

T.C.
MİLLÎ EĞİTİM BAKANLIĞI



MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

T-SQL

ANKARA 2008

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR	iii
GİRİŞ	1
ÖĞRENME FAALİYETİ - 1	3
1. T-SQL.....	3
1.1. Transact SQL Kavramı	3
1.2. Veri Tanımlama Dili (DDL)	4
1.2.1. CREATE (Nesne Oluşturmak)	4
1.2.2. ALTER (Nesnelerde Değişiklik Yapmak).....	5
1.2.3. DROP (Nesne Silmek).....	5
1.3. Veri İşleme Dili (DML)	5
1.3.1. SELECT Komutu	6
1.3.2. INSERT Komutu	7
1.3.3. UPDATE Komutu	7
1.3.4. DELETE Komutu	7
1.4. Veri Kontrol Dili (DCL)	8
1.4.1. GRANT Komutu	9
1.4.2. DENY Komutu	9
1.4.3. REVOKE Komutu	10
UYGULAMA FAALİYETİ	11
ÖLÇME VE DEĞERLENDİRME	15
ÖĞRENME FAALİYETİ - 2	16
2. T-SQL İLE ÇALIŞMAK.....	16
2.1. Değişkenler	16
2.1.1. Nesne ve Değişken İsimlendirme Kuralları.....	16
2.1.2. Değişken Tanımlama	17
2.1.3. Açıklama Satırları.....	18
2.2. Yığın Kavramı	18
2.2.1. GO Komutu	18
2.2.2. USE Komutu.....	19
2.2.3. PRINT Komutu.....	19
2.3. İşlem Operatör Türleri	19
2.3.1. Karşılaştırma Operatörleri	19
2.3.2. Mantıksal Operatörler	21
2.3.3. Aritmetiksel Operatörler	22
2.4. Fonksiyonlar	23
2.4.1. Kümeleme Fonksiyonları.....	23
2.4.2. T-SQL’de Gruplandırma	25
2.4.3. Tarih ve Zaman Fonksiyonları.....	27
2.4.4. Karakter Fonksiyonları	29
2.5. SQL Denetim Deyimleri	33
2.5.1. IF..ELSE Yapısı.....	33
2.5.2. CASE Yapısı.....	34
2.5.3. WHILE Döngüsü	35
UYGULAMA FAALİYETİ	38
ÖLÇME VE DEĞERLENDİRME	46
MODÜL DEĞERLENDİRME	48

CEVAP ANAHTARLARI.....	49
KAYNAKÇA.....	50

AÇIKLAMALAR

KOD	481BB0044
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veri Tabanı Programcılığı
MODÜLÜN ADI	T-SQL
MODÜLÜN TANIMI	T-SQL kullanarak veri tabanı işlemleri yapabilmeye ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Ağ Veri Tabanı Kurulumu modülünü bitirmiş olmak
YETERLİK	
MODÜLÜN AMACI	Genel Amaç Gerekli ortam sağlandığında T-SQL'i kullanabileceksiniz. Amaçlar 1. T-SQL komutlarını kullanabileceksiniz. 2. T-SQL elemanlarını kullanıp çalışabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Atölye, laboratuvar, bilgi teknolojileri ortamı (internet) vb. kendi kendinize veya grupla çalışabileceğiniz tüm ortamlar Donanım: Ağ veri tabanını çalıştırabilecek yeterlikte bilgisayar, yedekleme için gerekli donanım (cd yazıcı, flash bellek), raporlama için yazıcı, kâğıt ve kalem
ÖLÇME VE DEĞERLENDİRME	<ul style="list-style-type: none">➤ Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz.➤ Modül sonunda uygulanacak ölçme araçları ile modül uygulamalarında kazandığınız bilgi ve beceriler ölçülerek değerlendirilecektir.

GİRİŞ

Sevgili Öğrenci,

Okul yaşantınızda öğreneceğiniz her konu, yaptığınız uygulama ve tamamladığınız her modül; bilgi dağarcığınızı geliştirecek ve ileride atılacağınız iş yaşantınızda size başarı olarak geri dönecektir. Eğitim sürecinde daha özverili çalışır ve çalışma disiplini kazanırsanız başarılı olmamanız için hiçbir neden yoktur.

Son yıllarda yapılan birçok proje, çok sayıda bilgisayar tarafından kullanılabilir şekilde tasarlanmaktadır. Bu yüzden, ağ ortamında birden fazla kullanıcı aynı proje üzerinde çalışabilmektedir. Bu işlemleri çok sık kullandığınız veri tabanı programıyla da yapabilmeye rağmen ağ ortamında güvenlik ve hızlı erişim açısından en iyi sonucu veren SQL Server veri tabanı da yapabilirsiniz. Bu programla milyonlarca kaydın olduğu tablolar üzerinde işlem yaparken tüm kullanıcılara hitap ederek istenilen sorgu sonuçlarını da en hızlı şekilde elde edebilirsiniz.

Bu modülle T-SQL kullanarak veri tabanı işlemlerini ve T-SQL elemanlarını kullanmayı öğreneceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

T-SQL için kullanılan dilleri ve bu dillerle ilgili komutları öğreneceksiniz.

ARAŞTIRMA

- T-SQL'in bir önceki SQL Server sürümlerine göre yeniliklerini araştırınız.

1. T-SQL

1.1. Transact SQL Kavramı

Microsoft'un veri tabanı sorgulama dilidir. Transact-SQL, SQL Server ve istemci (client) arasında iletişimi sağlayan SQL sorgulama dilinin gelişmiş bir versiyonudur. Transact Structured Query Language kelimelerinin kısaltmasıdır.

T-SQL kullanarak veri tabanına kayıt eklenebilir, silinebilir, güncellenebilir ya da sorgulama ve raporlama yapılabilir.

T-SQL ile döngü veya mantıksal işlemler yapmak için bir derleyiciye gerek yoktur. Herhangi bir programlama dili öğrenmeden de T-SQL ile tüm amaçlarınıza hitap edecek projeler gerçekleştirebilirsiniz.

T-SQL ifadelerini çalıştırabilmek için bir Management Studio ile SQL Server'a erişmeniz gerekir.

SQL deyimleri veritabanları üzerinde çeşitli işlemleri yerine getirir. Veri tabanından sorgulama yapmak için SELECT, ekleme yapmak için INSERT güncelleme yapmak için UPDATE, silme yapmak için DELETE, yeni tablo oluşturmak için CREATE TABLE gibi komutlara sahiptir.

Bu komutlar, işlevlerine göre şu şekilde ayrılır:

- DDL (Data Definition Language): Veri tanımlama dili
- DML (Data Manipulation Language) : Veri işleme dili
- DCL (Data Control Language): Veri kontrol dili

1.2. Veri Tanımlama Dili (DDL)

SQL Server içinde veri tabanı, tablo ve kullanıcı tanımlı veri tipleri gibi nesnelere oluşturmak ve bunları yapılandırmak için kullanılır. Temel komutları aşağıdaki şekildedir:

<u>Temel Komutlar</u>	<u>Açıklama</u>
CREATE	Nesne oluşturmak için kullanılır.
ALTER	Nesneler üzerinde değişiklik yapmak için kullanılır.
DROP	Nesneleri silmek için kullanılır.

1.2.1. CREATE (Nesne Oluşturmak)

Veri tabanındaki nesnelere oluşturulabilmesi için CREATE komutu kullanılır. Oluşturulacak nesnenin özellikleri dikkate alınarak farklı parametreler kullanılmalıdır.

➤ Genel Yazımı

```
CREATE nesne_adi
```

Örnek:

```
CREATE DATABASE Person
```

Person adında bir veri tabanı oluşturulur.

Örnek:

```
CREATE TABLE PERSONEL(  
    PERSONEL_ID int,  
    AD varchar(10),  
    SOYAD varchar(10)  
)
```

Bu şekilde bir yazımla PERSONEL adında bir tablo oluşturulur. Tablo sütunları da PERSONEL_ID, AD, SOYAD'dır.

Örnek:

```
CREATE TABLE PERSONELYAKIN(  
    PERSONEL_ID int,  
    YAKIN_ID int,  
    YAKIN_AD varchar(10),  
    YAKIN_SOYAD varchar(10)  
)
```

Bu örnekte de PERSONELYAKIN adında bir tablo oluşturulmuştur. Tablo sütunları da PERSONEL_ID, YAKIN_ID, YAKIN_AD, YAKIN_SOYAD'dır.

1.2.2. ALTER (Nesnelerde Değişiklik Yapmak)

Daha önceden oluşturulmuş bir nesne özelliğinin değiştirilmesini sağlar.

➤ **Genel Yazımı**

ALTER nesne **nesne_adi** değişim_cümlesi

Örnek:

```
ALTER TABLE PERSONEL  
ADD BABA_AD varchar(20) NOT NULL
```

Bu şekildeki bir yazımla PERSONEL tablosuna BABA_AD sütunu eklenmiştir. NOT NULL ile de bu sütuna veri girişi zorunlu hâle getirilmiştir.

Örnek

```
ALTER TABLE PERSONEL  
ALTER COLUMN AD varchar(15) NOT NULL
```

Bu yazım ile de varolan AD sütununun alabileceği karakter sayısı 15 olarak değiştirilmiş ve veri girişi zorunlu hâle getirilmiştir.

1.2.3. DROP (Nesne Silmek)

Bir nesnenin silinmesini sağlayan komuttur. DROP komutu tüm nesneler için kullanılır.

➤ **Genel Yazımı**

DROP nesne nesne_adi

Örnek:

```
DROP TABLE PERSONEL
```

Bu şekilde bir yazımla PERSONEL tablosu silinmiş olur.

1.3. Veri İşleme Dili (DML)

Veri tabanı içindeki veriler ile ilgili işlemler yapılmasını sağlar. Temel komutları aşağıdaki şekildedir.

Temel Komutlar

SELECT

INSERT

UPDATE

DELETE

Açıklama

Veri tabanındaki verileri seçmeyi sağlar.

Veri tabanına yeni veriler eklemek için kullanılır.

Veriler üzerinde değişiklik (güncelleme) yapmak için kullanılır.

Veri tabanından veri silmek için kullanılır.

