SQL Eğitim Kitabı

Murat YÜCEDAĞ

Değerli okurlarım kitabımıza başlamadan önce bir ricam olacak.

Kitapta hata gördüğünüzde hemen kızmayın, sizlerin de desteğiyle hatalarımızı ve eksikliklerimizi tarafıma bildirmeniz durumunda bir sonraki baskıda düzeltilecektir. :)

Murat YÜCEDAĞ

muratyucedag.com

facebook.com/murattyucedag

twitter.com/murattyucedag

linkedin.com/in/murat-yücedağ-186933149

udemy.com/user/murat-yucedag-3

youtube.com/user/YazilimHerYerde

github.com/MuratYucedag

yucedagmurat23@gmail.com

İçindekiler

1

Veritabanı Kavramları

Bu Bölümde

Bu bölümde;

Veritabanı kavramını, veritabanının ne olduğunu, nerelerde kullanıldığını, veri ile bilgi arasındaki farkı, nereden veritabanına ihtiyacımız olduğunu, veritabanı kavramları olan; tablo, satır, sütun gibi ifadelerin ne işe yaradığını ve veritabanı kullanmanın avantajlarını, SQL kavramını, SQL dilini kullanan veritabanı yönetim sistemlerinin neler olduğunu ve SQL server Managment Studio programının indirme ve kurulum işleminin nasıl yapılacağını öğreneceğiz.

Giriş

Veritabanı sadece bir programlama konusu olarak değil, hayatın her alanında karşımıza çıkan bir başlıktır. Kullandığınız bütün elektronik aletlerin az ya da çok, küçük ya da büyük mutlaka bir veritabanı bulunuyor. Veri için "modern çağın petrolü" tanımlaması yapılıyor. Gerçekten de öyle. Ancak artık sadece veriye sahip olmak yetmiyor, veriyi işlemek, onu anlamlı bütünler haline getirmek de en az veriye sahip olmak kadar önemli olmaya başladı. Bu kitapta amacım sizlerle beraber bolca sorgu yazıp SQL'i olabildiğince anlamlı hale getirebilmektir. O halde veritabanı kavramıyla girişimizi yapalım.

Veritabanı Nedir?

Yapılandırılmış bilgi ve verilerin depolandığı alanlardır. Veritabanı kavramının oluşması için önce veriye sonra da bu verilerin depolanacağı bir ortama ihtiyacımız var. Yukarıda söylediğimiz gibi; aslında aklınıza gelen bütün elektronik aletlerin az ya da çok mutlaka bir veritabanı alanları bulunmaktadır.

Örneğin; yıllar önce kullandığınız şu anda pek tercih edilmeyen tuşlu telefonlarda, mevcutta kullandığınız bilgisayarlarda, televizyonlarda, buzdolaplarında ve diğer tüm cihazlarda bir veritabanı alanı bulunuyor. Veritabanında amaç veriyi saklayabilmek ve ihtiyaçlar doğrultusunda kullanabilmektir. Hangi programlama dili veya teknoloji ile çalışırsanız çalışın mutlaka yolunuz veritabanları ile kesişecektir. Yani siz bir oyun geliştirici de olsanız, bir mobil programcı da olsanız veya bir web developer da olsanız mutlaka veritabanı kullanacaksınız. Bundan dolayı bir veritabanı dilini bilmek her halükarda hayat kurtaracaktır. Veritabanının oluşması için veri ve bilgi kavramlarına ihtiyaç vardır. Öyleyse veri ve bilgi kavramlarını ele alalım.

Veri ve Bilgi Nedir?

Veri için işlenmemiş bilgi kavramına verilen addır şeklinde bir tanımlama yapabiliriz. Veriler tek başlarına anlamlı şeyler ifade etmezler. Verilerin anlamlı hale gelmesi için işlenmeleri gerekir. Öyleyse bilgi için: "verinin işlenip anlamlı hale getirilmesi" şeklinde bir tanımlama yapabiliriz.

Veri: Elimde bir şey var ama ne olduğunu bilmiyorum.

Bilgi: Elimde bir adet portakal var.

Anlama: Portakal bir meyvedir.

Bilgelik: Turuncu portakallar tatlıdır.

Veri, bilgi, anlama ve bilgelik kavramları bu şekilde örneklendirilebilir. Veri ile bilgi arasındaki ilişkiyi maddeler halinde sıralayacak olursak;

» Veri hamdır işlenmemiştir, bilgi verilerin işlenmiş halidir.

» Veri rakamlar, sayılar, harfler gibi ifadelerdir, bilgi ise bunların işlenen anlamlı bütünler halidir.

» Veri bilgiden bağımsızdır, bilgi ise veriye muhtaçtır.

» Bilgi olmadan veri olabilir ama veri olmadan bilgi olmaz.

» Veri tek başına anlamsızdır, bilgi ise tek başına anlam ifade eder.

» Veri hammaddedir, bilgi ise o hammaddenin ürünüdür.

VERİTABANI KAVRAMLARI

Veritabanlarının anlamlı birer bütün haline gelebilmesi için beraberinde kullanılan bazı kavramlar bulunmaktadır. Bu kavramları tek tek ele alalım.

1. Tablolar

Veritabanlarımızın içerisinde yer alacak verilere ait kayıtlarının tutulduğu yerlerdir. Tabloların anlamlı bütünler haline gelmesi için satır ve sütun kavramlarına, veritabanlarının anlamlı bütün haline gelmesi için ise tablolara ihtiyaç vardır. Ticari bir işletme düşünelim. Bu işletme beyaz eşya ürünleri satan bir mağaza olsun. Bu mağazada yapılan satışların, kalan ürün stoklarının, kasada bulunan toplam ücretin ne kadar olduğunun bulunması gibi durumlar için bir ticari otomasyon uygulamasına ihtiyaç vardır. Ticari otomasyonlar mağazalarda yapılan işlemlerin bilgisayar ortamında tutulmasını sağlayan programlardır. Mağazada kullanılan ticari otomasyon programında yer alması gereken temel menüler şunlardır;

» Ürün bilgileri

» Müşteri bilgileri

» Satış bilgileri

» Personel bilgileri

» Kasa bilgileri

» Fatura bilgileri

» Gider bilgileri şeklinde arttırılabilir.

Burada saymış olduğumuz her bir madde bizim veritabanımıza ait tablodur. Bu tabloların bir araya gelerek oluşturduğu yapı ise veritabanımız olacaktır. Tabloların oluşturulabilmesi için sütunlara ihtiyaç vardır. O halde satır ve sütun kavramlarını ele alalım.

2

Veritabanı, Tablo Oluşturma ve Veri Türleri

Bu Bölümde

Bu bölümde;

SQL'e nasıl bağlantı sağlanacağını, veritabanına bağlandıktan sonra karşımıza gelen penceredeki temel başlıkları, yeni bir veritabanının nasıl oluşturulacağını ve oluşturulan veritabanına nasıl tablo ekleneceğini göreceğiz.

Tablo ekranındaki design kısmını inceleyeceğiz. Programlama dillerinde değişken ne ise SQL'de veri türü odur.

Ayrıca, bu bölümde programlama dillerindeki değişkenlerin SQL'deki karşılığıyla veri türlerinin neler olduğunu, bunların arasındaki farkları, en sık kullanacağımız veri türlerini ve SQL tablo design penceresinde yer alan veri türlerinin tamamının neler olduğunu öğreneceğiz.

GİRİŞ

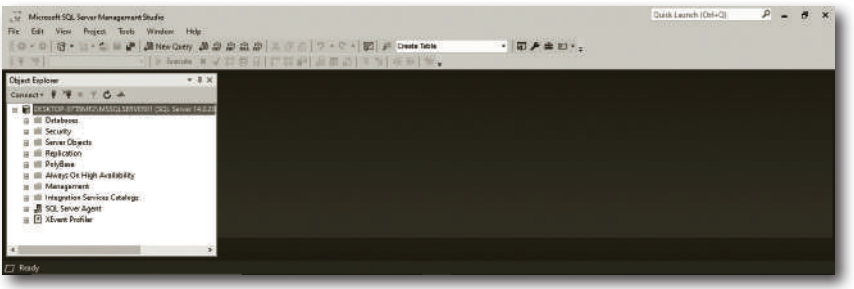
Geçtiğimiz bölümde kurulumunu tamamladığımız Managment Studio programımızı artık çalıştırabiliriz. **Başlat** menüsünden **SQL Server Managment Studio** programını açarak ilk adımı atalım. SQL'i açtığımız zaman bizi aşağıdaki gibi bir ekran karşılayacaktır.

Bu ekranda yer alan başlıkları tek tek açıklayalım.

Server Type: Sunucu tipi anlamına gelmektedir. Bizim kitap boyunca çalışacağımız sunucu türü database engine yani Veritabanı motoru olacaktır.

Server Name: Sunucu adı anlamına gelen bu kısımda veritabanımızın local'de yani yerelde çalışacak olan sunucu ismini görmekteyiz.

Authentication: Yetkilendirme anlamına gelen bu kısım SQL'de yetkinin kim tarafından ele alındığını ifade eder, uzaktaki bir SQL sunucusuna bağlanmak için bu seçeneğin altındaki değerlerden ilgili birimler seçilebilir. Biz kitap boyunca kendi local sunucumuzda çalışacağımız için Windows yetkilendirmesini kullanacağız. Eğer farklı bir SQL üzerinde işlem yapacaksak o zaman bu seçeneğin hemen altında bulunan kullanıcı adı ve şifre kısımlarından ilgili sunucunun bilgileri girilebilir. **Connect** seçeneğine tıklayarak bağlantımızı gerçekleştirelim.

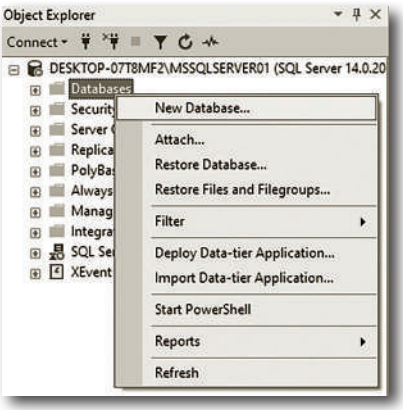


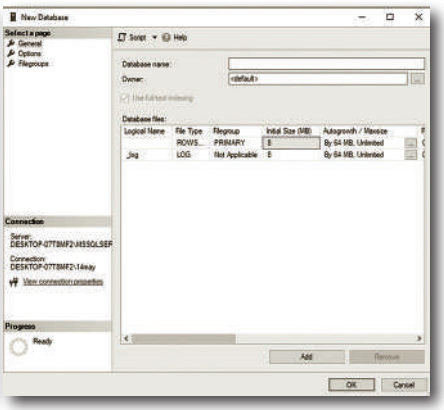
SQL sunucumuza bağlantımızı gerçekleştirdikten sonra karşımıza yukarıda bulunan görseldeki gibi bir ekran gelecektir. Burası **Object Explorer** yani nesne gezgini penceresidir. Buradaki klasörler üzerinde en çok işlem yapacağımız klasör **databases** klasörü olacaktır. Yeni bir Veritabanı oluşturmadan önce ne üzerine ve hangi başlıkları içerecek bir Veritabanı oluşturacağımıza karar verelim. Veritabanı olarak müşterilere yapılan satışların kayıtlarının tutulduğu, ürün stoklarının takip edildiği, müşteri raporlarının hazırlandığı bir veritabanı oluşturacağız. Veritabanında verilerimizi **Türkçe** olarak hazırlasak da veritabanı ve tablo isimlerini tamamen İngilizce olarak tutacağız. Böylece daha profesyonel bir çalışma yapmış olacağız.

