INSERTEDCOUNT>0
SET @LOG_ACTIONTYPE='UPDATE'

IF @DELETEDCOUNT>0 AND @INSERTEDCOUNT=0
SET @LOG_ACTIONTYPE='DELETE'

INSERT INTO ITEMS_LOG
([ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
[CATEGORY3], [CATEGORY4], [BRAND], [LOG_ACTIONTYPE], [LOG_DATE],
[LOG_USERNAME], [LOG_PROGRAMNAME], [LOG_HOSTNAME])

SELECT [ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
[CATEGORY3], [CATEGORY4], [BRAND], @LOG_ACTIONTYPE, GETDATE(), SUSER_NAME(),
PROGRAM_NAME(), HOST_NAME()
FROM DELETED
END
-- TRG_ITEMS_DELETE_UPDATE TEST
UPDATE ITEMS SET ITEMNAME='isim degistirildi' WHERE ID=6
DELETE FROM ITEMS WHERE ID=7
SELECT * FROM ITEMS_LOG

```

116 %

Results Messages

ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4	BRAND
1	3	PILLI SESLİ UCAK	53,2	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
2	3	PILLI SESLİ UCAK	80	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
3	6	16	19,55	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
4	7	17	1,01	EV	KITAP-DERGİ-KİRTASIYE	KİRTASIYE	KİRTASIYE GERECLERİ	KİRTASIYELER

LOG_ACTIONTYPE	LOG_DATE	LOG_USERNAME	LOG_PROGRAMNAME	LOG_HOSTNAME
UPDATE	2020-08-12 00:53:18.093	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU
DELETE	2020-08-12 01:00:31.080	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU
UPDATE	2020-08-12 01:14:57.993	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU
DELETE	2020-08-12 01:14:58.000	AzureAD\recepaydogdu	Microsoft SQL Server Management Studio - Query	RECEPAYDOGDU

Results Messages

ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4	BRAND
1	1	PIL KODAK XTRA HEAVY 9 V	5,15	EV	ELEKTRİK-ELEKTRONİK	PİL	PİL	KODAK
2	2	PIL KODAK AA'2 MAX ALKALİN KALEM	8,26	EV	ELEKTRİK-ELEKTRONİK	PİL	KALEM PİLLER	KODAK
3	4	11	20,83	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
4	5	15	30,95	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
5	6	16	19,55	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
6	8	18	1,68	EV	KITAP-DERGİ-KİRTASIYE	KİRTASIYE	KİRTASIYE GERECLERİ	KİRTASIYELER
7	9	19	31,13	EV	KITAP-DERGİ-KİRTASIYE	KİRTASIYE	KİRTASIYE GERECLERİ	KİRTASIYELER

Instead of Trigger

Instead of Trigger'larda After yerine Instead of yazılır. Bu işlem sayesinde örneğin, delete işleminden sonra değil, delete işlemi yerine çalış emri verilir. Tablodan ne kadar delete etmek istesek de o işlem gerçekleşmez ancak log tablosuna kayıt atılır.

```
-- INSTEAD OF TRIGGER
CREATE TRIGGER TRG_ITEMS_DELETE_UPDATE_INSTEADOF
ON ITEMS
INSTEAD OF DELETE, UPDATE
AS
BEGIN
--Eger ben update yapıyorsam inserted ve deleted tablolarım dolu
--Eger ben delete yapıyorsam inserted boş, deleted tablom dolu
DECLARE @DELETEDCOUNT AS INT
DECLARE @INSERTEDCOUNT AS INT

SELECT @DELETEDCOUNT=COUNT(*) FROM DELETED
SELECT @INSERTEDCOUNT=COUNT(*) FROM INSERTED

DECLARE @LOG_ACTIONTYPE AS VARCHAR(20)

IF @DELETEDCOUNT>0 AND @INSERTEDCOUNT>0
    SET @LOG_ACTIONTYPE='UPDATE'

IF @DELETEDCOUNT>0 AND @INSERTEDCOUNT=0
    SET @LOG_ACTIONTYPE='DELETE'

INSERT INTO ITEMS_LOG
([ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
[CATEGORY3], [CATEGORY4], [BRAND], [LOG_ACTIONTYPE], [LOG_DATE],
[LOG_USERNAME], [LOG_PROGRAMNAME], [LOG_HOSTNAME])
SELECT [ID], [ITEMCODE], [ITEMNAME], [UNITPRICE], [CATEGORY1], [CATEGORY2],
[CATEGORY3], [CATEGORY4], [BRAND], @LOG_ACTIONTYPE, GETDATE(), SUSER_NAME(),
PROGRAM_NAME(), HOST_NAME()
FROM DELETED
END