1.3.1. SELECT Komutu

Verilere erişmek için en sık kullanılan komuttur. Bir tablodaki bir veya daha çok alan için SELECT komutu yazılabilir.

➤ **Genel Yazımı**

```
SELECT sütun_adi1, [sütun_adi2],..... [*]  
FROM tablo_adi
```

Örnek:

```
SELECT * FROM PERSONEL
```

Bu yazımla PERSONEL tablosundaki tüm alanlar seçilmiş olur.

Örnek:

```
SELECT PERSONEL_ID, AD  
FROM PERSONEL
```

Bu yazım ile de PERSONEL tablosundaki PERSONEL_ID ve AD alanları seçilmiş olmaktadır.

Örnek:

```
SELECT AD+ ' ' + SOYAD  
FROM PERSONEL
```

PERSONEL tablosunda yer alan AD ve SOYAD alanlarını tek bir sütun gibi birleştirerek göstermeyi sağlayan SELECT ifadesidir.

Örnek:

```
SELECT ad FROM rehber
```

ifadesiyle rehber tablosundaki sadece “ad” alanı bilgilerinin elde edilmesini sağlar.

Örnek:

```
SELECT * FROM rehber WHERE ad='Ali'
```

ifadesiyle ad alanındaki Ali ismindeki tüm kayıtların elde edilmesini sağlar.

Örnek:

```
SELECT * FROM rehber WHERE ad='Tuncay' ORDER BY ad ASC
```

Verilen koşullara göre sütundaki bilgileri artan (ASC) ya da azalan (DESC) sırada ekrana getirir. WHERE ile oluşturulan koşul ifadelerinde mantıksal operatörler de kullanılabilir (and, or,not).

Örnek:

```
SELECT * FROM rehber ORDER BY ad, soyad
```

ifadesiyle ad alanına göre kayıtları, adı aynı olanları da soyad alanına göre seçme işlemini gerçekleştirir.

1.3.2. INSERT Komutu

Veri tabanına yeni bir kayıt eklemek için kullanılır.

➤ Genel Yazımı

```
INSERT INTO tablo_adi (sütunadi1 [,sütunadi2,.....])  
VALUES (deger1 [,deger2, .....])
```

Örnek:

```
INSERT INTO PERSONEL (AD,SOYAD)  
VALUES ('Ceylin','Yılmaz')
```

şeklindeki ifadeyle Personel tablosunun ad ve soyad alanlarına yeni değerler ekler.

1.3.3. UPDATE Komutu

Kayıtları güncellemek için kullanılır. Hangi kayıtların güncelleneceği bir koşul veya koşullarla belirtilebilir.

➤ Genel Yazımı

```
UPDATE tablo_adi SET alan_adi=deger WHERE şart
```

Örnek:

```
UPDATE PERSONEL SET AD='Ceylin' WHERE SOYAD='Yılmaz'
```

biçimindeki bir bildirim soyadı Yılmaz olan kayıtların ad bilgisini Ceylin olarak değiştirir.

Eğer birden fazla Yılmaz soyadı olsaydı hepsinin ad alanı Ceylin olarak değiştirilecekti.

SET sözcüğü değiştirilecek kolonları ve değerleri belirtir. WHERE sözcüğü ise değiştirilecek satırı belirtir.

1.3.4. DELETE Komutu

Tablodan kayıt silmek için kullanılır.

➤ **Genel Yazımı**

```
DELETE FROM tablo_adi  
WHERE şart
```

Örnek:

```
DELETE FROM PERSONEL  
WHERE SOYAD='Yılmaz'
```

SOYAD değeri Yılmaz olan tüm kayıtları siler.

1.4. Veri Kontrol Dili (DCL)

DCL, bir veri tabanı ile ilişkili kullanıcıları ve rollerin izinlerini değiştirmek için kullanılır. Diğer bir deyişle verilere erişim yetkilerini düzenlemede kullanılır. Temel komutları aşağıdaki şekildedir.

<u>Temel Komutlar</u>	<u>Açıklama</u>
GRANT	Bir kullanıcının verileri kullanmasına ve T-SQL komutlarını çalıştırmasına izin verir.
DENY	Bir kullanıcının verileri kullanmasını kısıtlar.
REVOKE	Daha önce yapılan tüm kısıtlama ve izinleri iptal eder.

DCL komutlarını kullanabilmek için SQL Server'da varsayılan değer (default) olarak yetki sahibi olan gruplar: sysadmin , dbcreator , db_owner , db_securityadmin 'dir.

Sunucuya dışarıdan bir erişim sağlamak için bir giriş (login) oluşturulmalıdır.

Bunun için;

```
CREATE LOGIN Tuncay1 WITH PASSWORD ='123456'
```

Bu rol ile veri tabanına bir kullanıcı olarak erişim için aşağıdaki satırlar yazılmalıdır:

```
CREATE USER Tuncay  
FOR LOGIN Tuncay1
```

Eğer User adı ile Login adı aynı ise FOR LOGIN satırına gerek kalmaz.

Veri tabanında uygulama rolü oluşturulması için de aşağıdaki satırlar kullanılabilir:

```
CREATE APPLICATION ROLE Lab1  
WITH PASSWORD = 'sifre'  
, DEFAULT_SCHEMA=Lab1;
```

1.4.1. GRANT Komutu

Veri tabanı kullanıcılarına, veri tabanı rolüne veya uygulama rolüne izinler vermek için kullanılan komuttur.

➤ **Genel Yazımı**

```
GRANT {ALL veya izinler}
ON {izin_verilenler}
TO {hesaplar}
```

ALL ifadesi, tüm hakların verilebileceğini gösterir.

Örnek:

Tuncay1 adlı kullanıcıya tablo oluşturma izni şöyle verilebilir:

```
GRANT CREATE TABLE
TO Tuncay1
```

Aynı anda Lab1 rolüne de aynı izin verilebilir.

```
GRANT CREATE TABLE TO Lab1, Tuncay1
```

Örnek:

Bir kullanıcıya hem veri tabanı hem de tablo oluşturma izni şöyle verilir:

```
GRANT CREATE DATABASE, CREATE TABLE TO Ahmet
```

1.4.2. DENY Komutu

Kullanıcıların haklarını kısıtlayan komuttur.

➤ **Genel Yazımı**

```
DENY { ALL veya izinler } TO {kullanıcılar}
```

Örnek:

```
DENY CREATE TABLE
TO Tuncay1
```

Tuncay1 adlı kullanıcının tablo oluşturma yetkisi kısıtlanır.

Örnek:

```
DENY SELECT ON PERSONEL TO Tuncay1
```

PERSONEL tablosunda Tuncay1 adlı kullanıcı SELECT komutuyla ilgili işlem yapamaz.

1.4.3. REVOKE Komutu

Daha önce yapılan tüm kısıtlama ve izinleri iptal eden komuttur. Bir nesneyi oluşturan kullanıcının REVOKE ile nesne üzerindeki yetkilendirme ve kullanma hakkı yok edilemez.

REVOKE komutunu, sys_admin rolüne veya db_owner, db_securityadmin sabit veri tabanı rollerine sahip kullanıcılar ve nesne için dbo olan kullanıcı çalıştırabilir.

➤ Genel Yazımı

```
REVOKE {ALL veya izinler} {TO veya FROM} {hesaplar}
```

Örnek:

```
REVOKE ALL TO PUBLIC
```

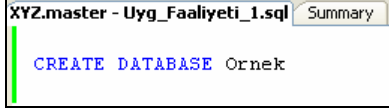
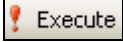
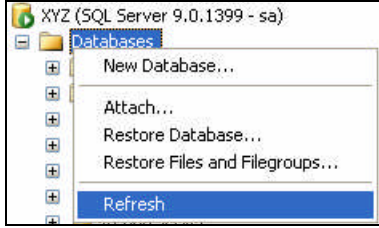
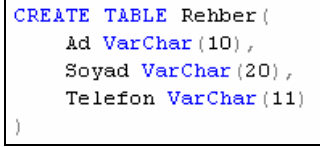
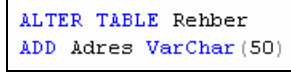
PUBLIC rolüne verilmiş olan tüm yetkiler kaldırılır.

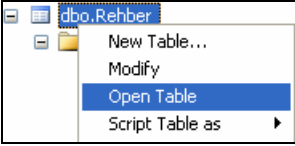
Örnek:


```
REVOKE SELECT ON PERSONEL TO PUBLIC
```

PUBLIC rolüne PERSONEL tablosunda seçim için verilen izin kaldırılır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
<ul style="list-style-type: none">➤ T-SQL'i kullanarak Ornek adında bir veri tabanı oluşturunuz.	<ul style="list-style-type: none">➤ CREATE DATABASE komutunu kullanabilirsiniz.  <p>Resim 1.1: Veri tabanı oluşturma</p>
<ul style="list-style-type: none">➤ Yazdığımız sorguyu çalıştırınız.	<ul style="list-style-type: none">➤ Klavyeden F5 tuşuna basabilir veya Execute  komutuna tıklayabilirsiniz.
<ul style="list-style-type: none">➤ Object Explorer penceresinde sorgusunu yazdığınız veri tabanının oluşturulup oluşturulmadığını kontrol ediniz.	<ul style="list-style-type: none">➤ Sorgu çalıştırıldıktan sonra Database'ı Refresh etmeyi unutmayınız.  <p>Resim 1.2: Refresh</p>
<ul style="list-style-type: none">➤ Ornek veri tabanı altında Rehber adında bir tablo oluşturunuz.➤ Tablo sütunları Ad VarChar(10), Soyad VarChar(20), Telefon VarChar (11) şeklinde olsun.	 <p>Resim 1.3: Tablonun oluşturulması</p>
<ul style="list-style-type: none">➤ Oluşturduğunuz tabloya Adres adında bir sütun ekleyiniz.	 <p>Resim 1.4: Yeni bir sütun ekleme</p>