Yeni Veritabanı Oluşturma

Yeni bir veritabanı oluşturmanın birden fazla yöntemi bulunmaktadır. İlk yöntem olarak sağ tuş menüleri üzerinden işlem yaparak başlayacağız. Databases isimli klasörümüzün üzerinde sağ tuşla tıklayıp **New Database** yani Veritabanı seçeneğini seçerek başlayalım.

Daha sonra karşımıza aşağıdaki ikinci resimde göründüğü gibi bir pencere ekranı gelecektir.





3

Tabloların Oluşturulması

Bu Bölümde

Bu bölümde veritabanıı ve tablo oluşturma işleminden sonra bu tablonun nasıl oluşturulacağını, tabloya nasıl sütun ekleneceğini, örnek bir tablo oluşturmayı, tabloda en doğru veri türlerinin nasıl seçileceğini, SQL new table yani yeni tablo ekranında yer alan menülerin neler olduğunu öğreneceğiz.

Ek olarak sorgular üzerinden nasıl tablo oluşturacağımızı öğreneceğiz. Yeni sorgu ekranından tamamen sorgularla nasıl tablo oluşturacağımızı, DDL kavramının ne olduğunu, Create, Alter ve Drop komutlarını, bu komutların DML komutları ile olan farklarını, örnek bir müşteri tablosunu, oluşturduğumuz tabloda kısıtlayıcılar kullanmayı, oluşturduğumuz tabloda sütunlara default yani varsayılan değer ataması yapmayı, oluşturulan bir tablonun sorgu üzerinden nasıl güncelleneceğini ve nasıl silineceğini öğreneceğiz.

TABLOLAR

Veritabanımıza hazırlık yaparken tablolarımızı oluşturmaya başlayabiliriz. Tablo oluşturmanın da birden fazla yöntemi var. Bu farklı yöntemleri ilerleyen bölümlerde sizlerle paylaşacağım. İlk olarak yine sağ tuş menüsü ile başlayacağız ancak tablolarımızı oluşturmadan önce oluşturacağımız tabloların neler olduğuna bakalım.

Product: ürün tablosu olarak kullanacağımız tablomuzda ürünlerin id değeri, adı, stok sayısı, alış fiyatı ve satış fiyatı gibi bilgilerini tutacağız.

Category: kategori tablosu olarak kullanacağımız tablomuzda kategorilerin id ve isimlerini tutacağız. Bunların dışında daha pek çok tablomuz olacak ve diğer tabloları oluşturmadan önce ele alıyor olacağız.

KATEGORİ TABLOSU

Tablolarımıza kategori tablosu ile başlayalım. Kategori tablomuzun ismini ve sütun adlarını tamamen İngilizce olarak adlandıralım. Bu da bir kural değil ama daha profesyonel bir yaklaşım olarak adlandırılabilir. **Tables** kısmına sağ tuşla tıklayıp **New** > **Table...** diyerek ilk adımı atalım.

Sonraki adımda karşımıza aşağıda yandaki gibi bir ekran gelecektir.

Gördüğümüz üzere 3 Adet alan yer almaktadır.

Column Name: bu kısım sütun adı olarak adlandırılmaktadır. Kategori tablomuzda 2 tane sütun kullanacağız. Bunlar;

» Category ID

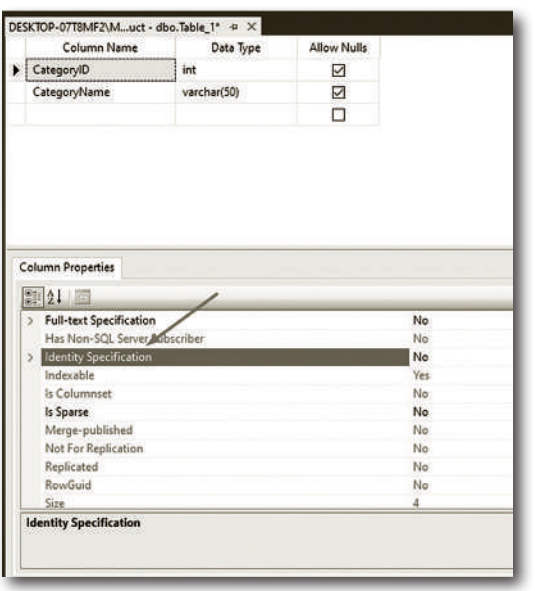
» Category Name

ID: Tanımlayacağımız neredeyse bütün tablolarda mutlaka bir ID alanı yer alacaktır. ID alanı ile amacımız ilgili satıra ait benzersiz bir sayısal değer tanımlamaktır. Mesela üniversitelerde her öğrenciye bir öğrenci numarası verilir. Bu numara benzersizdir. Bu öğrenci mezun olduktan sonra aynı numara başka bir öğrenciye verilmez. Bizim ID alanımızda buna benzer bir görev üstlenecektir. ID kelimesi İngilizcedeki Identity yani kimlik kelimesinin kısaltmasıdır.

Data Type: Bu alan veri türü olarak kullanılacak alandır. Tıpkı programlama dillerinde kullanılan değişkenler gibidir. Nasıl ki bir programlama dilinde int, string, char gibi değişken türleri varsa aynı şekilde SQL'de oluşturulan tablolara ait sütunlarda da birer veri tipi olmak zorundadır. Veri tipi olmadan sütun tanımlaması yapılamaz.

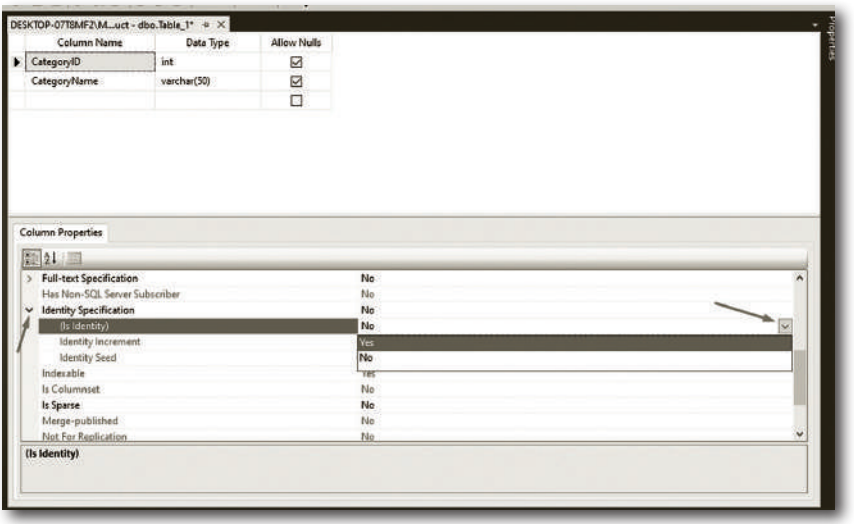
Allow Nulls: Bu alan ilgili sütuna ait değerin boş geçilip geçilemeyeceğinin belirlendiği alandır. Mesela bir ürün tanımladınız ve başlangıçta bunun fiyatını bilmiyorsunuz. Böyle bir durumda ürünün fiyat değerini boş bırakmak isteyebiliriz. Hata vermemesi adına allow nulls yani null değere izin verilmesi seçeneği aktif hale getirilebilir.

Kategori tablomuzda 2 tane sütun yer alacaktır. Bunlardan ilki kategori adını temsil edecek olan **Category Name**, diğeri ise kategorilerin ID değerini tutacak olan Category ID olacaktır. Kategori ID sütununu int olarak belirleyelim çünkü ID değeri sayısal bir alan olacak. Category Name ise değişken uzunlukta karakterden oluşacak olan bir alandır. Mesela Beyaz Eşya kategorisi isim uzunluğu olarak aradaki boşlukla beraber 10 karakterden oluşur. Bilgisayar kategorisi de 10 karakterden oluşuyor. Küçük Ev Aletleri kategorisi ise aradaki boşluklarla beraber 17 karakterden oluşuyor. Karakter uzunlukları farklı olduğu için kullanabileceğimiz en doğru veri tipi varchar olacaktır. Eğer tablomuzda Latin Alfabesi dışındaki karakterler de olacaksa o zaman nvarchar türü tercih edilmelidir.



Kategori adı için 50 karakterlik alan ayırmak yeterli olacaktır. Daha optimal bir tür olmasını isterseniz boyutu biraz daha azaltabilirsiniz.

Category ID alanını otomatik artan hale getirmemiz lazım. ID alanı ile amacımız ilgili satırların benzersiz bir kimlik değerine sahip olmasıdır. Burada ID değerlerini tek tek elle girmek yerine otomatik artan haline getirip benzersiz olmasını sağlayabiliriz. Otomatik artanın aktif hale getirilebilmesi için ilgili sütunun veri türünün tam sayı olması gerekiyor. Int veri türü şu anda bizim bu isteğimize olumlu karşılık verecektir.



CategoryID alanı seçiliyken alt kısımda yer alan **Column Properties** yani sütun özellikleri penceresini açalım. Burada sütunla ilgili birtakım özellikler yer almaktadır. Zamanı geldikçe ihtiyaca göre bu özellikleri kullanacağız. Bizim üzerinde çalışacağımız alan **Identity Specification** yani kimlik özelleştirme alanıdır. **Specification** alanının sol tarafında yer alan ikona tıklayalım ve kimlik özelleştirmesi alanını aktif hale getirelim. Burada **IsIdentity** alanının sağ tarafında **No** yazmaktadır. Bunu **Yes** olarak değiştirelim. Burası Yes olduğu zaman bu kimliğin özelleştirme için hazır hale geldiğini programa bildirmiş olacağız.

4

Veri Girişleri

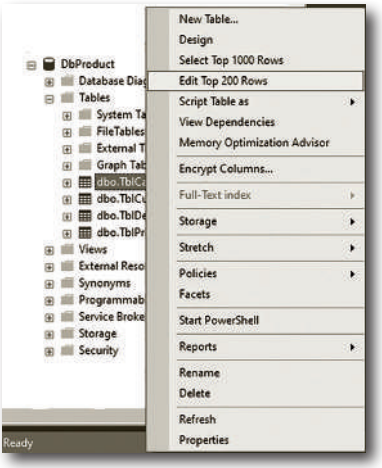
Bu Bölümde

SQL denilince akla hep veritabanı geliyordu, veritabanı deyince de akla veri seti gelmelidir. Çünkü veri olursa veritabanı oluşur.

İşte bu bölümde biz örnek veri girişleri yapıp nasıl veri seti oluşturacağımızı, birden fazla tabloya nasıl veri girişi yapacağımızı, bunlara ek olarak sipariş tablomuzu DDL komutlarıyla Create metodunu kullanarak nasıl oluşturacağımızı göreceğiz.

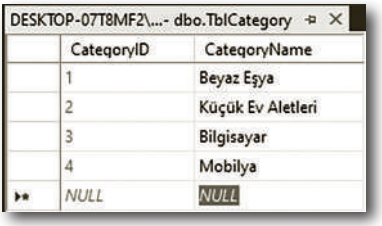
VERİ GİRİŞİ

Kitabımızın bu bölümünde manuel olarak veri girişi yapacağız. Veri girişi yapmanın 2 temel yöntemi var, birisi manuel olarak yapmak diğeri sorgularla yapmaktır. Biz ilk olarak manuel veri girişi gerçekleştireceğiz. Veri girişlerine kategori tablomuzla başlayalım. **TblCategory** tablosu üstüne sağ tuşla tıklayıp **Edit Top 200 Rows** alanını seçelim.



Karşımıza aşağıdaki gibi bir ekran gelecektir.

Burada dikkat ettiyseniz ID alanı pasif olarak geliyor. Yani ID alanına veri girişi yapamazsınız. Çünkü bu kısmı otomatik artan yaptığımız için kendisi direkt eklenecektir. Biz **Category Name** alanına aşağıdaki gibi veri girişleri yapalım.