-- INSTEAD OF TRIGGER TEST
DELETE FROM ITEMS WHERE ID=10
SELECT*FROM ITEMS_LOG
SELECT*FROM ITEMS
```

106 % <

Results		Messages					
ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4
1	3	PILLI SESLİ UÇAK	53,2	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK
2	3	PILLI SESLİ UÇAK	80	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK
3	6	OYUNCAK BEZ BEBEK	19,55	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK
4	7	OFİCA SERİT DAKSIL	1,01	EV	KİTAP-DERGİ-KİRTASIYE	KİRTASIYE	KİRTASIYE GERECL
5	10	PRT.UDAG WA0030 SAKLAMA KABI	15,55	EV	MUTFAK GERECLERİ	MUTFAK EŞYA-GERECLERİ	SAKLAMA KABI

ID	ITEMCODE	ITEMNAME	UNITPRICE	CATEGORY1	CATEGORY2	CATEGORY3	CATEGORY4	BRAND
5	6	isim degistirildi	19,55	OYUNCAK	ZEKA GELİSTİRİCİ	OYUNCAKLAR	BEBE OYUNCAK	OYUNCAK
6	8	OFFICA MAKET BİCAGI UCU	1,68	EV	KİTAP-DERGİ-KİRTASI...	KİRTASIYE	KİRTASIYE GE...	KİRTASI...
7	9	OFİCA FH4550 HESAP MAKİNASI	31,13	EV	KİTAP-DERGİ-KİRTASI...	KİRTASIYE	KİRTASIYE GE...	KİRTASI...
8	10	PRT.UDAG WA0030 SAKLAMA KABI	15,55	EV	MUTFAK GERECLERİ	MUTFAK ES...	SAKLAMA KABI	UDAG
9	11	PRT.UDAG WA0046 TUVALET KAGIT...	19,83	TEMİZLİK	EV GERECLERİ	EV PLASTİK ...	PLASTİK	UDAG

10 ID'li item silinmedi ama log tablosuna kayıt edildi.

Transaction Kavramı ve OLTP Sistemler

SQL Server üzerinde yaptığımız her bir işlem bir transaction olarak geçer.

OLTP (On Line Transactional Processing) sistemler ise transactional yapıyı destekleyen veri tabanlarıdır.

EFT İLE PARA TRANSFERİ İŞLEMİ

A Kişisinden, B Kişisine 1.000 TL para transferi

A kişisinin hesabından
1.000 TL çıkar.

Transactions tablosuna
çıkış hareketi girilir.

A kişisinin bakiyesi
güçellenir.

Bakiye 1.000 TL azaltılır.

B kişisinin hesabına 1.000