<p>➤ Oluşturduğunuz tabloya kendi bilgilerinizi ve ailenizden iki kişinin bilgilerini ekleyiniz.</p>	<pre>INSERT INTO Rehber (Ad, Soyad, Telefon, Adres) VALUES ('Nazlı', 'ZEKİ', '03121234567', 'Ankara') INSERT INTO Rehber (Ad, Soyad, Telefon, Adres) VALUES ('Mustafa', 'BÜYÜK', '02121234567', 'İstanbul') INSERT INTO Rehber (Ad, Soyad, Telefon, Adres) VALUES ('Kadir', 'BÜYÜK', '02321234567', 'İzmir')</pre> <p style="text-align: center;">Resim 1.5: Bilgilerin tabloya eklenmesi</p>																				
<p>➤ Eklediğiniz bilgilerin tabloya yazılıp yazılmadığını kontrol ediniz.</p>	<p>➤ Tablo üzerinde sağ tıklayarak Open Table komutunu tıklayabilir ve tablonuzu açabilirsiniz.</p>  <p style="text-align: center;">Resim 1.6: Open Table komutu</p> <table border="1" data-bbox="728 948 1255 1085"> <thead> <tr> <th>Ad</th> <th>Soyad</th> <th>Telefon</th> <th>Adres</th> </tr> </thead> <tbody> <tr> <td>Nazlı</td> <td>ZEKİ</td> <td>03121234567</td> <td>Ankara</td> </tr> <tr> <td>Mustafa</td> <td>BÜYÜK</td> <td>02121234567</td> <td>İstanbul</td> </tr> <tr> <td>Kadir</td> <td>BÜYÜK</td> <td>02321234567</td> <td>İzmir</td> </tr> <tr> <td>NULL</td> <td>NULL</td> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table> <p style="text-align: center;">Resim 1.7: Tablo içeriği</p>	Ad	Soyad	Telefon	Adres	Nazlı	ZEKİ	03121234567	Ankara	Mustafa	BÜYÜK	02121234567	İstanbul	Kadir	BÜYÜK	02321234567	İzmir	NULL	NULL	NULL	NULL
Ad	Soyad	Telefon	Adres																		
Nazlı	ZEKİ	03121234567	Ankara																		
Mustafa	BÜYÜK	02121234567	İstanbul																		
Kadir	BÜYÜK	02321234567	İzmir																		
NULL	NULL	NULL	NULL																		
<p>➤ Kendi telefon numaranızı başka bir telefon numarasıyla değiştiriniz.</p>	<p>➤ UPDATE komutunu kullanabilirsiniz.</p> <pre>UPDATE Rehber SET Telefon='05051234567' WHERE Ad='Nazlı'</pre> <p style="text-align: center;">Resim 1.8: Bilginin güncellenmesi</p>																				
<p>➤ Ailenizden birinin bilgilerini siliniz.</p>	<pre>DELETE FROM Rehber WHERE Ad='Kadir'</pre> <p style="text-align: center;">Resim 1.9: Bilginin silinmesi</p>																				
<p>➤ Tablonuzda kalan bilgileri görüntüleyiniz.</p>	<p>➤ SELECT komutunu kullanabilirsiniz.</p> <pre>SELECT * FROM Rehber</pre> <p style="text-align: center;">Resim 1.10: Kayıtların gösterilmesi</p>																				

<p>➤ Results penceresinden sonucu kontrol ediniz.</p>	 <p>Resim 1.11: Results penceresi</p>
<p>➤ Tablonuzdaki bilgileri Ad sütununa göre artan ve azalan sırada görüntüleyiniz.</p>	<pre>SELECT * FROM Rehber ORDER BY Ad ASC SELECT * FROM Rehber ORDER BY Ad DESC</pre> <p>Resim 1.12: Artan ve azalan sırada gösterim</p>
<p>➤ Sunucuda “Kullanici” adında bir login oluşturunuz.</p>	<pre>CREATE LOGIN Kullanici WITH PASSWORD='123'</pre> <p>Resim 1.13: Login oluşturulması</p>
<p>➤ Sunucudaki Security/Logins alanında Login'in oluşturulup oluşturulmadığını kontrol ediniz.</p>	 <p>Resim 1.14: Logins</p>
<p>➤ Aynı isimde bir kullanıcı oluşturunuz.</p>	<pre>CREATE USER Kullanici</pre> <p>Resim 1.15: Kullanıcı oluşturulması</p>
<p>➤ Örnek veri tabanındaki Security/Users alanında kullanıcının oluşturulup oluşturulmadığını kontrol ediniz.</p>	 <p>Resim 1.16: Users</p>

- “Kullanici” adlı user’a SELECT ifadesinin kullanımını engelleyiniz.

```
DENY SELECT ON Rehber  
TO Kullanici
```

Resim 1.17: DENY ile erişimin kısıtlanması

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşluk doldurmalı sorularda boşluklara uygun cevapları yazınız.

1. Aşağıdakilerden hangisi veri tanımlama dili komutlarından değildir?
A) CREATE B) DELETE C) DROP D) ALTER
2. Veri tabanı kullanıcıasına izin vermek için Revoke komutu kullanılır (D/Y).
3. Eğer User (kullanıcı) ile Login (giriş) adı aynıysa satırını yazmaya gerek yoktur.
4. Aşağıdakilerden hangisi T-SQL ile yapılamaz?
A) Kayıt silinebilir. B) Kayıt eklenebilir.
C) Form oluşturulabilir. D) Raporlama yapılabilir.
5. Aşağıdakilerden hangisi DML komutudur?
A) SET B) UPDATE C) INTO D) CREATE
6. Bir veri tabanı ile ilişkili kullanıcıları ve rollerin izinlerini değiştirmek için kullanılan dile denir.
7. Bir kullanıcının veri tabanı veya tabloya erişimini engellemek için DENY komutu kullanılır. (D/Y)
8. DCL'de sunucuya dışarıdan bir erişim için oluşturulur.
9. Aşağıdakilerden hangisi veri tabanında çeşitli işlemleri yerine getiren dillerden değildir?
A) DCL B) DDL C) DML D) DSL
10. Kullanıcıya verilen tüm kısıtlamaları kaldırmak için GRANT komutu kullanılır. (D/Y)

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer modüle geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

T-SQL değişkenlerini, operatörlerini, deyim bloklarını ve fonksiyonlarını kullanıp çalıştırabileceksiniz.

ARAŞTIRMA

- T-SQL’de değişken kullanmaya ihtiyaç duyulmasının nedenlerini araştırınız.

2. T-SQL İLE ÇALIŞMAK

2.1. Değişkenler

Değişken, verilerin bellekte geçici olarak kaydedilmesini ve gerektiğinde kullanılmasını sağlayan değerdir. T-SQL kullanmanın en büyük kolaylıklarından biri de değişken kullanımına olanak tanımasıdır. Burada ifade edilen; değişken diğer tüm programlama dillerinde yer alan bir veri tipi ile sınırlandırılmış, oluşturulmasının ardından hafızada belli bir yer kaplayan, üzerine veri ataması yapılabilen ve daha sonra ismi kullanılarak program içerisinden çağrılıp kullanılacak yapıdır.

SQL Server’da da değişkenler yerel ve genel olmak üzere ikiye ayrılır. Yerel değişkenler, "@" ön eki ile tanımlanır (@değişken). Genel değişkenler ise SQL Server tarafından tanımlanmıştır ve kullanıcı tarafından oluşturulamaz. "@@" ön eki ile tanımlanırlar (@@SERVERNAME). Genel değişkenler genellikle SQL Server hakkındaki bilgileri verir. SQL Server’da tanımlanmış birçok genel değişken vardır.

2.1.1. Nesne ve Değişken İsimlendirme Kuralları

Nesne veya değişkene bir isimlendirme yaparken aşağıdaki kurallara dikkat etmelisiniz:

- Harf veya alt çizgi (_) ile başlamalıdır.
- Türkçe karakterler ve boşluk isimlendirmede kullanılmamalıdır.
- Değişken ismi SQL’de özel anlamı olan sembollerle (@, @@, #, ##, \$) başlamamalıdır.
- T-SQL komutları değişken ismi olarak verilmemelidir (SELECT,UPDATE vb).

- SQL ifadeleri prensip olarak büyük harfle yazılır.
- Nesne isimleri kısa ve anlamlı olmalıdır.
- Nesne isimlendirilirken işlerin kolaylaştırılması açısından tekil isim tercih edilmelidir (TabloOgrenciler yerine tblOgrenci gibi).
- NULL terimi, daha önce hiçbir şey girilmemiş (değersiz) anlamındadır. Klavyedeki SPACE (ASCII 32) tuşu ile NULL aynı değerleri içermez. NULL boş veya bilinmeyen değerler için kullanılır.

2.1.2. Değişken Tanımlama

SQL Server'da değişkenler DECLARE ifadesi kullanılarak oluşturulur.

➤ Yazım Şekli

```
DECLARE @degisken_adi <veri_tipi> [(boyut)]
```

Örnek:

```
DECLARE @ogr_no VarChar(10)
DECLARE @tckimlik_no int
```

Aralara virgül koyarak da birden fazla değişkeni tek bir DECLARE ifadesi ile oluşturabilirsiniz.

Örnek:

```
DECLARE @ogr_no varchar(10),@tckimlik_no int
```

Varchar, int türlerinde değişken tanımlayabildiğiniz gibi tablo türünde değişken de tanımlayabiliriz.

Örnek:

```
DECLARE @degisken_adi TABLE (tablo tanımı)
```

şeklindedir.

Bir değişken oluşturulduğunda NULL değere sahiptir. Değişkenlere değer atamanın SET, SELECT ve tablolar için INSERT INTO gibi birkaç farklı şekli vardır.

➤ SET ifadesi kullanılarak değişkene değer atama

```
SET @degisken_adi=değer
```

şeklinde yapılır.

➤ **SELECT ifadesiyle değer atama**

```
SELECT @değişken_adi=değer
```

şeklinde yapılır.

➤ **Tablo değişkenlere INSERT INTO ifadesi ile değer atama**
INSERT INTO @tablo_değişken SELECT adi, soyadi FROM person

ifadesi ile person tablosunun adı ve soyadı sütunlarının içerdiği değerlerden oluşan bir tabloyu @tablo_değişken adlı değişkene atamış olursunuz.

2.1.3. Açıklama Satırları

T-SQL'de bir satırın dikkate alınmamasını istiyorsanız "--" kullanabilir ya da /*.....*/ kullanabilirsiniz.