Kategori tablomuzda 4 tane değer girişi yaptık. Tabloya veri girişi yaptıktan sonra **CTRL + S** veya benzeri bir kaydetme işlemine gerek yok zira veri girişi yapılıp her alt satıra inme işleminden sonra otomatik olarak kayıt gerçekleşecektir.

Ürün Tablosuna Örnek Veri Girişleri

Kategori tablomuza veri girişi yaptıktan sonra şimdi de ürün tablomuza veri girişi yapalım. Ürün tablomuza **Edit Top 200 Rows** kısmına sağ tuş menüsü üzerinden erişim sağlayabiliyoruz. İlgili alanı açalım ve aşağıdaki gibi veri girişlerini yapalım.

DESKTOP-07T8MF2\M...- dbo.TblProduct | DESKTOP-07T8MF2\...- dbo.TblCategory

ProductID	ProductName	ProductStock	PurchasePrice	SalePrice
1	Buzdolabı	100	3500,00	3800,00
2	Çamaşır Makinesi	100	2000,00	2200,00
3	Su Isıtıcı	250	300,00	340,00
4	Laptop	50	12000,00	12850,00
5	Fırın	70	450,00	500,00
6	Ütü	150	600,00	650,00
7	Mikser	20	600,00	700,00
8	Çalışma Masası	50	230,00	280,00
9	Bulaşık Makinesi	80	1500,00	1750,00
10	Kulaklık	300	50,00	60,00
NULL	NULL	NULL	NULL	NULL

Ürün tablomuzda da veri girişlerini tamamlamış olduk. Bir tabloda ne kadar çok veri varsa verinin işlenmesi sonucunda o kadar verimli sonuçlar elde edilir. Bundan dolayı veri seti hazırlarken olabildiğince fazla veri girişi yapmaya özen gösterin. Müşteri tablomuza veri girişi yapalım.

DESKTOP-07T8MF2\...- dbo.TblCustomer

CustomerID	CustomerName	CustomerSurn...	CustomerCity	CustomerBala...
1	Ali	Çınar	Adana	30000,00
2	Ahmet	Sarı	Trabzon	25000,00
3	Ayşegül	Yıldız	Bursa	28000,00
4	Eylül	Özbey	Bursa	14000,00
5	Yılmaz	Kara	Adana	36000,00
6	Veysel	Taşkaya	Antalya	22000,00
7	Hakan	Öztürk	Mardin	20000,00
8	Ecenur	Sarı	Trabzon	8000,00
9	Tuğçe	Meşeli	Eskişehir	17000,00
10	Zeynep	Bulut	Afyon	36000,00
NULL	NULL	NULL	NULL	NULL

Müşteri tablomuza 10 tane veri girişi yaptık. Projemizde şu an için 3 temel tablo bulunuyor. Deneme tablosunu silebiliriz.

Sorgu 1: Deneme tablomuzu DDL komutları ile silen sorguyu yazalım.

```
Drop Table TblDeneme
```

Sorgumuzu çalıştırdığımız zaman deneme tablosu veritabanımızdan silinecektir. Sıradaki konumuz DML komutları olacak ama araya bir konu daha ekleme ihtiyacı hissettim. Onlardan ilki hareket tablosunun oluşturulması. Hareket veya işlemler tablosuyla amacımız müşterilerimizin yaptıkları alım işlemlerinin veritabanıına kaydedilmesidir.

Örneğin; Ali Çınar isimli müşterimiz x tarihinde y adet ürünü z TL'ye satın aldı. Bunları veritabanından işlemler tablomuza eklememiz gerekiyor. Bunun için DDL üzerinden işlemler ya da daha doğru isimle siparişler tablomuzu oluşturalım. Tablomuza işlemler kelimesinin İngilizce karşılığı olan `Transaction` ifadesini vermemiz pek sağlıklı olmaz zira Transaction kelimesi SQL'de kullanılan özel bir komut içindir. Bizde bunun yerine sipariş anlamına gelen Order kelimesini kullanabiliriz.

Sorgu 2 Siparişler tablomuzu oluşturalım.

```
Create Table TblOrder
(
OrderID int identity (1,1),
Customer varchar(50),
Product varchar(50),
OrderCount tinyint,
Price decimal(18,2),
TotalPrice decimal (18,2),
OrderDate Smalldatetime
)
```

Sipariş tablomuzda ID, müşteri adı, ürün adı, sipariş edilen ürün sayısı, ürün fiyatı, toplam fiyat ve tarih bilgileri tutulacaktır. Şu anda hazırladığımız tablo pek doğru şekilde hazırlanmadı. Eksiklerimiz var ancak bunları revize edeceğiz. Kitabımızda ilişkilere geçmeden önce DML komutlarına yer vermek istiyorum. DML komutları iyice oturduktan sonra ilişkilere başlayacağız.

5

Select ve Where

Bu Bölümde

Kitabımızın bu bölümünde artık SQL'in esas konuları olan DML komutlarını, Select işleminin nasıl yapıldığını, sorguları kullanarak tabloları okumayı, DML komutlarının neler olduğunu, Select ifadesinde bütün tablo verilerinin nasıl getirileceğini, yine Select ifadesinde tabloya ait sadece belli sütunları getirmeyi, verilen şarta göre listelemeyi, where komutu kullanımını, like işlecini, not like işlecini, belli bir harfe veya karaktere göre arama yapmayı, sayısal ve alfabetik kısıtlamaları öğreneceğiz. Bunlara ek olarak And, Or ve Not operatörlerinin neler olduğunu, bunların sorgularla nasıl kullanılacağını ve kullanılması gerektiğini öğreniyor olacağız.

DML NEDİR?

Bir önceki bölümde yaptığımız işlemlerin tamamı tabloları, bu tablolar içinde bulunan sütunları veya bizzat veritabanımızı etkiliyordu. Yaptığımız işlemlerin hiçbirinde verilerimize müdahale olmadı. **Data Manipulation Language** yani **Veri İşleme Dili** kelimelerinin baş harflerinden oluşan **DML** komutları herhangi bir tabloya yeni veri eklemek, var olan verileri silmek, mevcut veriler üzerinde güncelleme yapmak ya da ilgili veriler içinde sadece istenen nitelikleri sağlayan verileri listelemek amacıyla kullanılan SQL komutlarıdır.

DML komutları kitabımızın en kapsamlı bölümünü oluşturacaktır. 4 tane DML komutu vardır. Bunlar;

» Select

» Insert

» Update

» Delete komutlarıdır.

Kullandığınız bütün sosyal medya uygulamalarında, web sitelerinde, haber sitelerinde, bloglarda, bankacılık işlemlerinde, öğrenci not sistemlerinde ve daha aklımıza gelmeyen binlerce belki de milyonlarca platformda DML komutları kullanılmaktadır. Şimdi bu komutları tek tek ele alalım.

SELECT

Seç, listele anlamına gelen **Select** komutu mevcut bir veritabanındaki verileri listelemek için kullanılır. Select ifadesinde diğer SQL ifadelerinde olduğu gibi büyük küçük harf duyarlılığı yoktur, yani ister büyük ister küçük yazalım her 2 durumda da sorgumuz çalışacaktır. `Select` komutunun yazım kuralı şu şekildedir;

Select | sütun adları | from tablo adı

Sorgu 1: Ürün tablomuzda bulunan ürünlerin tamamını listeleyen sorguyu yazalım.

```
Select * From TblProduct
```

	ProductID	ProductName	ProductStock	PurchasePrice	SalePrice
1	1	Buzdolabı	100	3500.00	3800.00
2	2	Çamaşır Makinesi	100	2000.00	2200.00
3	3	Su Isıtıcı	250	300.00	340.00
4	4	Laptop	50	12000.00	12850.00
5	5	Fırın	70	450.00	500.00
6	6	Ütü	150	600.00	650.00
7	7	Mikser	20	600.00	700.00
8	8	Çalışma Masası	50	230.00	280.00
9	9	Bulaşık Makinesi	80	1500.00	1750.00
10	10	Kulaklık	300	50.00	60.00

Sorgumuzu yazıp çalıştırdığımız zaman karşımıza yukarıdaki gibi bir çıktı ekranı gelecektir. `Select` ifadesi bir tabloya ait verileri listelemek için kullanılan komuttur. * Sembolü İngilizce'de **all** anlamı taşıyan Türkçeye **"tümü, hepsi"** olarak çevireceğimiz anlama sahip görev üstlenir.

Yani biz bir tabloya ait verileri listelerken sadece belirli sütunları listeleyebileceğimiz gibi aynı zamanda tüm sütunlara ait değerleri de getirebiliriz. Tüm sütunlara ait değerleri getirebilmek için * sembolünden faydalanılır. `From` komutu İngilizcede bir şeyin nereye/nereden olduğunu bildirmek için kullanılır. Yani burada **–den –dan** görevini üstlenir. Son olarak verilerin listeleneceği tablo ismi yazılır ve sorgu çalıştırılır.

Sorgu 2: Ürün tablosuna ait verileri sadece ürün adı ve stok sayısı şeklinde listeleyen sorguyu yazalım.

```
Select ProductName,ProductStock From TblProduct
```

	ProductName	ProductStock
1	Buzdolabı	100
2	Çamaşır Makinesi	100
3	Su Isıtıcı	250
4	Laptop	50
5	Fırın	70
6	Ütü	150
7	Mikser	20
8	Çalışma Masası	50
9	Bulaşık Makinesi	80
10	Kulaklık	300

8

Aggregate, Alt Sorgular, Gruplandırmalar ve Sıralamalar

Bu Bölümde

Bu bölümde aggregate fonksiyonlarının neler olduğunu, nerelerde kullanılacağını, Count, Sum, Avg, Min ve Max metotlarının nasıl kullanılacağını öğreneceğiz.

Bunlara ek olarak sütunlara nasıl takma isim verileceğini, bir sorgudaki kayıtları tekrarsız olarak nasıl getireceğimizi, bir tablodaki kayıt sayısını, bir tabloda yer alan sayısal değerlerin toplamını, bir tablodaki kayıtlar içinde en yüksek ve en düşük oranlı değerleri, bir tablodaki sayısal alanda ortalama değeri hesaplamayı, alt sorguların ne olduğunu, neden alt sorguya ihtiyacımız olduğunu, alt sorguların Select ile kullanımını ve alt sorguların Update ile kullanımını öğreneceğiz.

Son olarak alt sorguların aggregate fonksiyonları ile nasıl kullanılacağını, gruplandırma kavramını, gruplandırmanın nasıl kullanılacağını, gruplandırmaya nerelerde ihtiyaç duyacağımızı, Having komutunu, Sıralandırma işlemlerinin nasıl yapılacağını, **A > Z** ve **Z > A** formatın sıralamaları öğreneceğiz.

AGGREGATE NEDİR?

SQL konuları arasında inanılmaz büyük bir yeri olan, istatistiksel işlemlerin ve bu bölümden sonra göreceğimiz gruplama konusunun vazgeçilmez kavramlarından biri olan Aggregate ya da Türkçe karşılığı ile toplama / kümeleme fonksiyonlarından en önemlileri olan Count, Sum, Avg, Min ve Max komutlarını bu bölümde uygulamaları ile birlikte ele alacağız. Kullanmış olduğumuz ürün, müşteri ve kategori tablolarımızda yalnızca ekleme, silme, güncelleme gibi temel yapılar değil, bunun dışında;

» Bir tablodaki toplam kayıt sayısı
» Sadece istenen nitelikteki kayıt sayısı
» Bir ürüne ait toplam stok sayısı
» Bir kategoriye ait toplam stok sayısı
» Bir tabloda bulunan en yüksek fiyatlı ürün
» Bir tabloda bulunan en düşük fiyatlı ürün
» Bir tabloda bulunan ürünlerin ortalama fiyatları

ve daha onlarca başlıkta kullanabiliriz.