TL girer.

Transactions tablosuna
giriş hareketi girilir.

B kişisinin bakiyesi
güçellenir.

Bakiye 1.000 TL arttırlır.

A Kişisinden, B Kişisine 1.000 TL para transferi

A kişisinin hesabından
1.000 TL çıkar.

Transactions tablosuna
çıkış hareketi girilir.



A kişisinin bakiyesi
güçellenir.

Bakiye 1.000 TL azaltılır.



B kişisinin hesabına 1.000
TL girer.

Transactions tablosuna
giriş hareketi girilir.



B kişisinin bakiyesi
güçellenir.

Bakiye 1.000 TL arttırlır.



B kişisine göre A kişi yalandı parayı gönderdim diyor ama göndermedi.

B kişinin hesap bakiyesi hala 0.

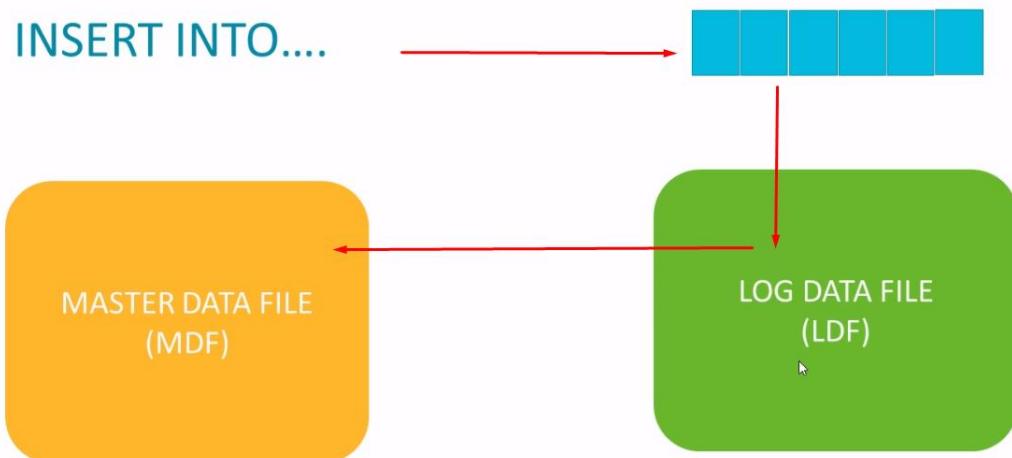
A kişisine göre B kişi yalandı hesaptan para çıktı ama o almadım diyor.

A kişinin hesap bakiyesi hala 0.

OLTP sisteme buradaki tüm işlem bir transaction olarak görülür ve herhangi bir adımda problem çıktığında sistem kendisini hiçbir şey olmamış gibi geri alır.

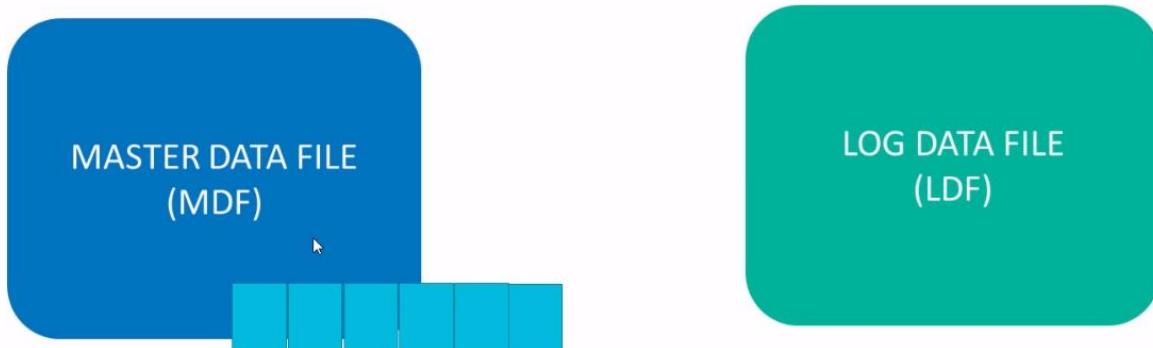
Bu iş yazılımcının yeteneğine bırakılmadan veritabanı tarafında halledilmesi gereklidir.

SQL Server OLTP Yapısı



Insert into ile eklediğimiz data page'ler ilk önce LOG DATA FILE'a yazılır. LOG DATA FILE üzerinden MASTER DATA FILE'a gelir. Herhangi bir problem yoksa transaction commit edilir ve artık gerçek data içerisinde yazılmış olur.

INSERT INTO....



ROLLBACK

MDF tarafından bir problem ile karşılaşılması durumunda sistem kendisini ROLLBACK yapar. Data page MDF'de hiç yazılmadan LDF'ye geri gönderilir ve sonrasında o data page silinir, hiç olmamış gibi görünür.

Database Oluşturma

Senaryo: Ahmet, Ömer'e 1.000 TL para gönderiyor.

A screenshot of SQL Server Management Studio (SSMS) showing a query execution window. The query is:

```
SELECT C.CUSTOMERNAME, A.ACCTNO, A.ACCTNAME, A.CURRENCY, B.BALANCE FROM ACCOUNTBALANCE B  
INNER JOIN ACCOUNTS A ON A.ID=B.ACCTID  
INNER JOIN CUSTOMERS C ON C.ID=A.CUSTOMERID
```

The results pane shows two rows of data:

	CUSTOMERNAME	ACCTNO	ACCTNAME	CURRENCY	BALANCE
1	AHMET	HESAPNO1	AHMETİN HESABI	TL	1000
2	ÖMER	HESAPNO2	ÖMER İN HESABI	TL	0

Banka Transaction Uygulaması

```
--AHMET ÖMER'E 1.000 TL PARA GÖNDERİYOR.  
INSERT INTO MONEYTRANSACTIONS  
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)  
VALUES  
(1, 2, 1000, '20200414 14:21:36', 'TL', '12345678', '')  
  
UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE-1000  
WHERE ACCOUNTID=1  
  
INSERT INTO MONEYTRANSACTIONS  
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)  
VALUES  
(2, 1, 1000, '20200414 15:21:36', 'TL', '1234567', '0987654321')  
  
UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE+1000  
WHERE ACCOUNTID=2  
  
SELECT C.CUSTOMERNAME, A.ACCTNO, A.ACCTNAME, A.CURRENCY, B.BALANCE FROM ACCOUNTBALANCE B  
INNER JOIN ACCOUNTS A ON A.ID=B.ACCTID  
INNER JOIN CUSTOMERS C ON C.ID=A.CUSTOMERID
```

116 %

	CUSTOMERNAME	ACCTNO	ACCTNAME	CURRENCY	BALANCE
1	AHMET	HESAPNO1	AHMETİN HESABI	TL	0
2	ÖMER	HESAPNO2	ÖMER İN HESABI	TL	1000

--Peki zincirin herhangi bir yerinde bir hata çıkarsa?
TRUNCATE TABLE MONEYTRANSACTIONS

```
UPDATE ACCOUNTBALANCE SET BALANCE=1000 WHERE ACCOUNTID=1  
UPDATE ACCOUNTBALANCE SET BALANCE=0 WHERE ACCOUNTID=2
```

116 %

	CUSTOMERNAME	ACCTNO	ACCTNAME	CURRENCY	BALANCE
1	AHMET	HESAPNO1	AHMETİN HESABI	TL	1000
2	ÖMER	HESAPNO2	ÖMER İN HESABI	TL	0

```

--Peki zincirin herhangi bir yerinde bir hata cikarsa?
/*
TRUNCATE TABLE MONEYTRANSACTIONS

UPDATE ACCOUNTBALANCE SET BALANCE=1000 WHERE ACCOUNTID=1
UPDATE ACCOUNTBALANCE SET BALANCE=0 WHERE ACCOUNTID=2
*/

BEGIN TRAN --transaction islemi basladi
INSERT INTO MONEYTRANSACTIONS
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)
VALUES
(1, 2, 1000, '20200414 14:21:36', 'TL', '12345678', ' ')
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE-1000
WHERE ACCOUNTID=1
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

INSERT INTO MONEYTRANSACTIONS
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)
VALUES
(2, 1, 1000, '20200414 15:21:36', 'TL', '1234567', '0987654321')
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE+1000
WHERE ACCOUNTID=2
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

COMMIT TRAN --islemi gerceklestir.
--Bu adima kadar islem LDF dosyasinda tutulur.
--COMMIT TRAN sonrasinda MDF dosyasina aktarilir.

```

116 %

	CUSTOMERNAME	ACCOUNTNO	ACCOUNTNAME	CURRENCY	BALANCE
1	AHMET	HESAPNO1	AHMETIN HESABI	TL	0
2	ÖMER	HESAPNO2	ÖMER İN HESABI	TL	1000

Peki hatalı işlem yapılrsa ?

```
--Peki zincirin herhangi bir yerinde bir hata cikarsa?

TRUNCATE TABLE MONEYTRANSACTIONS

UPDATE ACCOUNTBALANCE SET BALANCE=1000 WHERE ACCOUNTID=1
UPDATE ACCOUNTBALANCE SET BALANCE=0 WHERE ACCOUNTID=2


BEGIN TRAN --transaction islemi basladi
INSERT INTO MONEYTRANSACTIONS
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)
VALUES
(1, 2, 1000, '20200414 14:21:36', 'TL', '12345678', '')
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE-1000
WHERE ACCOUNTID=1
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

INSERT INTO MONEYTRANSACTIONS
(ACCOUNTID, TRANTYPE, AMOUNT, DATE_, CURRENCY, EFTCODE1, EFTCODE2)
VALUES
(2, 1, 1000, '20200414 15:21:36', 'TL', '1234567', 'abcdefghasd0987654321')
IF @@ERROR > 0 --Hata verecek, EFTCODE2 10 karakterden uzun olmamali.
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

UPDATE ACCOUNTBALANCE SET BALANCE=BALANCE+1000
WHERE ACCOUNTID=2
IF @@ERROR > 0
BEGIN
    ROLLBACK TRAN --islemi geri al
    RETURN
END

COMMIT TRAN --islemi gecerlestir.
--Bu adima kadar islem LDF dosyasinda tutulur.
--COMMIT TRAN sonrasinda MDF dosyasina aktarilir.
```

	CUSTOMERNAME	ACCOUNTNO	ACCOUNTNAME	CURRENCY	BALANCE	
1	AHMET	HESAPNO1	AHMETIN HESABI	TL	1000	Değişim olmadı..
2	OMER	HESAPNO2	ÖMER İN HESABI	TL	0	

Transaction ile Tabloyu Kilitleme (NOLOCK)

```
SELECT * FROM ITEMS WHERE ID=4

BEGIN TRAN
UPDATE ITEMS SET UNITPRICE=55 WHERE ID=4
--Transaction actik fakat kapatmadik.
--Onceden olusturdugumuz trigger da calisti :)

SELECT * FROM ITEMS_LOG WHERE ID=4

ROLLBACK

--Rollback isleminden sonra trigger'in ekledigi kayit da silindi.
--Trigger'lar, Transaction'larin bir parcasidir.