Örnek:

```
-- CREATE TABLE ogrenci
```

veya

```
/* CREATE TABLE ogrenci */
```

şeklindeki kullanımlardan birini seçebilirsiniz.

2.2. Yığın Kavramı

SQL Server'da yığın, sorguların sırayla işleme alınması demektir. Çalışma esnasında SQL Server'a gönderilen birden fazla sorgu yığınlar hâlinde ele alınır.

2.2.1. GO Komutu

Bir yığının sonunu belli etmek için **GO** komutu kullanılır. Bir yığın SQL Server'da işlemeye başladığı anda önce Parse(ayrıştırma) edilir. Daha sonra derlenerek (Compile) çalıştırılır(Execute).

➤ **Genel Yazımı**

```
Komutlar
```

```
Komutlar
```

```
GO
```

şeklindedir.

2.2.2. USE Komutu

T-SQL'de çalışacağınız veri tabanını belirme işlemi için bu komut kullanılmalıdır. USE komutuyla hangi veri tabanı üzerinde işlem yapılacağı belirtilir.

➤ Genel Yazımı

USE Veritabani_Adi
şeklindedir.

Örnek:

USE master

2.2.3. PRINT Komutu

Değişkenlerin değerlerini, hataları vb. diğer ihtiyaç duyulan olaylarda PRINT komutu kullanılır. DEBUG işlemi için ihtiyaç duyulan bir komuttur.

➤ Genel Yazımı

PRINT @degisken_adi

şeklindedir.

Örnek:

USE Deneme
GO

```
DECLARE @ad VARCHAR(10)
SELECT @ad='Mustafa'
GO
```

```
PRINT @ad
GO
```

ad değişkeninin içeriği PRINT ile gösterilmiş olacaktır.

2.3. İşlem Operatör Türleri

T-SQL'de işlem yapabilmek için bazı operatörlere ihtiyaç duyulur.

2.3.1. Karşılaştırma Operatörleri

Karşılaştırma ifadesinde karşılaştırılan verilerin türü aynı olmalıdır. Yani karakter veri türü ile ancak karakter türünde başka bir veri; bir sayısal veri ile ancak sayısal olan başka bir veri karşılaştırılabilir.

OPERATÖR	ANLAMI
<	Küçük
>	Büyük
=	Eşit
<=	Küçük veya eşit
>=	Büyük veya eşit
<>	Eşit değil
!=	Eşit değil
LIKE	Metin Karşılaştırma Operatörü

Örnek:

Öğrenci veri tabanında Tablo 2'deki ikinci notu 56'dan küçük olan öğrencileri listeleyen T-SQL kod satırlarıdır.

```
SELECT *
FROM tablo2
WHERE nt2<=56
```

Örnek:

Adı Ali olmayan kayıtları listelemek için kullanılan T-SQL kod satırlarıdır.

```
SELECT *
FROM Personel
WHERE ad<>'Ali'
```

➤ **Joker Karakterler**

Sadece LIKE operatörüyle kullanılan joker karakterler, bir veya daha fazla harfin yerine geçer. Belirli aralıklardaki belli harfle başlayan ve biten sorgularda joker karakterler kullanılır.

Joker Karakterler	Anlamı
%	Birden fazla harf ya da rakamın yerini tutar.
_	Bir tek harf veya rakamın yerini tutar.
[HARF]	Herhangi bir harf yerine gelebilecek harfleri belirtir.
[^HARF]	Herhangi bir harf yerine gelemeyecek harfleri belirtir.
[A-Z]	A ile Z arasındaki harfleri belirtir.

Örnek:

tablo1 tablosundaki ad alanı içinde baş tarafı "Er" ile başlayan isimleri listeleyen T-SQL kodlarıdır.

```
SELECT *
FROM tablo1 WHERE ad LIKE 'Er%'
```

Örnek:

tablo1 tablosundaki soyad alanı içinde baş tarafı “DE” ile başlayan soyadları listeleyen T-SQL kodlarıdır.

```
SELECT *  
FROM tablo1 WHERE soyad LIKE '%DE%'
```

2.3.2. Mantıksal Operatörler

NOT, OR ve AND mantıksal operatörleri yardımı ile birden çok koşulun gerçekleşmesine bağlı olarak ifade edilebilecek karmaşık ya da birleşik koşullu listelemeleri gerçekleştirmek mümkün olmaktadır. BETWEEN ifadesi de AND operatörü ile aynı işlemi gerçekleştirebilmektedir.

Örnek:

Doğum tarihi 1970'ten önce olan, maaşı 700 – 1200 arasında olan personeli listeleyen kodlardır.

```
SELECT *  
FROM Person  
WHERE dogum_tarih<{01/01/70} AND  
Maas>=700 AND maas<=1200
```

Örnek:

BETWEEN..AND yapısıyla, maaşı 700 ile 1200 YTL arasında olanları gösteren T_SQL kodlarıdır.

```
SELECT *  
FROM Person  
WHERE maas BETWEEN 700 AND 1200
```

Örnek:

Doğum tarihi 1970'ten büyük ve cinsiyeti erkek olan veya doğum tarihi 1975'ten büyük ve cinsiyeti kadın olan personeli listeleyen T-SQL kodlarıdır.

```
SELECT *  
FROM Person  
WHERE dogum_tarih>={01/01/70} AND cinsiyet=Erkek  
OR dogum_tarih>={01/01/75} AND cinsiyet=Kadın
```

NOT: AND operatörü OR operatörüne göre daha önceliklidir.

Örnek:

Sınıf tablosunda adı Ali olmayanları listeleyen T-SQL kodlarıdır.

```
SELECT * FROM Sınıf  
WHERE NOT ad='Ali'
```

Örnek:

```
SELECT *  
FROM Öğrenci  
WHERE bölüm='Bilgisayar' OR bölüm='Elektronik' OR bölüm='Elektrik'
```

OR operatörü yerine In operatörü de kullanarak kod satırlarını yazabiliriz.

```
SELECT *  
FROM tablo1  
WHERE bölüm In ('Bilgisayar','Elektronik','Elektrik')
```

2.3.3. Aritmetiksel Operatörler

T-SQL'de kullanılan aritmetiksel operatörler şunlardır.

OPERATÖR	İŞLEVİ
%	Mod alma
*	Çarpma
/	Bölme
+	Toplama
-	Çıkarma

SELECT komutu ile veri tabanında mevcut tablolardan listeleme yaparken tabloda ayrı bir sütun (alan) olarak yer almamış ve ancak bir hesaplama sonucunda üretilebilecek bilgileri de listeleme içine almak mümkündür.

Örnek:

Ogr_Notlar tablosunda notların not ortalamasını hesaplayan T-SQL kodlarıdır.

```
SELECT (nt1+nt2+nt3) /3  
FROM Ogr_Notlar
```

Öncelik sırası, matematikte ve diğer bilgisayar dillerinde olduğu gibidir. Üs alma, hepsinden öncedir. Sonra çarpma (*) ve bölme (/) gelir. Toplama (+) ve çıkarma (-) en son önceliklidir. Parantez kullanılarak öncelik sırası değiştirilebilir.

2.4. Fonksiyonlar

2.4.1. Kümeleme Fonksiyonları

SQL, tablo içinden çeşitli matematiksel işlemlerin sonucunu otomatik olarak üretmeyi sağlayan fonksiyonlara sahiptir.

➤ **SUM Fonksiyonu**

Fonksiyonla belirtilen sütun ile ilişkili olarak toplama işlemini yapar.

Örnek:

Ogr_Notlar tablosunda nt1 sütununun not toplamını hesaplayan T-SQL kodlarıdır.

```
SELECT SUM(nt1)
FROM Ogr_Notlar
```

Örnek:

Maaşları 500 YTL'nin altında olan personelin maaşları toplamını hesaplayan T-SQL kodlarıdır.

```
SELECT SUM(maas)
FROM Person
WHERE maas<500
```

➤ **AVG Fonksiyonu**

Aritmetiksel ortalama (average) hesaplamak için kullanılır.

Örnek:

İşçilerin maaşları ortalamasını veren T-SQL kodlarıdır.

```
SELECT AVG(maas)
FROM Person
```

Bu fonksiyon ile de koşula bağlı olarak hesaplatma yaptırılabilir.

➤ **MAX Fonksiyonu**

Tablo içinde, belirtilen sütun (alan) içindeki en büyük değeri bulur.

Örnek:

Ogr_Notlar tablosu içindeki en yüksek öğrenci not ortalamasını veren T-SQL kodlarıdır.

```
SELECT MAX(not_ortalama)
FORM Ogr_Notlar
```

➤ MIN Fonksiyonu

Tablo içinde belirtilen sütun (alan) içindeki en küçük değeri bulur.

Örnek:

Ogr_Notlar tablosu içindeki en düşük öğrenci not ortalamasını veren T-SQL kodlarıdır.

```
SELECT MIN(not_ortalama)
FORM Ogr_Notlar
```

➤ COUNT Fonksiyonu

Tablo içerisinde herhangi bir sayma işlemi gerçekleştirmek için kullanılır.

Örnek:

Ogr_Notlar tablosu içindeki öğrenci not ortalamalarının sayısını veren T-SQL kodlarıdır.

```
SELECT COUNT (not_ortalama)
FROM Ogr_Notlar
```

COUNT fonksiyonu, DISTINCT sözcüğü ile de kullanılabilir. DISTINCT, her bir kaydın tekil olarak yer alması istendiğinde bu deyim kullanılır. Normalde SELECT ile aynı özelliğe sahip birden fazla kayıt listelenebilir.

Örnek:

Bir okulda birden fazla bölüme kayıtlı öğrenci olabilir. Sadece okuldaki bölüm adlarını listeleyen T-SQL kodlarıdır.

```
SELECT DISTINCT bolum
FROM Ogrenci
```

Örnek:

Bir okulda birden fazla bölüme kayıtlı öğrenci olabilir. Sadece okuldaki bölümlerin sayısını listeleyen T-SQL kodlarıdır.

```
SELECT COUNT (DISTINCT bolum)
FROM Ogrenci
```

COUNT komutunda, * karakterinin kullanılması, bütün sütunların işleme sokulmasını sağlar.