Şimdi bu fonksiyonlarımızı tek tek ele alalım.

COUNT FONKSİYONU

Count fonksiyonu bir tabloda istenen nitelikteki değerin kaç adet olduğu sonucunu verir. Count komutu Select ifadesi ile beraber kullanılır. Kullanımını örnekler üzerinde görelim.

Sorgu 1: Ürün tablomuzdaki toplam ürün sayısını bulalım.

```
Select Count(*) From TblProduct
```

	(No column name)
1	11

Select ifadesinden sonra Count fonksiyonumuzu çağırdık. Aggregate fonksiyonlarımızın tamamında parantez içerisine bir parametre değeri göndermemiz gerekiyor. Bizim burada saydırmak istediğimiz değer tablodaki bütün kayıtlar olduğu için * sembolünü kullandık. Daha sonra tablo adımızı yazarak sorgumuzu tamamladık.

Sorgu 2: Ürün tablomuzda stok sayısı 100'den fazla olan ürün adedini bulalım.

```
Select Count(*) From TblProduct where ProductStock>100
```

	(No column name)
1	3

Ürün tablomuzda ürün stok değeri 100'den büyük olan ürünlerin sayısını Count fonksiyonu ile bulduk.

Sorgu 3: Ürün adı içerisinde a harfi geçen ürünlerin sayısını bulan sorguyu yazalım.

```
Select Count(*) From TblProduct where ProductName like '%a%'
```

	(No column name)
1	6

Daha önceki bölümlerde ürün adı içinde a harfi geçen ürünleri listelemiştik. Bu kez içinde a harfi geçen ürünlerin sayısını bulduk.

SUM FONKSİYONU

Diyelim ki tablomuzda ürünlerimizin sayısını değil de toplam stok sayısını görmek istiyoruz. Mesela buzdolabı 20 tane, ütü 15 tane ve bilgisayar 25 tane olsun, burada count sorgusu yazarsak sonuç olarak bize 3 cevabı dönecektir. Bizim burada amacımız ürünlerimizin çeşitlilik türünde sayısını bulmak değil stoklarımızın toplam değeri olan 20 + 15 + 25 işleminin sonucunu bulmak. İşte bu işlem için sum fonksiyonunu kullanacağız. Tıpkı count ifadesinde olduğu gibi saymasını istediğimiz değeri parantez içerisine parametre değeri olarak göndereceğiz.

Sorgu 4: Ürün tablomuzdaki ürünlerin toplam stok değerlerini bulalım.

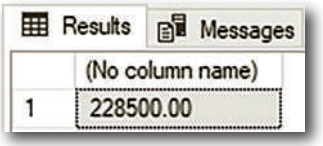
```
Select Sum(ProductStock) From TblProduct
```

	(No column name)
1	1225

Sum fonksiyonunu kullanmak için toplama işlemi yapılacak olan sütun Sum komutundan sonraki parantezin içine yazılır böylece sorgumuzu çalıştırdık.

Sorgu 5: Müşteri tablomuzdaki müşterilerimizin toplam bakiyesini hesaplayan sorguyu yazalım.

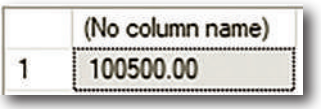
```
Select Sum(CustomerBalance) From TblCustomer
```



Müşteri tablomuzdaki bakiyelerin toplamını bu şekilde ekranda listelemiş olduk.

Sorgu 6: Müşteri tablomuzda yer alan müşteriler içinde şehri Adana veya Bursa olan şehirdeki müşterilerin toplam bakiyesini hesaplayan sorguyu yazalım.

```
Select Sum(CustomerBalance) From TblCustomer where CustomerCity='adana' or
CustomerCity='bursa'
```



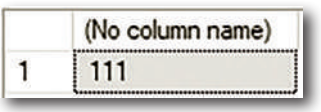
Müşteri tablomuzdaki müşteriler içinde Sum fonksiyonunu kullanarak şehri Adana veya Bursa olan müşterilerimizin bakiye toplamını bulmuş olduk.

AVG FONKSİYONU

Count fonksiyonu ile bir tabloda bulunan ürün sayımızı verirken, Sum fonksiyonu ile bu tabloda bulunan herhangi bir aritmetik sütun değerlerinin toplamını vermektedir. AVG ise istenen sütunların ortalama değerini verir. Bunu ilgili sütunların değerini toplayıp topladığı kayıt sayısına böler. Örneğin; ürün tablomuzdaki stok sayımızın ortalama değerini bulmak istediğimizde bunu sum ve count değerlerinin birbirine bölümü ile elde edebiliriz. Bunun yerine çok daha pratik bir yöntem olarak AVG komutumuzu kullanacağız.

Sorgu 7: Ürün tablomuzdaki ürünlerin ortalama stok sayısını hesaplayan sorguyu yazalım.

```
Select Avg(ProductStock) From TblProduct
```

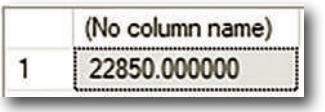


Avg komutu da tıpkı Count ve Sum gibi kullanılmaktadır. Sorgumuzu çalıştırdığımız zaman ürün tablomuzda bir ürün için ortalama stok değerini bulduk. Yani

burada SQL bize stok sayılarını toplayıp Count değerine bölerek sonucu hesaplamış oldu.

Sorgu 8: Şehri Adana ve Bursa olmayan müşterilerin ortalama bakiyesini hesaplayan sorguyu yazalım.

```
Select Avg(CustomerBalance) From TblCustomer where CustomerCity!='adana' or
CustomerCity!='Bursa'
```



	(No column name)
1	22850.000000

Müşteri tablomuzda ortalama bakiye değerini arıyoruz ancak şehir bilgisi Adana veya Bursa olan müşterileri bu hesaplamaya dahil etmek istemiyoruz. Bu sebeple CustomerCity yani müşteri şehir bilgisi Adana veya Bursa olan müşterileri bu işlemden muaf tutmak için eşit değilse anlamına gelen != sembollerini kullandık.

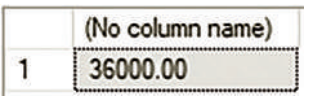
MAX FONKSİYONU

Bir tabloda bulunan aritmetik değerler içerisinde ilgili sütuna ait en yüksek değeri veren fonksiyondur. Örnek sorgu üzerinde kullanımını görelim.

Sorgu 9: Müşteri tablomuzdaki en yüksek bakiye değerini veren sorguyu yazalım.

```
Select Max(CustomerBalance) From TblCustomer
```

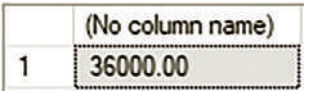
Max komutu veya min komutu da tıpkı diğer 3 aggregate fonksiyonları gibidir. Maksimum yani en yüksek değerini bulmak istediğimiz sütunu max içerisine yazıp sorgumuzu çalıştırdık.



	(No column name)
1	36000.00

Sorgu 10: Şehri Adana, Ankara veya Bursa olan şehirler içinde en yüksek bakiye bilgisini veren sorguyu yazalım.

```
Select Max(CustomerBalance) From TblCustomer where CustomerCity
In('Adana','Bursa','Ankara')
```



	(No column name)
1	36000.00

Sorgumuzda birden fazla or ifadesi kullanmak yerine in komutu ile bir parantez içerisinde ilgili şehirleri yazıp bu şehirler içinde en yüksek bakiye bilgisini getirmiş olduk. En yüksek bakiyenin hangi şehre ait olduğunu bulmak istersek burada devreye alt sorgular girecektir. Bu konuyu bir sonraki bölümde ele alacağız.

10

Metinsel Fonksiyonlar

Bu Bölümde

Öğrenmesi oldukça keyifli olan bu bölümde özellikle alfabetik yapılar üzerinde metinsel fonksiyonları nasıl kullanacağımı, bu fonksiyonların neler olduğunu öğreneceğiz. Char Index, Concat, Lower, Upper, Reverse gibi birçok metinsel fonksiyonu uygulamalı olarak hem kelime hem de tablolardaki sütun bazında nasıl kullanacağımızı öğreneceğiz.

GİRİŞ

SQL'de kullandığımız sorgularda birtakım hazır fonksiyonlar sayesinde uzun uzadıya yapacağımız işlemleri tek kelimelik kısa işlemlerle çözebiliriz. Örneğin, istediğimiz herhangi bir sütuna ait değeri tersinden yazdırabilir, sol veya sağ tarafından istediğimiz kadar karakteri silebilir, atlayabilir veya ilgili sütun değerlerini tamamen büyük veya küçük harfe dönüştürmek isteyebiliriz. İşte bu ve benzeri daha pek çok işlemi SQL'in bize sağlamış olduğu metinsel fonksiyonlar sayesinde kolaylıkla yapabiliriz. Kitabımızın bu bölümünde kullanacağımız metinsel fonksiyonları tek tek ele alalım.

ASCII

Girilen karakterin ASCII kod tablosundaki karşılığını verir, eğer tek karakter yerine kelime yani birden fazla harf veya karakteri barındıran bir ifade girilirse bu ifadedeki ilk karakterin ASCII karşılığını verir.

Sorgu 1: Girilen herhangi bir harfin Ascii kod tablosundaki karşılığını veren sorguyu yazalım.

```
Select Ascii('A')
```

	(No column name)
1	65

Metinsel fonksiyonları genelde `Select` komutuyla beraber kullanacağız. Ascii karşılığını bulmak istediğimiz karakteri ascii metodu içinde çağırarak sorgumuzu çalışırdık.

CHAR

ASCII komutunda girilen karakterin sayısal karşılığını alıyorduk. `Char` fonksionunda ise ASCII kodu girilen karakterin hangi karakter olduğunu ekrana yazar.

Sorgu 2: 75 değerinin char karşılığını bulan sorguyu yazalım.

```
Select Char(75)
```

	(No column name)
1	K

Ascii kod karşılığını girdiğimiz verinin karakter karşılığını göstermiş olduk.

CHAR INDEX

Verilen 2 parametreli değerde girilen ilk parametrenin 2. parametrede kaçıncı indexten itibaren başladığını verir.

Sorgu 3: Sorgu ekranımıza 1.parametre olarak merhaba, 2.parametre olarak ise günaydın merhaba nasılsınız yazıp merhaba kelimesinin kaçıncı karakterden itibaren başladığını bulalım.

```
Select Charindex ('merhaba','günaydın merhaba nasılsınız')
```

	(No column name)
1	10

Görüldüğü gibi merhaba kelimesinin 2. parametrede başlangıç karakteri 10'dur. Günaydın kelimesi 8 harften oluşur arada bir boşluk vardır boşluk 9. karakter oluyor sonrasında merhaba kelimesi 10. karakterden itibaren başlıyor.

Bu 2 parametreye ek olarak Char Index fonksiyonunda 3. parametreyi de alabilir. Bu parametre arama işleminin kaçıncı index ya da karakterde başlayacağının belirlendiği parametre değeridir. Örnek üzerinde daha net bir şekilde görelim.

Sorgu 4: Sorgu ekranımıza Mustafa Kemal Atatürk'ün "Dinlenmemek üzere yürümeye karar verenler asla yorulmazlar" sözünde "asla" kelimesinin 10. karakterden itibaren kaçıncı karakter olduğunu bulan sorguyu yazalım.

```
SELECT CHARINDEX ('Asla','Dinlenmemek üzere yürümeye karar verenler asla yorulmazlar',10)
```

	(No column name)
1	43

Burada asla kelimesi 43. index değeri olarak karşımıza çıktı. Peki, cümlemizin içine bir tane daha asla kelimesi ekleyelim ve sorgumuzu yeniden çalıştıralım.