--Transaction islemi sirasinda islem yaptigimiz satir kilitlenir.
--Diger kullanilar islem yaptigimiz tablodan veri cekmek istediklerinde
--Kilide takilmadan islem yapmalari gereklidir. Bu islemi NOLOCK ile yapariz.
```

```
DBCC OPENTRAN --Acik transactionlar'i gosterir
```

140 %

Messages

```
Transaction information for database 'ETRADE'.

Oldest active transaction:
    SPID (server process ID): 51
    UID (user ID) : -1
    Name : user_transaction
    LSN : (108:4456:1)
    Start time : Aug 20 2020 3:08:45:650AM
    SID : 0x010500000000000c01000000029cc7ebfe986d49a0bbab110dd79edf
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

Completion time: 2020-08-20T03:22:57.5171062+03:00
```

```
SELECT * FROM ITEMS
WITH (NOLOCK)
WHERE ID=4
```

Backup / Restore İşlemleri

Backup Alma Yöntemleri

FULL BACKUP

- Tüm veritabanının yedegisinin alınması işlemidir.
- Sistem çalışırken online olarak alınabilir.
- Sıkıştırma parametresi ile %95'lere kadar sıkıştırılarak alınabilir.
- Sadece backup alınan zamana dönülebilir.

DIFFERENTIAL BACKUP

- Alınan son full backup ile differential backup alınan zaman arasındaki değişim farkının alındığı backup türüdür.
- Full backup'a göre daha az yer kaplar.
- Sistem çalışırken online olarak alınabilir.
- Sıkıştırma parametresi ile %95'lere kadar sıkıştırılarak alınabilir.
- Sadece backup alınan zamana dönülebilir.

TRANSACTION LOG BACKUP

- Transaction log dosyasının yedeklenmesidir.
- En son alınan backup hangisi ise (Full, Differential ya da Transaction Log farketmez) onunla arasındaki değişimi alır.
- Sistem çalışırken online olarak alınabilir.
- Sıkıştırma parametresi ile %95'lere kadar sıkıştırılarak alınabilir.
- Saniyelik olarak istediğimiz ana dönme imkanı verir.

Backup Planlama Stratejisi

00:00 Full Backup

148 GB Backup Boyutu
24 Saat Risk

**06:00 Full
Backup (100GB)**

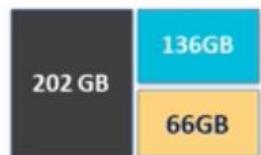
**12:00 Full
Backup
(112 GB)**

**472 GB Backup Boyutu
6 Saat Risk**

**18:00
Full Backup
(124 GB)**

**00:00
Full Backup
(136 GB)**





702 GB Backup Boyutu
Yarım Saat Risk

SQL Server Agent

SQL Server Agent Kavramı

SQL Server'da otomatik ve periyodik olarak çalışmasını istediğimiz işlemler için kullandığımız yapıdır.

- **Yedek alma**
- **Index bakım planları**
- **Database shrink planları**
- **Veri ambarı doldurma**
- **Otomatik raporlar oluşturma**
- **Mail gönderme**
- **...**

Job, Schedule, Alert gibi kavramlar vardır.

Job: Periyodik ve otomatik olarak çalışmasını istediğimiz iş blokları.

Schedule: Bu iş bloklarının takvimleri, hangi periyotlarda çalışacağıının bilgisinin tutulduğu yerler.

Alert: Uyarı mekanizmaları.

Tüm konfigürasyonunu MSDB sistem database'inde tutar.

Kaynakça

- Uygulamalarla SQL Öğreniyorum – Ömer Çolakoğlu
<https://www.udemy.com/course/sql-ogreniyorum/>
- Uçtan Uca SQL Server – Ömer Çolakoğlu
<https://www.udemy.com/course/uctan-uca-sql-server-egitimi/>
- Her Yönüyle SQL Server – Ömer Çolakoğlu
<https://www.btkakademi.gov.tr/portal/course/her-yonuyle-sql-server-9007#!/about>