2.4.2. T-SQL'de Gruplandırma

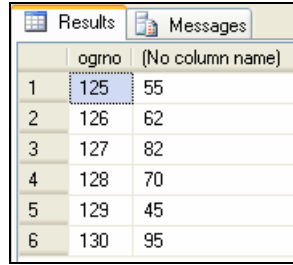
“GROUP BY” yardımcı sözcüğü bir alana göre kayıtları gruplamak için kullanılır. Genel kullanımını aşağıdaki gibidir:

```
SELECT [ DISTINCT | ALL ] <sütun(lar)> FROM <tablo adı (lar)>
[ WHERE <şart (lar)> ]
[ GROUP BY <sütunlar>]
```

Örnek:

Öğrenci notları tablosunda öğrenci numarasına göre her bir öğrencinin almış olduğu not ortalamalarını gösteren T-SQL kodlarıdır.

```
SELECT ogrno, AVG(ort)
FROM tablo2
GROUP BY ogrno
```



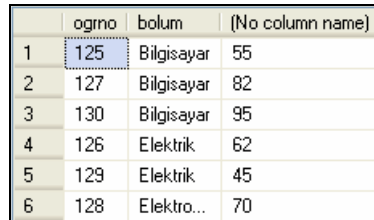
	ogrno	(No column name)
1	125	55
2	126	62
3	127	82
4	128	70
5	129	45
6	130	95

Resim 2.1: Group By kullanımı

Örnek:

Öğrenci notları tablosunda öğrenci numarasına ve bölüm adına göre her bir öğrencinin almış olduğu not ortalamalarını gösteren T-SQL kodlarıdır.

```
SELECT ogrno,bolum,MAX(ort)
FROM Tablo2
GROUP BY ogrno,bolum
```



	ogrno	bolum	(No column name)
1	125	Bilgisayar	55
2	127	Bilgisayar	82
3	130	Bilgisayar	95
4	126	Elektrik	62
5	129	Elektrik	45
6	128	Elektro...	70

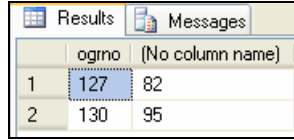
Resim 2.2: Group By kullanımının diğer görünümü

Gruplandırarak kümeleme fonksiyonlarını uygularken koşul da verilebilir. Bu durumda, grup üzerindeki hesaplamalarla ilişkili koşul belirtirken **HAVING** sözcüğünü kullanmak gerekir.

Örnek:

Öğrenci notları tablosunda öğrenci numarasına göre her bir öğrencinin almış olduğu not ortalamasını 70’den büyük gösteren T-SQL kodlarıdır.

```
SELECT ogrno,AVG(ort)
FROM Ogr_Notlar
GROUP BY ogrno
HAVING AVG(ort)>70
```



	ogrno	(No column name)
1	127	82
2	130	95

Resim 2.3: HAVING kullanımı sonucu ekran görüntüsü

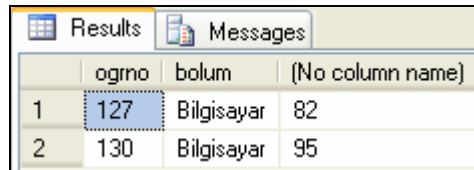
HAVING sözcüğü, SELECT komutunda GROUP BY sözcükleri bulunmadığı zaman geçersizdir. HAVING sözcüğünü izleyen ifade içinde SUM, COUNT (*), AVG, MAX ya da MIN gibi kümeleme fonksiyonlarından en az biri bulunmalıdır.

WHERE sözcüğü bir tablonun tek tek satırları üzerinde işlem yapan koşullar için geçerli iken HAVING sözcüğü ,sadece gruplanmış veriler üzerindeki işlemlerde geçerlidir.

Örnek:

Öğrenci notları tablosunda öğrenci numarasına göre bölümü “Bilgisayar” ve öğrencinin almış olduğu not ortalaması 65’ten büyük olanları gösteren T-SQL kodlarıdır.

```
SELECT ogrno,bolum,AVG(ort)
FROM Ogr_Notlar
WHERE bolum='Bilgisayar'
GROUP BY ogrno,bolum
HAVING AVG(ort)>65
```



	ogrno	bolum	(No column name)
1	127	Bilgisayar	82
2	130	Bilgisayar	95

Resim 2.4: Sorgunun ekran görüntüsü

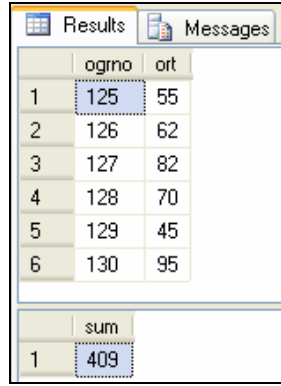
➤ COMPUTE ve COMPUTE BY Deyimleri

Toplama fonksiyonunu kullanarak sonuç olarak bir toplam değeri üretir. COMPUTE deyimi her SELECT ifadesiyle kullanılır. COMPUTE BY ise ORDER BY deyimine gerek duyar.

Örnek:

Öğrenci notları tablosundaki öğrenci numarasına göre ortalama sütunundaki değerleri toplayarak bir sonuç üreten T-SQL kodlarıdır.

```
SELECT ogrno, ort
FROM Ogr_Notlar
ORDER BY ogrno
COMPUTE SUM(ort)
```



	ogrno	ort
1	125	55
2	126	62
3	127	82
4	128	70
5	129	45
6	130	95
	sum	
1	409	

Resim 2.5: COMPUTE BY deyiminin kullanımının sonucu

2.4.3. Tarih ve Zaman Fonksiyonları

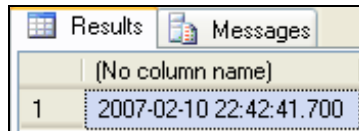
Tarih ve zaman üzerinde işlem yapmayı sağlayan fonksiyonlardır.

➤ GETDATE () Fonksiyonu

Şimdiki tarih ve saat değerini gösterir.

Örnek:

```
SELECT GETDATE()
```



	(No column name)
1	2007-02-10 22:42:41.700

Resim 2.6: Sistem tarih ve saatinin gösterilmesi

➤ DATEADD () Fonksiyonu

Verilen bir tarihe istenilen sayıda bir tarih bilgisi eklemek için kullanılır.

DATEADD fonksiyonunda

dd gün

mm ay

yy yıl anlamındadır.

Örnek:

Belirtilen tarih bilgisine 90 gün ekleyen T-SQL kodudur.

```
SELECT DATEADD(dd, 90, '02.10.2007')
```

	(No column name)
1	2007-05-11 00:00:00.000

Resim 2.7: Tarihe gün eklenmesi

```
SELECT DATEADD(mm, 2, '02.10.2007')
```

	(No column name)
1	2007-04-10 00:00:00.000

Resim 2.8: Tarihe 2 ay eklenmiştir.

```
SELECT DATEADD(yy, 2, '02.10.2007')
```

	(No column name)
1	2009-02-10 00:00:00.000

Resim 2.9: Tarihe 2 yıl eklenmiştir.

➤ DATEDIFF () Fonksiyonu

Belirtilen iki tarih arasındaki gün sayısını göstermektedir.

Örnek:

```
SELECT DATEDIFF(dd, '04.04.1974', '02.10.2007')
```

	(No column name)
1	12000

Resim 2.10: İki tarih arasındaki gün sayısının gösterimi

Ayrıca hafta için wk, saat için hh, dakika için mi, saniye için ss kullanılabilir.

➤ **DATEPART () Fonksiyonu**

Tarihle ilgili sayısal bilgilerin alınmasını sağlar.

Örnek:

```
SELECT DATEPART(dd, '01.04.1974')
```

```
SELECT DATEPART(mm, '01.04.1974')
```

```
SELECT DATEPART(yy, '01.04.1974')
```

2.4.4. Karakter Fonksiyonları

Karakter alanlarla ilgili işlem yapmak için bu fonksiyonlar kullanılır.

➤ **CHAR ()**

ASCII kodu verilen karakteri görüntüler.

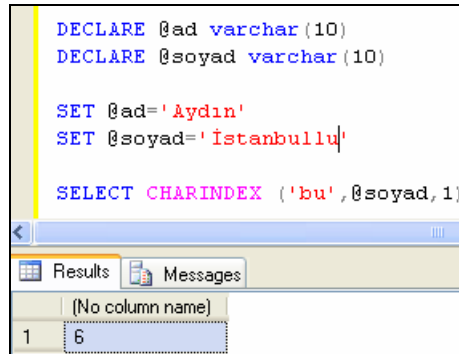
```
SELECT CHAR(65)           A harfini verir.
```

```
SELECT ASCII(A)           65 rakamını verir. Değer int tipindedir.
```

➤ **CHARINDEX()**

Bir metin içerisindeki metin parçasını istenilen konumdan itibaren arar.

```
SELECT CHARINDEX ('bu',@soyad,1)
```



```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İstanbulu'

SELECT CHARINDEX ('bu',@soyad,1)
```

Results	
(No column name)	
1	6

Resim 2.11: CHARINDEX () fonksiyonunun kullanımı

➤ **LEFT ()**

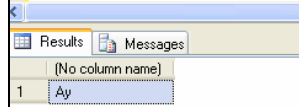
Metnin baş taraftan itibaren istenilen sayıdaki harflerini alır.