Sorgu 5: Sorgu cümlemize bir tane daha asla kelimesi ekleyip 3.parametremizi yine 10 olarak tutalım.

```
SELECT CHARINDEX ('Asla','Dinlenmemek asla üzere yürümeye karar verenler asla yorulmazlar',10)
```

	(No column name)
1	13

Görüldüğü gibi sorgumuzda asla kelimesiyle yani 2. Parametrede gönderdiğimiz kelime ile ilk karşılaştığı kısmın CHAR INDEX değerini vermiş oldu. Son olarak aynı cümlede 3. parametremizin değerini 15 yapalım.

Sorgu 6: Sorgu cümlemize bir tane daha asla kelimesi ekleyip 3.parametremizi 15 olarak tutalım.

```
SELECT CHARINDEX ('Asla','Dinlenmemek asla üzere yürümeye karar verenler
asla yorulmazlar',15)
```

	(No column name)
1	48

3. parametre değerimiz arama işlemine 15. indexten itibaren başladı. Bundan dolayı boşlukla beraber 13. indexte yer alan ilk asla kelimesini atlamış oldu. 2.asla kelimesiyle cümlemizin baştan 48. karakterinde karşılaştık. Yani 3. parametre değerinde baştaki kelime ve harfleri atlasak bile aradığımız Charindex değerinin sonucu yine cümlenin başından itibaren sayılıyor.

CONCAT

Girilen parametreleri birleştiren fonksiyondur.

Sorgu 7: Sorgu ekranımıza girdiğimiz parametreleri concat fonksiyonu ile birleştirelim.

```
Select Concat('küçük ','hanımlar ','küçük ','beyler ')
```

	(No column name)
1	küçük hanımlar küçük beyler

`Concat` fonksiyonu parantez içerisinde parametre olarak birbirinden ayrı girdiğimiz kelimeleri bir cümle olarak birleştirdi. Cümle tamamen birleşik olmasın diye kelimelerden sonra bir karakter boşluk bıraktık.

CONCAT_WS

Yapı olarak `concat` fonksiyonuna benzeyen `concat_ws` fonksiyonu concat'tan farklı olarak girilen parametreleri birleştirmek yerine bu parametreler arasına girilen ilk parametreyi ekler.

Sorgu 8: Sorgu ekranımıza girdiğimiz parametreleri concat_ws fonksiyonu ile birleştirip ilk parametre değeri olarak – sembolü koyalım.

```
Select Concat_ws('-','Türk','Öğün','Çalış','Güven')
```

	(No column name)
1	Türk-Öğün-Çalış-Güven

`Concat_ws` komutu ile ilk parametre olarak girdiğimiz tire sembolünü diğer parametrelerimizin arasına eklemiş olduk.

12

Tarih Zaman Fonksiyonları

Bu Bölümde

Kitabımızın bu bölümünde tarih zaman fonksiyonlarına değineceğiz. Bundan önceki bölümlerde metinsel ve aritmetik fonksiyonları öğrendik. Tarih zaman fonksiyonları özellikle şartlı listelemelerde çok sık kullanılan bir yaklaşımdır. Tarih zaman fonksiyonları aracılığıyla 2 tarih arasındaki farkı, DatePart, DateDiff, DataAdd, DateName gibi komutların hem tekil hem de tablolara balı sütunlar üzerinde nasıl kullanılacağını göreceğiz.

GİRİŞ

Tarih zaman fonksiyonları veritabanı sistemlerinin vazgeçilmez bir parçasıdır. Yapılan bir ürünün satışının ne zaman gerçekleştiği, bir öğrencinin kütüphaneden herhangi bir kitabı ne zaman emanet aldığı ve ne zaman iade etmesi gerektiği, bir üyenin herhangi bir web sitesinde bir bloğa ait yorumu ne zaman yaptığı gibi çeşitli durumlarda tarih zaman fonksiyonları kullanılmaktadır. Tarih zaman için kullanacağımız pek çok fonksiyon olacak. Bu fonksiyonlara geçmeden önce örnek sorgularımız için yeni bir veritabanı oluşturalım. Veritabanımızı küçük bir kitaplık sistemi üzerine kuralım. Bu veritabanımızın 3 tane tablosu olsun. Bu tabloları;

» Kitaplar

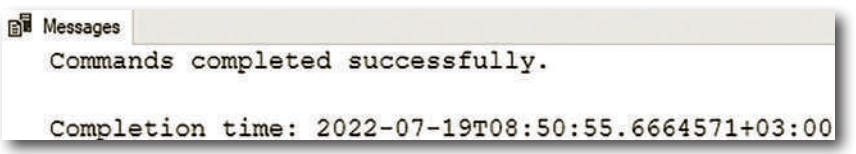
» Üyeler

» Emanet şekilde adım adım hazırlayalım.

Veritabanı ve tablolarımızı DDL komutları ile oluşturalım. Önce veritabanımızla başlayalım.

Sorgu 1: Kitaplık veritabanımızı DbLibrary ismiyle oluşturalım.

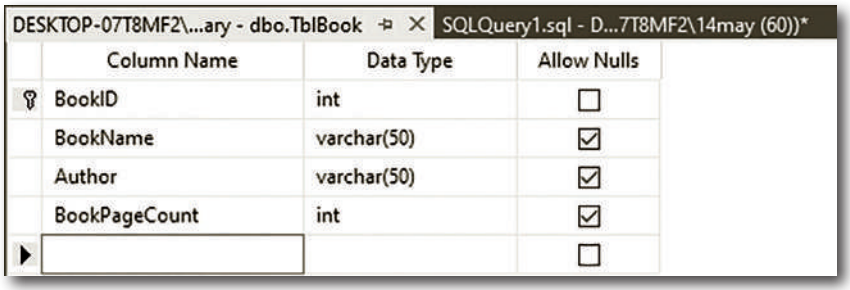
```
Create Database DbLibrary
```



DDL komutlarımızı kullanarak DbLibrary isminde yeni bir veritabanı oluşturarak sorgumuzu yazmaya başladık.

Sorgu 2: Kitaplar tablomuzu DDL üzerinden oluşturalım.

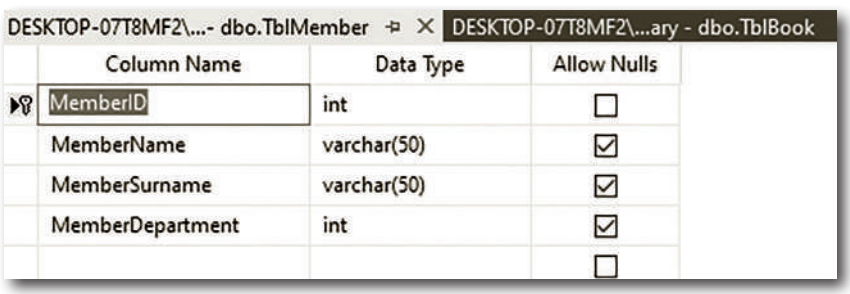
```
Use DbLibrary
Create Table TblBook
(
BookID int identity(1,1) Primary Key,
BookName Varchar(50),
Author Varchar(50),
BookPageCount int,
)
```



Use DbLibrary diyerek üzerinde çalıştığımız veritabanımızı seçerek başladık. Create Table komutu ile SQL'e yeni bir tablo oluşturacağımızı bildirdik. Tablomuz kitaplarımızı tutacağı için TblBook ismini verdik. Ardından sütunlarımızı oluşturmaya başladık. Kitap ID için BookID sütununu tanımladık. Kitap ID alanımız int türünde birincil anahtar ve otomatik artan bir yapıya sahiptir. Kitap adı için **varchar(50)** türünü tercih ettik. Yazar için de varchar(50) kullandık. Dilerseniz yazarları ayrı bir tabloda tutup ilişki içine de alabilirsiniz. Son olarak sorgulama kriterimizi biraz daha genişletebilmek adına kitabımızın sayfa sayısı için BookPageCount isminde bir sütun tanımlayıp bu sütunumuzun veri tipini int yaptık.

Sorgu 3: Üyeler tablomuzu DDL üzerinden oluşturalım.

```
Use DbLibrary
Create Table TblMember
(
MemberID int identity(1,1) Primary Key,
MemberName Varchar(50),
MemberSurname Varchar(50),
MemberDepartment int,
)
```



DATENAME MONTH

Sorgunun çalıştırdığı ana ait ayın hangi ay olduğu bilgisini verir.

Sorgu 20: Üzerinde bulunduğumuz ayın ismini Dataname ile bulalım.

```
Select Datename(Month,GetDate())
```

	(No column name)
1	July

`Datename` komutunu `Month` ile beraber kullandığımız zaman bize içerisinde bulunduğumuz ayın ismini İngilizce olarak verecektir.

DATENAME WEEKDAY

Sorgunun çalıştırdığı ana ait günün hangi gün olduğu bilgisini verir.

Sorgu 21: Üzerinde bulunduğumuz günün ismini Dataname ile bulalım.

```
Select Datename(WeekDay,GetDate()) as 'Gün'
```

	Gün
1	Tuesday

`Datename Weekday` komutunu kullanarak üzerinde çalıştığımız güne ait değeri sorgu sonucu olarak ve tabi yine İngilizce bir şekilde listelemiş olduk.

Sorgu 22: Parametre olarak girdiğimiz tarihin gün ve ay bilgisini getiren sorguyu yazalım.

```
Select Datename(WeekDay,'14.04.2008') as 'Gün',DateName(Month,'14.04.2008') as 'Ay'
```

Results Messages

```
Msg 241, Level 16, State 1, Line 1
Conversion failed when converting date and/or time from character string.

Completion time: 2022-07-19T10:57:33.8203660+03:00
```

Sorgumuzu çalıştırdığımız zaman karşımıza yukarıdaki gibi bir hata ekranı gelecektir. SQL tarih formatında bizden gün/ay/yıl değil ay/gün/yıl parametresi istiyor.

Sorgumuzu aşağıdaki gibi revize edip tekrar çalıştıralım.

```
Select Datename(WeekDay,'04.14.2008') as 'Gün',DateName(Month,'04.14.2008') as 'Ay'
```

	Gün	Ay
1	Monday	April

Bu kez sorgumuzu başarılı bir şekilde çalıştı. 14 Nisan 2008 tarihine ait gün ve ay ismi bilgisini getirmiş olduk.

DATEDİFF

2 tarih arasındaki farkı veren tarih fonksiyonudur. `Datediff` fonksiyonu 3 tane parametre alır. 1. parametre farkın hangi zaman aralığında (gün, ay, yıl vs.) olacağı, 2. parametre 1. tarih ve son parametre ise 2. tarih değerleri olacaktır.

Sorgu 23: DateDiff fonksiyonu ile 2 tarih arasındaki farkı hesaplayalım.

```
Select DateDiff(Day,'04.14.2022','04.20.2022') as 'Gün Farkı'
```

	Gün Farkı
1	6

`DateDiff` için 3 tane parametre alan bir sözdizimi mevcuttur. Burada 1. paramatre olarak 2 tarih arasındaki farkın hangi formatta olacağı bilgisini verdik. 2. parametre değeri olarak başlangıç tarihi 3. parametre ise bitiş tarihi oldu. Böylece bize 2 tarih arasındaki farkı getirmiş oldu.

Sorgu 24: 2 tarih arasındaki farkı bu kez ay olarak hesaplayalım.

```
Select DateDiff(Month,'05.01.2020','05.01.2022') as 'Ay Farkı'
```

	Ay Farkı
1	24

2 Tarih arasındaki fark için yine `Datediff` komutunu kullandık, interval yani değer aralığı olarak ise ay formatını seçtik böylece 2 tarih arasında kaç ay fark olduğunu hesaplamış olduk. Temel `Date` fonksiyonlarını kullandığımıza göre artık tablo bazında işlemlere başlayabiliriz. Bunun için ilk olarak ödünç tablomuz üzerinde çalışmaya başlayalım.

Sorgu 25: 1 Eylül 2022 tarihinde verilen kitapları listeleyen sorguyu yazalım.

```
Select * From TblBorrow where TakenDate='09.01.2022'
```

	BorrowID	MemberID	BookID	TakenDate	BroughtDate
1	1	1	4	2022-09-01 00:00:00	NULL
2	2	1	8	2022-09-01 00:00:00	NULL
3	3	2	5	2022-09-01 00:00:00	NULL

Sorgumuzu yazıp çalıştırdığımız zamana bize 1 Eylül tarihinde ödünç alınan kitaplar listelenecektir. Tarih parametresi girilirken yine ay/gün/yıl formatı esas alınmalıdır.

Sorgu 26: Eylül ayının 3'ünden itibaren ödünç alınan kitapları listeleyelim.

```
Select * From TblBorrow where Day(TakenDate)>=3
```

	BorrowID	MemberID	BookID	TakenDate	BroughtDate
1	4	3	9	2022-09-04 00:00:00	NULL

Sorgumuzda dikkat edilmesi gereken bir nokta var, burada `TakenDate` şartının önüne `Day` ifadesini ekledik. `Day` ifadesini eklediğimiz zaman sorgumuz gün bazında yapıldı yani Eylül ayının 3'ü ve sonrası olarak işlem gerçekleşti. Peki, şimdi tablomuza bir tane de Ekim ayı ödünç kitap verisi ekleyip bu sorguyu tekrar çalıştırmayı deneyelim.

Sorgu 27: Ödünç tablomuza Ekim ayında ödünç kitap alınmış formatta 1 tane örnek veri girişi yapalım.

```
insert into TblBorrow values(4,7,'10.22.2022',Null)
```

	BorrowID	MemberID	BookID	TakenDate	BroughtDate
1	1	1	4	2022-09-01 00:00:00	NULL
2	2	1	8	2022-09-01 00:00:00	NULL
3	3	2	5	2022-09-01 00:00:00	NULL
4	4	3	9	2022-09-04 00:00:00	NULL
5	5	4	7	2022-10-22 00:00:00	NULL

Eğer bu sorguda son parametreyi boş bırakırsanız SQL size hata mesajı döndürecektir. Biz şu anda ödünçlerin iade edilme tarihlerini girmiyoruz ancak bunu sorgu kısmından girerken Null olarak belirtmemiz önemli bir detaydır.

13

View

Bu Bölümde

Kitabımızın bu bölümünde ilişkili tabloların en önemli konularından biri olan View kullanımına değineceğiz. Normalde Join üzerinde elimizle uzun uzun yazdığımız sorguları View sayesinde nasıl pratik bir şekilde yapacağımızı göreceğiz. Ek olarak View yapısını hem sihirbaz üzerinde hem de sorgularla kullanıp DDL ile View oluşturacağız.

VIEW NEDIR?

İstenilen SQL tablolarını tıpkı gerçek bir tabloymuş gibi sanal bir tabloda toplayan, özellikle ilişki ve uzun birleştirmelerde karmaşık sorguları oldukça basit hale getiren yapılardır. Uzun bir SQL JOIN sorgusunu kod bloğumuz veya sorgu ekranımız içinde çağırmak yerine yalnızca **VIEW** ismini çağırarak aynı sorguyu çalıştırabiliriz. Yani tıpkı programlama dillerindeki metotlara benzerler. Ancak ilerleyen bölümlerde göreceğiz prosedür kavramını da andıran view'lerin bazı eksikleri bulunmaktadır. Ne artıları vardır diye soracak olursak;

» Uzun SQL sorgularını tek bir kelime altında çağırmamıza olanak sağlayarak sorgu karmaşasını önler.

» Tablo adları veya sütunların görünmesini istemediğimiz kısımda sadece VIEW ismini çağırarak ilgili tabloların güvenliği sağlanabilir.

» Sihirbazı sayesinde özellikle join gibi ilişkili tablo birleştirmeleri seçim ekranları ile tamamen fare kontrolüyle gerçekleştirilebilir.

» View'ler sayesinde sorgu süreleri kısaltılır.

» İçerisinde where ile şartlı sorgular getirilebilir.

» Group by komutu ile gruplandırma yapılabilir.

Bu avantajlarının yanında handikapları da bulunmaktadır. Bunları da sıralayacak olursak;

» View içerisine parametre gönderilemez.

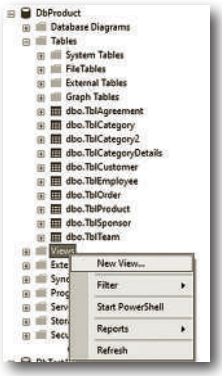
» Insert, update ve delete komutları kullanılabilir ancak çok fazla tavsiye edilmez, ana odak noktası select sorgularıdır.

En ciddi eksikleri bunlardır. Eksik noktalarına rağmen bize sağladığı artılar oldukça fazladır. Şimdi örnek bir view kullanımını önce sihirbaz üzerinde sonra da sorgu üzerinde görelim.

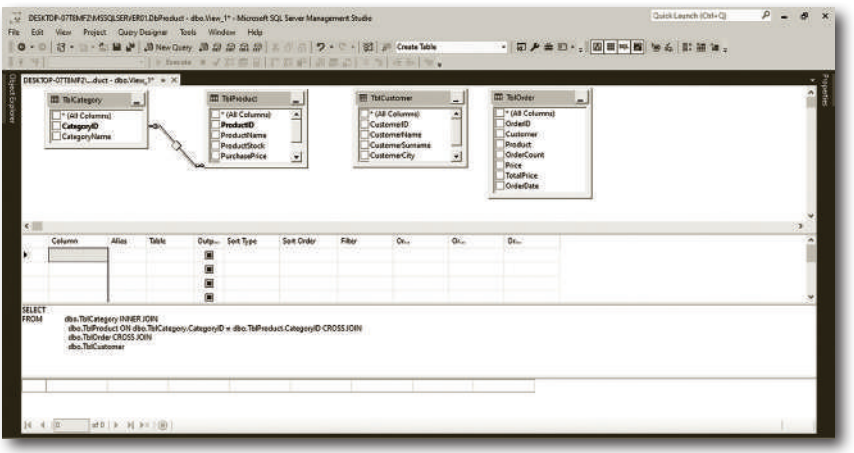
SIHIRBAZ ÜZERINDE VIEW KULLANIMI

Kitabımızın ilk bölümlerinde tablolar üzerinde sağ tuşla tıklayarak nasıl tablo oluşturabileceğimizi anlatmıştık. Şimdi benzer bir işlemi view için gerçekleştireceğiz. Sağ tuşla tıklayıp **New View...** seçeneğini seçelim. Yeniden ürün veritabanımıza geçmeyi unutmayalım zira işlemlerimizi ürün veritabanımız üzerinde gerçekleştireceğiz.

Bir sonraki görseli inceleyebilirsiniz.



Karşımıza aşağıdaki gibi bir pencere gelecektir. Bu pencereden Müşteri, Ürün, Kategori ve Sipariş tablolarını seçelim.



View bize diyagramda olduğu gibi ilişkili tabloları sorunsuz bir şekilde bağladı. Bizim daha önce oluşturduğumuz ancak ilişki içine almadığımız müşteri ve sipariş tablolarımız var. **Viewde** özellikle amacımız ilişkili verilerdeki değerleri sihirbaz üzerinden kolaylıkla **Select** sorgularına dönüştürmektir. O halde diyagramımızı açalım ve ilişkimizi düzenleyelim.

NOT	SQL Managment Studio 18.1 ile 18.4 sürümleri arasında Microsoft'un da kabul ettiği diyagram bazlı bir bug vardı. Bir veritabanında diyagram oluşturulduktan sonra SQL kapanıp açılırsa diyagram tekrar açıldığı zaman SQL kendini restartlıyor ve diyagramın açılmasına müsaade etmiyordu. Bunun önüne geçmek için diyagram silinip tekrar oluşturularak çözüm sağlanabilir veya SQL güncellemeleri yapılıp bir üst sürüme yükseltilebilir.

14

T-SQL

Bu Bölümde

T-SQL bölümü aslında kitabımızın 2. yarısı gibi düşünebilir. Hatta kitabımızın sonundaki kapsamalı **C# - SQL** örneğini de ele alırsak bu kitabı 3 ayrı modüle bölebiliriz.

T-SQL'den öncesi, T-SQL'den sonrası ve proje bölümü. Bu bölümde T-SQL'in ne olduğunu, değişken tanımlama işlemlerini, değişkenler üzerinden aritmetik ve alfabetik işlemlerin nasıl yapılacağını, değişken türlerinin tablo sütunlarına ait değerlerle nasıl kullanılacağını öğreneceğiz.

T-SQL NEDİR?

Temel veritabanı işlemleri bir süre sonra bizim için yeterli olmayacaktır. Örneğin: sorgu bloğumuzda değişken, döngü, karar yapısı gibi komut bloklarına ihtiyaç duyacağımız durumlar olabilir. İşte böyle durumlarda devreye T-SQL komutları girecektir. T-SQL sayesinde SQL üzerinde;

» Değişken
» Karar yapısı
» Döngü
» Geliştirici Tanımlı fonksiyon
» Hata kontrolleri
» Mantıksal karşılaştırmalar ve daha pek çok başlığı kullanabileceğiz.

DEĞİŞKENLER

C#, Java, Python, C gibi programlama dillerinde tanımlayıp kullandığımız değişken yapısını SQL üzerinde de T-SQL ifadeleri başlığı altında kullanabilmekteyiz. Değişkenler bellekte geçici olarak tutulup program veya uygulama çalıştığı sürece çağrılıp üzerinde işlem yapılmasını sağlayan yapılardır. Genel olarak programlama dillerinde geçerli olan değişken tanımlama kuralları burada da geçerlidir. Bu kuralları sıralayacak olursak;

» SQL'de değişkenler @ sembolü ile başlamak zorundadır.
» Değişkenler sayı veya semboller ile başlamaz.
» Değişkenlerin isimlerinde araya boşluk konulmaz.
» Değişken tanımlamalarında Türkçe karakterlerde hata vermez, ancak olabildiğince Türkçe karakter kullanımından kaçınılmalıdır.
» Olabildiğince net isimler seçilmelidir.
» Eğer değişken adı 2 veya daha fazla karakterden oluşuyorsa isimler ya birleşik yazılmalı ya da araya _ sembolü konulmalıdır.
» Değişken tanımlamaları `Declare` komutu ile yapılır.
» Değişkenlere değer atama işlemleri set komutu ile yapılır.

Kullanım şekli de aşağıdaki gibidir;

```
Declare @değişken_adi veri_türü
```

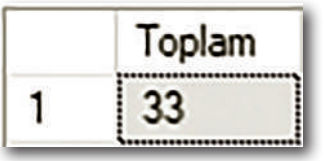
Sorgu 1: sehir isminde bir değişken oluşturup bu değişkenimize değer ataması yapalım.

```
Declare @sehir varchar(20)
Set @sehir='Adana'
Select @sehir
```

`Declare` komutuyla `sehir` isminde bir değişken tanımladık. Değişken tanımlarken **Türkçe** karakter kullanmaktan kaçındığımız için şehir yerine `sehir` ismini tercih ettik. Değişkenimizin türünü `varchar(20)` olarak belirledik. `Set` komutu SQL'de bir değişkene değer ataması yapmak için kullanılır. Bizde burada sehir değişkenimize değer ataması yapmak için set komutundan faydalandık. Değer atamasını yaptıktan sonra değişkenimiz ekranda gösterebilmek için `Select` komutundan faydalandık.