SELECT LEFT (@ad,2)

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İstanbulu'

SELECT LEFT (@ad,2)
```



(No column name)
1 Ay

Resim 2.12: LEFT () fonksiyonunun kullanımı

➤ **LEN ()**

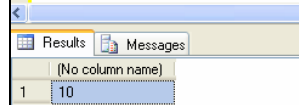
Metnin uzunluğunu veren fonksiyondur.

SELECT LEN(@soyad)

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İstanbulu'

SELECT LEN (@soyad)
```



(No column name)
1 10

Resim 2.13: LEN () fonksiyonunun kullanımı

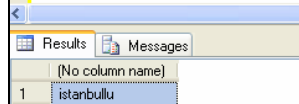
➤ **LOWER ()**

Metni küçük harfe çevirir.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İSTANBULLU'

SELECT LOWER (@soyad)
```



(No column name)
1 istanbullu

Resim 2.14: LOWER () fonksiyonunun kullanımı

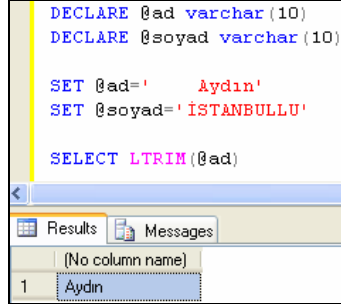
➤ **LTRIM ()**

Metnin başında bulunan boşlukları siler.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='   Aydın'
SET @soyad='İSTANBULLU'

SELECT LTRIM(@ad)
```



(No column name)
1 Aydın

Resim 2.15: LTRIM () fonksiyonunun kullanımı

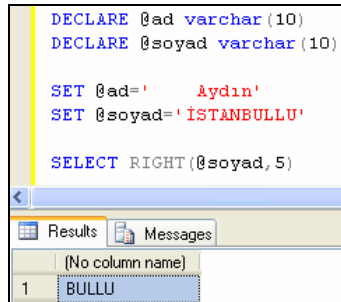
➤ **RIGHT ()**

Metnin sonundan istenilen kadar karakteri almak için kullanılır.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='   Aydın'
SET @soyad='İSTANBULLU'

SELECT RIGHT(@soyad, 5)
```



(No column name)
1 BULLU

Resim 2.16: RIGHT () fonksiyonunun kullanımı

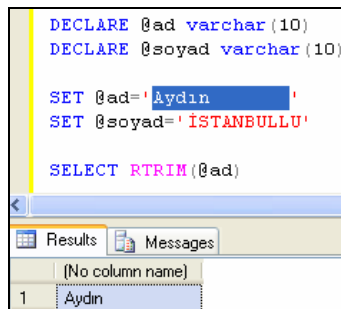
➤ **RTRIM ()**

Metnin sonundaki boşlukları siler.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın   '
SET @soyad='İSTANBULLU'

SELECT RTRIM(@ad)
```



(No column name)
1 Aydın

Resim 2.17: RTRIM () fonksiyonunun kullanımı

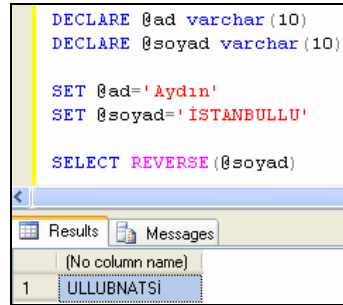
➤ **REVERSE ()**

Metni ters çevirir.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İSTANBULLU'

SELECT REVERSE (@soyad)
```



(No column name)
1 İLLUBNATSİ

Resim 2.18: REVERSE() fonksiyonunun kullanımı

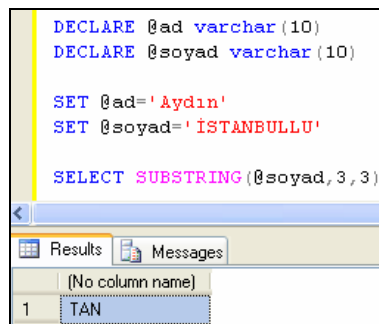
➤ **SUBSTRING ()**

Bir metinde belirtilen karakterden itibaren belirli sayıda karakter almak için kullanılır.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İSTANBULLU'

SELECT SUBSTRING (@soyad, 3, 3)
```



(No column name)
1 TAN

Resim 2.19: SUBSTRING () fonksiyonunun kullanımı

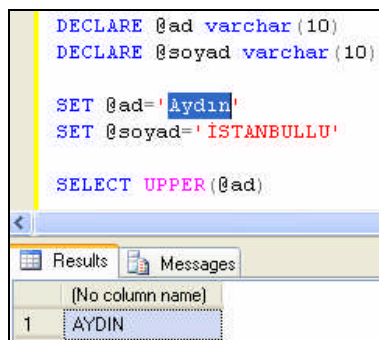
➤ **UPPER ()**

Metnin tümünü büyük harfe çevirir.

```
DECLARE @ad varchar (10)
DECLARE @soyad varchar (10)

SET @ad='Aydın'
SET @soyad='İSTANBULLU'

SELECT UPPER (@ad)
```



(No column name)
1 AYDIN

Resim 2.20: UPPER () fonksiyonunun kullanımı

2.5. SQL Denetim Deyimleri

Birden fazla komutu aynı anda işletebilmek için SQL’de bloklar kullanılmaktadır. Tüm programlama dillerinde olduğu gibi akış kontrollerinde ve döngü yapılarında kullanılan komutlar birden fazla ise mutlaka BEGIN..END bloğunda yazılmalıdır.

2.5.1. IF..ELSE Yapısı

Bir deyimın işletilmesini belli bir koşula bağlar.

➤ Kullanımı

IF koşul
{ deyim }
[ELSE
{ deyim }]

Örnek:

ogrno	bolum	nt1	nt2	nt3	ort
125	Bilgisayar	50	55	60	55
126	Elektrik	65	65	56	62
127	Bilgisayar	87	88	77	82
128	Elektronik	67	77	66	70
129	Elektrik	45	45	45	45
130	Bilgisayar	90	95	100	95

Resim 2.21: Öğrenci notları tablosu

Öğrencilerin not ortalamalarına göre ortalaması 85’in üzerinde olanların durumu için “PEKİYİ”, 85’ten küçük olanlar için ise “İYİ” yazdıran T-SQL kod satırlarıdır.

```
DECLARE @enbuyuk int
SELECT @enbuyuk=MAX(ort)
from tablo2

IF (@enbuyuk>=85)
BEGIN
    PRINT 'DURUMUNUZ PEKİYİ'
END
ELSE IF (@enbuyuk<84)
BEGIN
    PRINT 'DURUMUNUZ İYİ'
END
```

Messages
DURUMUNUZ PEKİYİ

Resim 2.22: IF..ELSE yapısının kullanılması ve sonucu

2.5.2. CASE Yapısı

Case yapısı, birçok durum için dallanmaya müsait bir yapıdır.

➤ Kullanımı

CASE değer
WHEN değer THEN işlem
WHEN değer THEN işlem
ELSE işlem
END

Örnek

Kütüphanedeki kitapların mevcut olup olmadığını gösteren basit bir T-SQL kodlarıdır.

Tablonun Resim 2.23'teki gibi olduğunu varsayınız.

KitapNo	Durum	KitapDurumu
1	1	
2	0	
3	1	
4	1	
5	0	
NULL	NULL	NULL

Resim 2.23: Kütüphane tablosu

Kitapların kütüphanede olup olmadığı KitapDurumu sütununda belirtilecektir.

```
SELECT KitapNo, Durum, 'KitapDurumu' =  
CASE  
WHEN Durum=0 THEN 'Yok'  
WHEN Durum=1 THEN 'Var'  
END  
FROM Kutuphane
```

KitapNo	Durum	KitapDurum
1	1	Var
2	0	Yok
3	1	Var
4	1	Var
5	0	Yok

Resim 2.24: Case yapısının kullanımı

Case yapısı kullanılarak kitap durumları KitapDurumu sütununa yazdırılmıştır.

2.5.3. WHILE Döngüsü

Tekrar gerektiren işlemlerde istenilen şart gerçekleşinceye kadar işlem yapmaya olanak sağlar. While ile bir işlemi istenilen kadar tekrarlayabilirsiniz. Genel yapısı şu şekildedir:

```
WHILE şart
BEGIN
    Tekrarlanması gereken kodlar
END
```

Şart gerçekleşinceye kadar BEGIN ile END arasına yazılan kodlar işlem görür.

Örnek:

Sayaç değişkeni 5 değerini alıncaya kadar tanımlanan toplam değişkenini beşer beşer arttıran programın T-SQL kod satırlarıdır.

```
DECLARE @toplaml int
DECLARE @sayac int

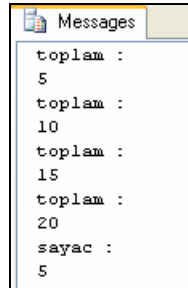
Select @toplaml=0
Select @sayac=1

WHILE (@sayac<5)
BEGIN
    SELECT @toplaml=@toplaml+5
    SELECT @sayac=@sayac+1
Print 'toplaml : '
Print @toplaml
END

Print 'sayac : '
Print @sayac
```

Resim 2.25: Kod satırları

Execute edilmesi sonucunda ekran görüntüsü Resim 2.26'daki gibi olur.



```
Messages
toplaml :
5
toplaml :
10
toplaml :
15
toplaml :
20
sayac :
5
```

Resim 2.26: Sonuç ekranı

➤ BREAK Komutu

İstenilen şart sağlandığında WHILE döngüsünden çıkmak için BREAK komutu kullanılır. Programın çalışması WHILE'in END'inin altındaki satırdan çalışmaya devam eder.

Örnek:

Sayacın 3 olması durumunda WHILE döngüsünden çıkıp toplamı ve sayacın değerini yazdıran T-SQL kodlarıdır.

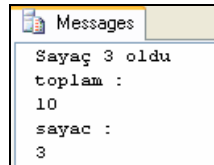
```
DECLARE @toplam int
DECLARE @sayac int

Select @toplam=0
Select @sayac=1

WHILE (@sayac<10)
BEGIN
    SELECT @toplam=@toplam+5
    SELECT @sayac=@sayac+1
    IF @sayac=3
    BEGIN
        PRINT 'Sayaç 3 oldu'
        BREAK
    END
END
Print 'toplam : '
Print @toplam
Print 'sayac : '
Print @sayac
```

Resim 2.27: Kod satırları

Execute edilmesi sonucunda ekran görüntüsü Resim 2.28'deki gibi olur.



Resim 2.28: Sonuç ekranı

➤ CONTINUE Komutu

Programın çalışmasını WHILE yapısının başına göndermek için kullanılan bir komuttur.

Örnek:

Sayaç değişkenin 3 değeri hariç diğer değerleri ve toplamı yazdıran T-SQL kodlarıdır.