Sorgu 2: Değişken olarak tanımlanan 2 sayıyı toplayan sorguyu yazalım.

```
Declare @sayi1 int, @sayi2 int, @sonuc int
Set @sayi1=24
Set @sayi2=9
Set @sonuc=@sayi1+@sayi2
Select @sonuc as 'Toplam'
```



`Declare` komutunu kullanarak `int` türünde 3 tane değişken tanımladık. Normalde C# veya diğer dillerin çoğunda tek satırda bütün değişken isimleri yan yana yazılıp değişken türünün 1 kez çağrılması yeterlidir. Ama SQL'de her bir değişken için ayrı ayrı olarak türünün belirtilmesi gerekiyor. `Sayi1` için 24, `Sayi2` için 9 atamalarını yapıp sonuç değişkenimiz aracılığıyla sayılarımızı topladık ve sonucu ekranda gösterdik.

NOT

Programlama dillerinde birden fazla değişkeni tek satırda tanımlayabiliyorduk. SQL'de de aynı durumu kullanabiliriz. Yalnızca bir fark var o da programlama dillerinde tek satırda birden fazla değişken tanımlaması yaparken ilgili değişkenin türünün her bir değişken için tek tek yazılması zorunluluğunun olmamasıdır. Yani C dilinde int sayi1, sayi2, sayi3, … şeklinde aynı türdeki değişkenleri yan yana tanımlayabiliriz. Ancak SQL'de bu değişkenlerin her birinin isminden sonra mutlaka türlerinin de yazılması gerekmektedir. Bu durumları göz önünde bulundurarak değişkenlerimizi tanımlayıp, SET komutu ile atamaları yapıp SELECT ile ekrana yazdırma işlemini gerçekleştirdik. Değişkenler sadece dışarıdan bizim verdiğimiz parametreleri almaz. Tablo içinden gelebilecek herhangi bir değeri de değişkenlere atayıp bunun üzerinden işlem yaptırabiliriz.

Sorgu 3: Ürünler tablomuzdaki toplam ürün sayısın bir değişkene atayıp değişken üzerinden listeleyen sorguyu yazalım.

```
Declare @ToplamUrun int
Set @ToplamUrun=(Select Count(*) From TblProduct)
Select @ToplamUrun as 'Toplam Ürün Sayısı'
```

	Toplam Ürün Sayısı
1	14

`ToplamUrun` isminde `int` türünde bir değişken oluşturduk. Bu değişkenimize değer ataması olarak ürün tablomuzdaki verilerin `Count` ile sayısını hesaplatıp atadık. Ardından `ToplamUrun` değişkenimizi `Select` ile gösterdik.

PRINT KOMUTU

Hazırlamış olduğumuz sorgu sonuçlarını bazen tablo türünde değil de mesaj çıktısı olarak metinsel bir şekilde görmek isteyebiliriz. `Print` ifadesinin kullanımını örnek sorgu üzerinde görelim

Sorgu 4: Sehir isminde bir değişken oluşturup şehrimizin atadığımız değerini print komutu ile yazdıralım.

```
Declare @sehir varchar(20)
Set @sehir='Bursa'
Print 'Girilen Şehir:' + @sehir
```

```
Messages
Girilen Şehir:Bursa

Completion time: 2022-07-20T15:14:32.1336979+03:00
```

`Declare` komutunu kullanarak sehir isminde varchar türünde bir değişken tanımladık. Tanımladığımız bu değişkene `Bursa` değerini atadık. Yazdırma işlemi için ise bu kez `Select` değil `Print` ifadesini kullandık.

16

Prosedürler ve Fonksiyonlar

Bu Bölümde

Kitabımızın bu bölümünde prosedür kavramını, prosedürünü ne olduğunu ve nerelerde kullanıldığını, prosedürlerin nasıl oluşturulacağını, mevcut prosedürler düzenleme yapmayı, DDL komutları ile prosedürlerin ilişkisini, prosedürlerin nasıl çalıştırılacağını, prosedürlerde parametre kullanımını ve prosedürlerde output kullanımını göreceğiz.

Ek olarak fonksiyon kavramını, fonksiyonların nerede ve nasıl kullanılacağını, DDL komutlarını kullanarak nasıl fonksiyon oluşturacağımızı, tablo türü fonksiyonların neler olduğunu ve nasıl oluşturulacağını öğrendik.

PROSEDÜR NEDİR?

Programlama dillerindeki metot yapısına ciddi anlamda benzeyen ve viewlere göre daha avantajlı olan SQL komutlarıdır. Prosedürler uzun SQL ifadelerini tek kelimelik yapılara sığdırmakta, bu sayede hem sorgu yükünü azaltmakta hem de sorgu güvenliğini sağlamaktadır. Prosedür oluştururken genellikle DDL komutları kullanılır. Oluşturma işlemi için **Create**, silme işlemi için **Drop** ve güncelleme için ise **Alter** ifadesi kullanılmaktadır.

Viewlerde herhangi bir view çalıştırılmak istendiğinde "`Select * From View_Ad1`" şeklinde bir kullanım gerçekleştiriyorduk. Prosedürlerde ise ilgili prosedürler execute ifadesi ile çalışır. Viewlere göre en önemli avantajlarından bir tanesi dışarıdan parametre alabilmesidir. Prosedürler bazı noktalarda **"stored procedure"** yani depolanmış yordamlar olarak da karşımıza çıkabilir. Prosedürlerin avantajlarını sıralayacak olursak;

» Uzun SQL sorgularını tek kelimelik komutlar haline getirip sorgu tekrarını önlerler.

» Dışarıdan parametre alabilirler.

» C# veya benzeri dillerde SQL'de hazırlanmış olan uzun sorgular yerine sadece prosedürün adı çağrılarak işlem yapılabilir.

» Performansı artırır.

» Sorgu güvenliğini artırır.

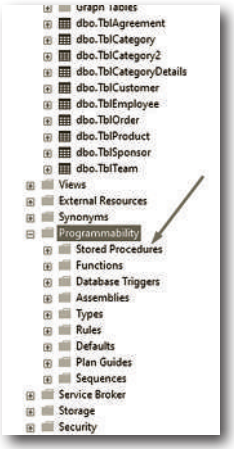
» Ağ trafiğini yormayacaktır.

» İlk derleme anından sonra tekrar derlenmeye ihtiyaç duymaz.

PROSEDÜR OLUŞTURMA

Prosedür oluşturmak için 2 temel seçeneğimiz olacak. Bunlardan ilki **Prosedür** klasörümüz üzerinde sağ tuşla tıklayıp **New** seçeneği ile sorgu ekranı üzerinden oluşturmaktır.

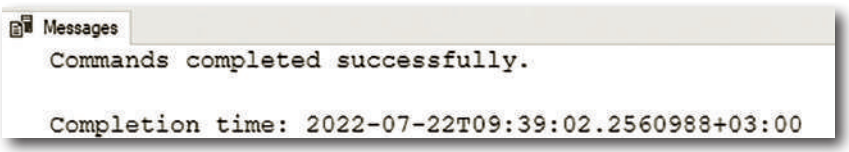
2. seçenek ise **New** seçeneğine girmeden tamamen sorgu üzerinden prosedür oluşturmaktır. Biz kitabımızda 2. seçeneği kullanacağız. Kendi oluşturduğumuz prosedürler dışında SQL'in bize hazır sunduğu prosedürler de bulunmaktadır. Bunları ilerleyen sayfalarda göreceğiz. İlk olarak prosedürlerimizin yer alacağı klasörümüze bakalım.



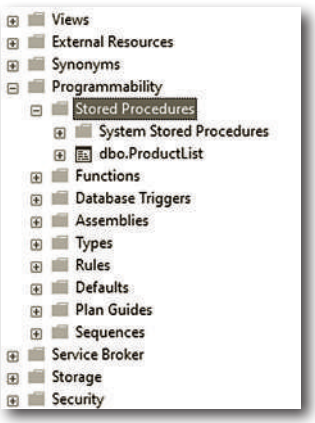
Oluşturduğumuz prosedürlere **Programmability** klasörünün altından ulaşacağız. Şimdi DDL komutları ile prosedürlerimizi oluşturmaya başlayalım.

Sorgu 1: Kategori tablomuzda bulunan değerleri listeleyen prosedürü hazırlayalım.

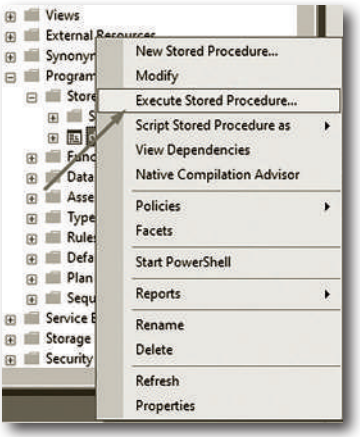
```
Create Procedure ProductList
As
Select * From TblProduct
```



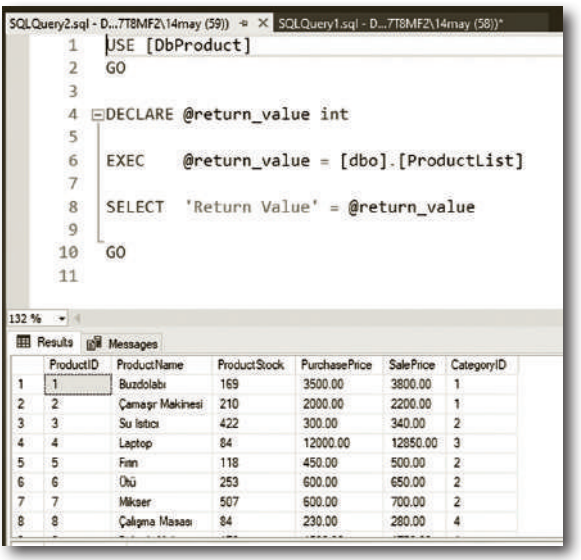
Sorgumuzu yazıp çalıştırdığımız zaman karşımıza üstte bulunan görseldeki gibi komutun başarılı bir şekilde çalıştığı mesajı gelecektir. **Programmability** klasörümüz üzerinde sağ tuşla tıklayıp **Refresh** seçeneğine tıkladığımız zaman ilgili prosedürümüzün buraya geldiğini göreceğiz.



Görüldüğü gibi prosedürümüz artık hazır. Prosedürümüzü çalıştırmak için prosedürümüzün üzerinde sağ tuşla tıklayıp **Execute Stored Procedure** seçeneğine tıklayabiliriz.



Execute seçeneğine tıkladığımız zaman karşımıza aşağıdaki gibi bir pencere gelecektir.

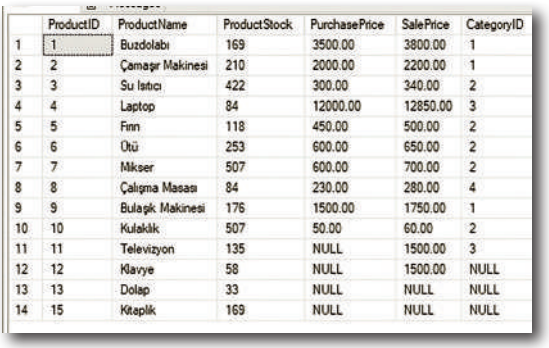


	ProductID	ProductName	ProductStock	PurchasePrice	SalePrice	CategoryID
1	1	Buzdolabı	169	3500.00	3800.00	1
2	2	Çamaşır Makinesi	210	2000.00	2200.00	1
3	3	Su Isıtıcı	422	300.00	340.00	2
4	4	Laptop	84	12000.00	12850.00	3
5	5	Fırın	118	450.00	500.00	2
6	6	Ütü	253	600.00	650.00	2
7	7	Mikser	507	600.00	700.00	2
8	8	Çalışma Masası	84	230.00	280.00	4

Eğer bu işlemin zahmetli olduğunu düşünüyorsanız o zaman sorgu yazarak prosedürümüzü çalıştırabiliriz.