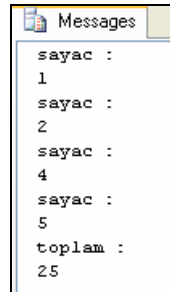
```
DECLARE @sayac int

Select @toplam=0
Select @sayac=0

WHILE (@sayac<5)
BEGIN
    SELECT @toplam=@toplam+5
    SELECT @sayac=@sayac+1
    IF @sayac=3 CONTINUE
    Print 'sayac : '
    Print @sayac
END
Print 'toplam : '
Print @toplam
```

Resim 2.29: Kod satırları

Execute edilmesi sonucunda ekran görüntüsü Resim 2.30'daki gibi olur.



```
Messages
sayac :
1
sayac :
2
sayac :
4
sayac :
5
toplam :
25
```

Resim 2.30: Sonuç ekranı

Sonuç ekranında 3 değerinin yazılmadığını görüyoruz.

UYGULAMA FAALİYETİ -1

İşlem Basamakları	Öneriler
<p>➤ Veri tabanınızda “Sınıf” adında bir tablo oluşturunuz.</p>	<p>➤ Tablonun sütun adları ve veri türleri Ogr_No int, Ad VARCHAR(15), Soyad VARCHAR(20), Cinsiyet VARCHAR(1), Yas int olarak belirleyebilirsiniz.</p> <pre>CREATE TABLE Sınıf(Ogr_No int, Ad VARCHAR(15), Soyad VARCHAR(20), Cinsiyet VARCHAR(1), Yas int)</pre> <p>Resim 2.31: Tablonun oluşturulması</p>
<p>➤ Tabloya beş adet kayıt girişi yapınız.</p>	<pre>INSERT INTO Sınıf (Ogr_No, Ad, Soyad, Cinsiyet, Yas) VALUES (101, 'Hasan', 'KAYA', 'E', 17) INSERT INTO Sınıf (Ogr_No, Ad, Soyad, Cinsiyet, Yas) VALUES (103, 'Veli', 'CAN', 'E', 16) INSERT INTO Sınıf (Ogr_No, Ad, Soyad, Cinsiyet, Yas) VALUES (104, 'Ayşe', 'KOCAER', 'K', 17) INSERT INTO Sınıf (Ogr_No, Ad, Soyad, Cinsiyet, Yas) VALUES (106, 'Fatma', 'YILMAZ', 'K', 18) INSERT INTO Sınıf (Ogr_No, Ad, Soyad, Cinsiyet, Yas) VALUES (109, 'Defne', 'YAĞMUR', 'K', 16)</pre> <p>Resim 2.32: Tabloya kayıtların eklenmesi</p>
<p>➤ Yaşı 17’den küçük olan öğrencileri seçiniz.</p>	<pre>SELECT * FROM Sınıf WHERE Yas < 17</pre> <p>Resim 2.33: Yaşı 17’den küçük öğrenciler</p>
<p>➤ Cinsiyeti erkek olmayan öğrencileri seçiniz.</p>	<pre>SELECT * FROM Sınıf WHERE Cinsiyet <> 'E'</pre> <p>Resim 2.34: Cinsiyeti E olmayan öğrenciler</p>
<p>➤ Soyadı “K” ile başlayan öğrencileri seçiniz.</p>	<pre>SELECT * FROM Sınıf WHERE Soyad LIKE '%K%'</pre> <p>Resim 2.35: Soyadı “K” ile başlayanlar</p>

<p>➤ Yaşı 17’den büyük kız öğrencileri seçiniz.</p>	<pre>SELECT * FROM Sinif WHERE Yas>17 AND Cinsiyet='K'</pre> <p>Resim 2.36: Yaşı 17’den büyük kız öğrenciler</p>
<p>➤ Adı “Ayşe” olmayanları seçiniz.</p>	<pre>SELECT * FROM Sinif WHERE NOT Ad='Ayşe'</pre> <p>Resim 2.37: Adı “Ayşe” olmayanlar</p>
<p>➤ Sınıf tablosunda yaş toplamlarını hesaplatınız.</p>	<pre>SELECT SUM(Yas) FROM Sinif</pre> <p>Resim 2.38: Yaş toplamları</p>
<p>➤ Yaşları 17 olan öğrencilerin yaş toplamlarını hesaplatınız.</p>	<pre>SELECT SUM(Yas) FROM Sinif WHERE Yas=17</pre> <p>Resim 2.39: Yaşı 17 olanların yaş toplamları</p>
<p>➤ Öğrencilerin yaş ortalamasını hesaplatınız.</p>	<pre>SELECT AVG(Yas) FROM Sinif</pre> <p>Resim 2.40: Yaş ortalamaları</p>
<p>➤ Öğrencilerin yaşları içinde en büyük yaş değerini seçiniz.</p>	<pre>SELECT MAX(Yas) FROM Sinif</pre> <p>Resim 2.41: En büyük yaş değeri</p>
<p>➤ Öğrenciler yaşları içinde en küçük yaş değerini seçiniz.</p>	<p>➤ SELECT MIN(Yas) FROM Sinif</p>
<p>➤ Tabloda kaç öğrenci olduğunu hesaplatınız.</p>	<pre>SELECT COUNT(*) FROM Sinif</pre> <p>Resim 2.42: Öğrenci sayısının bulunması</p>
<p>➤ Sadece öğrenci numaralarını seçiniz.</p>	<pre>SELECT DISTINCT Ogr_No FROM Sinif</pre> <p>Resim 2.43: Sadece öğrenci numaralarının seçimi</p>

<p>➤ Sadece öğrenci numaralarını ve yaşlarını seçiniz.</p>	<pre>SELECT Ogr_No, Yas FROM Sinif GROUP BY Ogr_No, Yas</pre> <p>Resim 2.44: Sadece öğrenci numaraları ve yaşların seçimi</p>
<p>➤ Yaşı 17'den küçük olanları, ad ve soyadlarıyla beraber seçiniz.</p>	<pre>SELECT Ad, Soyad, Yas FROM Sinif GROUP BY Ad, Soyad, Yas HAVING Yas < 17</pre> <p>Resim 2.45: Ad ve soyadlarıyla yaşı 17'den küçük olanların seçimi</p>
<p>➤ Yaşı 16'dan büyük olanları, öğrenci numaraları ve cinsiyetleriyle beraber seçiniz.</p>	<pre>SELECT Ogr_No, Cinsiyet FROM Sinif GROUP BY Ogr_No, Cinsiyet, Yas HAVING Yas > 16</pre> <p>Resim 2.46: Öğrenci numaraları ve cinsiyetleri yaşı 16'dan büyük olanların seçimi</p>

UYGULAMA FAALİYETİ - 2

İşlem Basamakları	Öneriler
➤ Sistem tarih ve saatini görüntüleyiniz.	➤ GETDATE() fonksiyonunu kullanabilirsiniz.
➤ Sistem tarihine 120 gün ekleyiniz.	➤ DATEADD () ve GETDATE()fonksiyonlarını aynı anda kullanabilirsiniz.
➤ Sistem tarihine 10 ay ekleyiniz.	➤ DATEADD () ve GETDATE ()fonksiyonlarını aynı anda kullanabilirsiniz.
➤ Sistem tarihine 1 yıl ekleyiniz.	➤ DATEADD () ve GETDATE()fonksiyonlarını aynı anda kullanabilirsiniz.
➤ 29 Ekim 1923 ile günümüz arasında kaç gün olduğunu bulunuz.	➤ DATEDIFF () ve GETDATE()fonksiyonlarını aynı anda kullanabilirsiniz.
➤ Karakterel türde ve uzunlukları 50 olan üç değişken tanımlayınız.	➤ DECLARE komutunu kullanabilirsiniz. Değişken adları olarak da @kelime1, @kelime2 ve @kelime3'ü tanımlayabilirsiniz.
➤ @kelime1 değişkenine “Pamuk” değerini, @kelime2 değişkenine “Kale” değişkenini atayınız.	➤ SET deyimini kullanabilirsiniz.
➤ @kelime1 ve @kelime2 değişkenlerinin içeriğini @kelime3 değişkenine SET deyimini kullanarak atayınız.	➤ SET deyimiyile + operatörünü kullanabilirsiniz.
➤ Değişkenlerin içeriğini mesaj penceresine yazdırınız.	➤ PRINT komutunu kullanabilirsiniz.
➤ @kelime1 ve @kelime2 değişkenlerinin içeriğini birleşik olarak mesaj penceresine yazdırınız.	➤ PRINT komutuyla + operatörünü kullanabilirsiniz.
➤ @kelime3 değişkeninin ilk dört karakterini seçiniz.	➤ LEFT() fonksiyonunu kullanabilirsiniz.
➤ Yeni bir değişken tanımlayarak @kelime3 değişkeninin ilk dört karakterini bu değişkene atayınız.	➤ SET deyimini ve LEFT () fonksiyonunu aynı anda kullanabilirsiniz.
➤ Elde ettiğiniz bu değeri mesaj penceresine yazdırınız.	➤ PRINT komutunu kullanabilirsiniz.
➤ @kelime3 değişkeninin içeriğinin uzunluğunu bulunuz.	➤ LEN() fonksiyonunu kullanabilirsiniz.
➤ Tüm değişkenlerin içeriğini büyük harfe çeviriniz.	➤ UPPER () fonksiyonunu kullanabilirsiniz.
➤ Yeni bir değişken tanımlayarak	➤ SET deyimini ve RIGHT()fonksiyonunu

@kelime3 deęişkeninin saęından dört karakter seçiniz ve bu deęişkene atayınız.	kullanabilirsiniz.
➤ Yeni deęişkenin tüm karakterlerini küçük harfe çeviriniz.	➤ LOWER () fonksiyonunu kullanabilirsiniz.
➤ @kelime3 deęişkeninin altıncı karakterinden başlayarak 3 karakter seçiniz.	➤ SUBSTRING () fonksiyonunu kullanabilirsiniz.

UYGULAMA FAALİYETİ - 3

İşlem Basamakları	Öneriler
<ul style="list-style-type: none"> ➤ “int” veri türünde iki tane değişken tanımlayınız. 	<ul style="list-style-type: none"> ➤ DECLARE ifadesini kullanabilirsiniz.
<ul style="list-style-type: none"> ➤ Bu iki değişkene iki tam sayı atayınız. 	<ul style="list-style-type: none"> ➤ SET ifadesini kullanabilirsiniz.
<ul style="list-style-type: none"> ➤ Eğer birinci sayı ikinci sayıdan büyükse mesaj penceresine “Birinci sayı ikinci sayıdan büyüktür.” cümlesini yazdırınız. 	<ul style="list-style-type: none"> ➤ IF yapısını kullanabilirsiniz. <pre> DECLARE @sayi1 int DECLARE @sayi2 int SET @sayi1=7 SET @sayi2=5 IF @sayi1>@sayi2 BEGIN PRINT 'Birinci sayı ikinci sayıdan büyüktür' END </pre> <p style="text-align: center;">Resim 2.47: If yapısı</p>
<ul style="list-style-type: none"> ➤ Birinci sayı ikinci sayıdan büyük olmayabilir. Bunun ELSE bloğuyla diğer durumu da göz önüne alarak “Birinci sayı ikinci sayıdan küçüktür.” cümlesini yazdırınız. 	<pre> ELSE PRINT 'İkinci sayı birinci sayıdan büyüktür' </pre> <p style="text-align: center;">Resim 2.48: ELSE bloğu</p>
<ul style="list-style-type: none"> ➤ Değer atanan iki değişkenin toplamını yeni bir değişken tanımlayarak toplatınız. 	<pre> DECLARE @Toplam int SET @Toplam=@sayi1+@sayi2 </pre> <p style="text-align: center;">Resim 2.49: İki sayının toplamı</p>
<ul style="list-style-type: none"> ➤ Eğer iki sayının toplamı 5’ten büyükse “İki sayının toplamı 5’ ten büyüktür.” mesajını yazdırınız. 	<pre> IF @Toplam>5 PRINT 'İki sayının toplamı 5 ten büyük' </pre> <p style="text-align: center;">Resim 2.50: Toplamın karşılaştırılması</p>
<ul style="list-style-type: none"> ➤ İki sayının toplamı 5’ten küçük olabilir. Bu yüzden oluşabilecek diğer durumu da kontrol ettiriniz. 	<pre> ELSE PRINT 'İki sayının toplamı 5 ten küçük' </pre> <p style="text-align: center;">Resim 2.51: Diğer durum kontrolü</p>

<p>➤ İki sayının toplamının 9'dan büyük olması durumunda toplam değişkenini 10 ile çarpan, 10'dan küçük olması durumunda 2 ile çarpan CASE yapısını yazınız.</p>	<pre>SELECT @Toplam= CASE WHEN @Toplam>9 THEN @Toplam*10 WHEN @Toplam<10 THEN @Toplam*2 END</pre> <p>Resim 2.52: CASE yapısı</p>
<p>➤ CASE işlemi sonucunu mesaj olarak görüntüleyiniz.</p>	<pre>PRINT 'CASE işlemi sonucu : ' PRINT @Toplam</pre> <p>Resim 2.53: İşlemin sonucu</p>
<p>➤ “int” veri türünde sayac ve modtoplam adında iki değişken tanımlayınız ve ilk değerlerini 0 (sıfır) olarak atayınız.</p>	<pre>DECLARE @sayac int DECLARE @modtoplam int SET @sayac=0 SET @modtoplam=0</pre> <p>Resim 2.54: Değişken tanımlama ve ilk değer atama</p>
<p>➤ “sayac” değişkeni yukarıda hesapladığınız “toplam” değişkeninden küçük olduğu sürece “sayac” değişkeninin 2'ye göre modunu toplayan WHILE döngüsünü yazınız. “sayac” değişkenini beşer beşer artırınız.</p>	<pre>WHILE (@sayac<@toplam) BEGIN SET @sayac=@sayac+5 SET @modtoplam=@modtoplam+ (@sayac%2) END</pre> <p>Resim 2.55: While döngüsü kullanımı</p>
<p>➤ Elde ettiğiniz mod alma işleminin sonucunu, yani “modtoplam” değişkeninin sonucunu mesaj olarak yazdırınız.</p>	<pre>PRINT 'Mod işlemi toplamları sonucu=' PRINT @modtoplam</pre> <p>Resim 2.56: Mod işlemi sonucu</p>
<p>➤ SELECT 'Mod işlemi toplamları sonucu='+ CONVERT(VARCHAR, @modtoplam) şeklinde de yazılabilir. CONVERT() fonksiyonu veriyi bir formattan başka bir formata çevirmek için kullanılır. Böylece “int” türdeki “@modtoplam” değişkeninin içeriği karakter sel veri türüne çevrilmiştir.</p>	

Sorgunun tümü aşağıda verilmiştir:

```
DECLARE @sayi1 int
DECLARE @sayi2 int

SET @sayi1=3
SET @sayi2=6

IF @sayi1>@sayi2
BEGIN
    PRINT 'Birinci sayı ikinci sayıdan büyüktür'
END
ELSE
    PRINT 'İkinci sayı birinci sayıdan büyüktür'

DECLARE @Toplam int
SET @Toplam=@sayi1+@sayi2

IF @Toplam>5
    PRINT 'İki sayının toplamı 5 ten büyük'
ELSE
    PRINT 'İki sayının toplamı 5 ten küçük'

SELECT @Toplam=
CASE
    WHEN @Toplam>9 THEN @Toplam*10
    WHEN @Toplam<10 THEN @Toplam*2
END

PRINT 'CASE işlemi sonucu :'
PRINT @Toplam

DECLARE @sayac int
DECLARE @modtoplam int

SET @sayac=0
SET @modtoplam=0

WHILE (@sayac<@toplam)
BEGIN
    SET @sayac=@sayac+5
    SET @modtoplam=@modtoplam+(@sayac%2)
END

PRINT 'Mod işlemi toplamları sonucu='
PRINT @modtoplam
```

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşluk doldurmalı sorularda boşluklara uygun cevapları yazınız.

1. Birden fazla komutu aynı anda işletebilmek için bloğu kullanılır.
2. Tablodaki kayıtların sayısını bulmak için fonksiyonu kullanılır.
3. T-SQL'de hangi veri tabanıyla çalıştığınızı belirtmek için komutu kullanılır.
4. Verileri bellekte geçici olarak saklayan değerlere..... denir.
5. Metni küçük harfe çevirmek için LOWER(), büyük harfe çevirmek için UPPER() fonksiyonu kullanılır (D/Y).
6. Bir karakter sel bilgide arama yaparken aşağıdaki operatörlerden hangisi kullanılır?
A) * B) + C) % D) &
7. Aşağıdakilerden hangisi değişkenlere değer atama ifadelerinden değildir?
A) SET B) SELECT C) INSERT INTO D) DECLARE
8. Aşağıdakilerden hangisi <> operatörü yerine geçen mantıksal operatördür?
A) AND B) OR C) NOT D) BETWEEN
9. Sorgular önce ayrıştırılır daha sonra çalıştırılır. (D/Y)
10. Şart sağlandığında döngüden çıkmak için komutu kullanılır.
11. Metnin solundaki boşlukları atmak için LEFT(), sağındaki boşlukları atmak için RIGHT() fonksiyonu kullanılır. (D/Y)
12. T-SQL'de değişken tanımlarken komutu ve değişkenin önüne ön eki yazılır.
13. DEBUG işlemi için ihtiyaç duyulan ve sonuçları mesaj penceresinde yazan komut'tir.
14. Aşağıdakilerden hangisi belirtilen iki tarih arasındaki gün sayısını elde etmek için kullanılan tarih fonksiyonudur?
A) GETDATE() B) DATEADD()
C) DATEDIFF() D) DATEPART()

15. Metni ters çevirmek için aşağıdaki fonksiyonlardan hangisi kullanılır?
A) REVERSE () B) TURN () C) ROTATE () D) TRANSLATE ()
16. Sorguların sırayla işleme tabi tutulmasına yığıt denir. (D/Y)
17. Aritmetiksel ortalama almak için AVG() fonksiyonu kullanılır .(D/Y)
18. Bir deyimın işletilmesini bir koşula bağlayan yapı yapısıdır.
19. Tablodaki kayıtları gruplamak için GROUP BY, koşul vermek için HAVING ifadeleri kullanılır. (D/Y)
20. Belirtilen metnin uzunluğunu bulmak için fonksiyonu kullanılır.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 4 puan arasında olacak şekilde değerlendirecektir.

Değerlendirme Ölçütleri	Puan
1. Veri tabanı oluşturabilme	
2. Sorguyu çalıştırabilme	
3. Tablo oluşturabilme	
4. Sütunları veri türleriyle beraber tanımlayabilme	
5. Sütun ekleyebilme	
6. Tabloya bilgi girebilme	
7. Herhangi bir bilgiyi güncelleyebilme	
8. Bilgileri artan veya azalan sırada görüntüleyip seçebilme	
9. Login oluşturabilme	
10. User oluşturabilme	
11. Kısıtlama verebilme	
12. Tabloda sorgu yapabilme	
13. Sütunlar üzerinde matematiksel fonksiyonları kullanabilme	
14. Gruplandırma ve koşula göre gruplandırma yapabilme	
15. Tarih-saat fonksiyonlarını kullanabilme	
16. Değişken tanımlayabilme	
17. Değişkenlere değer atayabilme	
18. Karakterel fonksiyonları kullanabilme	
19. IF-ELSE ve CASE yapısını kullanabilme	
20. WHILE döngüsünü kullanabilme	
Toplam (100 puan olabilir.)	

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	B
2	Yanlış
3	FOR LOGIN
4	C
5	B
6	Veri Kontrol Dili
7	Doğru
8	Login
9	D
10	Yanlış

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	BEGIN-END
2	COUNT()
3	USE
4	Değişken
5	Doğru
6	C
7	D
8	C
9	Doğru
10	BREAK
11	Yanlış
12	DECLARE ve @
13	PRINT
14	C
15	A
16	Yanlış
17	Doğru
18	IF-ELSE
19	Doğru
20	LEN()

KAYNAKÇA

- GÖZÜDELİ Yaşar, “**Yazılımcılar için SQL Server 2005 ve Veri Tabanı Programlama**”, Seçkin Yayıncılık, Ankara, 2006.
- www.verivizyon.com
- www.sqlnedir.com