Sorgu 2: Execute sorgumuz ile prosedürümü çalıştıralım.

```
Execute ProductList
```



	ProductID	ProductName	ProductStock	PurchasePrice	SalePrice	CategoryID
1	1	Buzdolabı	169	3500.00	3800.00	1
2	2	Çamaşır Makinesi	210	2000.00	2200.00	1
3	3	Su Isıtıcı	422	300.00	340.00	2
4	4	Laptop	84	12000.00	12850.00	3
5	5	Fırın	118	450.00	500.00	2
6	6	Ütü	253	600.00	650.00	2
7	7	Mikser	507	600.00	700.00	2
8	8	Çalışma Masası	84	230.00	280.00	4
9	9	Bulaşık Makinesi	176	1500.00	1750.00	1
10	10	Kulaklık	507	50.00	60.00	2
11	11	Televizyon	135	NULL	1500.00	3
12	12	Klavye	58	NULL	1500.00	NULL
13	13	Dolap	33	NULL	NULL	NULL
14	15	Kitaplık	169	NULL	NULL	NULL

Görüldüğü gibi Execute ifadesiyle ilgili prosedürümüzü çalıştırabiliyoruz.

Sorgu 3: Ürün tablomuzda bulunan ürün adı, stok sayısı ve kategori adını Stored Procedure ile oluşturan sorguyu yazalım.

```
Create Procedure ProductListWithCategory
As
Select ProductName,ProductStock,CategoryName From TblProduct
inner join TblCategory
On TblProduct.CategoryID=TblCategory.CategoryID
```

Sorgumuzu oluşturduktan sonra Execute komutu ile prosedürümüzü çalıştıralım.

```
Execute ProductListWithCategory
```

	ProductName	ProductStock	CategoryName
1	Buzdolabı	169	Beyaz Eşya
2	Çamaşır Makinesi	210	Beyaz Eşya
3	Su Isıtıcı	422	Küçük Ev Aletleri
4	Laptop	84	Bilgisayar
5	Fırın	118	Küçük Ev Aletleri
6	Ütü	253	Küçük Ev Aletleri
7	Mikser	507	Küçük Ev Aletleri
8	Çalışma Masası	84	Mobilya
9	Bulaşık Makinesi	176	Beyaz Eşya
10	Kulaklık	507	Küçük Ev Aletleri
11	Televizyon	135	Bilgisayar

Sorgumuzu yazıp çalıştırdığımız zaman görseldeki gibi 10 tane ürün listelenecektir. Create ifadesinden sonra Procedure yerine Proc 'da yazabiliriz. Aynı şekilde sorgumuzu çalıştırmak için Execute yerine Exec yazmamız yeterli olacaktır. Her iki komut da aynı işlemi yapmaktadır.

17

Tetikleyiciler

Bu Bölümde

Bu bölümde kitabımızın en önemli konularından biri olan tetikleyicileri öğreneceğiz. Tetikleyicilerin neler olduğunu, neden gerekli olduklarını, tetikleyicilerin nasıl oluşturulacağını, mevcut bir tetikleyicinin nasıl güncelleneceği ve silineceğini, parametreli tetikleyicilerin nasıl kullanılacağını öğreneceğiz.

TETİKLEYİCİ NEDİR?

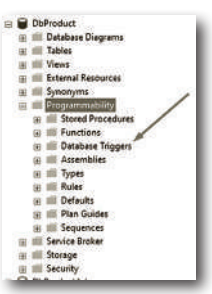
SQL'de en keyif alacağınız ve işinize inanılmaz derecede yarayacak olan başlıklardan bir tanesi tetikleyiciler ya da İngilizce karşılığı ile **Trigger** yapısıdır.

Tetikleyiciler; herhangi bir tabloda ekleme, silme, güncelleme gibi bir işlem yapıldıktan sonra bu tabloda yapılan işleme göre aynı tabloda veya bir başka tabloda farklı bir işlemin gerçekleştirilmesidir. Örneğin; satış tablonuzda herhangi bir ürünün satışını yaptıktan sonra ürün tablonuzda bu ürünün stok sayısı azalmalıdır. İşte bu noktada devreye tetikleyiciler girecektir.

SELECT işleminde ilgili tabloda herhangi bir değişiklik oluşmadığı için tetikleyici tanımlaması yapılmaz. Tetikleyici oluşturabilmek için ilgili tabloda mutlaka bir değişiklik olması gerekmektedir. Örneğin; INSERT işleminde tabloya yeni bir kayıt eklenerek tabloda bir değişiklik yapılmış olunur. Aynı durum silme ve güncelleme işlemleri için de geçerlidir. Tetikleyicilerin oluşturulması yapı olarak prosedürlere benzer. Oluşturma işlemleri için DDL komutlarından Create kullanılır. Aynı şekilde silme için drop ve güncelleme için alter komutları kullanılmaktadır. Tetikleyiciler aşağıdaki gibi oluşturulurlar;

```
Create Trigger Tetikleyici_Adı
On Tablo_Adı
After Insert & Delete & Update
As
İşlemler
```

`CREATE TRIGGER` ifadesiyle tetikleyicimizi oluşturmaya başlıyoruz. `ON` ifadesinin bulunduğu 2. satırda tetikleyicimizin hangi tabloda işlem yapıldıktan sonra çalıştırılacağı bilgisini giriyoruz. `AFTER` ifadesi ile tetikleyicimizin hangi işlemden sonra çalışacağı bilgisini yazıyoruz. Tetikleyicilerimizi ekleme, silme veya güncelleme işlemlerinin birinden sonra çalışırlar. `AS` komutumuzu ekleyip hemen altında işlemlerimizi yazmaya başlıyoruz.



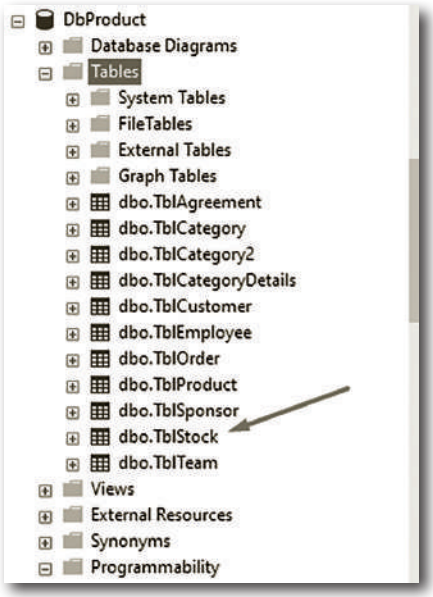
Tablo bazında olmayan veritabanı bazlı tetikleyicilerimize prosedürlerde ve fonksiyonlarımızda olduğu gibi yine **Programmability** klasörünün altından ulaşıyoruz. Tablolarda ise farklı bir yerde olduğunu ilerleyen sayfalarda göreceğiz. Şimdi tetikleyici kullanımlarına tek tek bakalım.

INSERT TRİGGER

DML komutları ile beraber kullanabileceğimiz 3 tane tetikleyici türü bulunmaktadır. Bunlardan ilki INSERT tetikleyicileridir. INSERT tetikleyicilerde işlemler ekleme işlemi yapıldıktan sonra gerçekleşir. Şimdi tetikleyicileri daha somut bir şekilde anlayabilmek için DDL komutları ile yeni bir tablo oluşturalım. Bu tablomuz ürünler tablomuzda bulunan ürünlerin sayısını tutsun. Biz ürün ekledikçe bu tablonun içindeki adet sütunu otomatik olarak artacak ve sildikçe azalacaktır. Önce DDL komutları ile ürün tablomuzdaki ürün sayısını tutacak olan sorgumuzu yazalım.

Sorgu 1: DDL komutları ile ismi stok olan ve içinde INT türünde adet isminde bir sütun barındıran tablomuzu oluşturalım.

```
Create Table TblStock
(
StockCount Int
)
```



Stok tablomuzu sorunsuz bir şekilde oluşturduk. Örneğimize devam edelim.

Sorgu 2: Stok tablomuza başlangıçta 0 değerini INSERT sorgusu ile girelim.

```
Insert Into TblStock VALUES (0)
```

Insert işlemi yaparken ilk defa VALUES'den önce tablo sütun adlarını girmedik. Eğer ki özellikle otomatik artan bir alan yok ise; sütun adları yazılmadan VALUES ifadesinin sağ tarafına sırasıyla ilgili sütunlara göre veriler girilerek INSERT işlemi sorunsuz şekilde çalıştırılabilir. Şimdi bu tablomuzun adet sütununa ürün tablomuzdaki kayıt sayısını sorgu ile gönderelim.

Sorgu 3: Stok tablomuzun adet sütununun değerini ürün tablomuzdaki kayıt sayısı olarak güncelleyen sorguyu yazalım.

```
Update TblStock Set StockCount=(Select Count(*) From TblProduct)
Select * From TblStock
```

	StockCount
1	14

Eğer ürün sayısını elle girmemiz istenseydi direkt olarak adet sütununa 14 yazıp işlemi tamamlayabilirdik. Ancak bizden soruda istenen şey ürün tablosundaki kayıtların da sorgu ile bulunmasıdır. Stok tablomuzdaki adet sütununu ürün tablomuzdaki ürün sayısı ile sorgumuzdaki gibi bir alt sorgu yardımı ile yaptık. İç sorgudan gelen değer ürün tablosundaki ürünlerin toplam kayıt sayısını tutacaktır. Yani sorgumuz Update TblStock StockCount =14 olmuş oldu. Stok tablomuzda tek sütun ve tek satırlık kayıt olduğu için Where şartı yazmamız gerekmedi.

Sorgu 4: Ürün tablomuza yeni bir ürün kaydı yapılınca stok tablomuzda bulunan adet sütununun değerini 1 artıran tetikleyici sorgusunu yazalım.

```
Create Trigger IncreaseStock
On TblProduct
After Insert
As
Update TblStock Set StockCount=StockCount+1
```

IncreaseStock isminde bir tetikleyici oluşturarak sorgumuza başladık. Tetikleyicimize verdiğimiz isim 2 kelimeden oluştuğu için ve SQL'de tablo ismi, veritabanı ismi, değişken ismi gibi adlandırmalarda boşluk kullanılamadığından

19

Index

Bu Bölümde

Kitabımızın bu bölümünde Index kavramının ne olduğunu, büyük ölçekli tablolarda arama işlemini nasıl yapacağımızı, Index kavramına neden ihtiyacımız olduğunu ve nasıl kullanılacağını öğreneceğiz.

INDEX NEDİR?

Şu ana kadar çalıştığımız veritabanı ve tablolar hep küçük ölçekli verilerden oluşuyordu. Bundan dolayı tablolarımız üzerinde **Select** ile listeleme yaparken istediğimiz veriye ait sonuçlar oldukça hızlı bir şekilde 1 saniyenin altındaki zaman aralığında karşımıza geliyordu.

Peki, tablolarımızın içeriği büyürse yani 10 satırlı değil de 1 milyon satırlı bir veritabanında örneğin bir bankadaki müşterilere ait bilgilerin tutulduğu veya bir e-ticaret platformunda yüzbinlerce ürünü tutan bir veritabanında da yazacağımız **Select** sorgusu aynı hızda çalışır mı? Cevabımız **Hayır** olacaktır. İşte bu noktada ilgili sorgumuzun daha hızlı çalışabilmesi için Index denilen yapıya ihtiyaç vardır. Index'ler tablolarda belirlenen sütunlardaki aramaların daha hızlı bir şekilde yapılmasını sağlar. Burada aslında bir veri yapısı vardır denilebilir.

Normalde tablomuzda Search yani arama işlemi yaparken bu arama işlemi **scan** denilen tarama yöntemiyle baştan sona yapılır. Küçük ölçekli tablolarda herhangi bir performans zafiyeti olmadan gayet verimli bir şekilde scan yapılabilir. Ancak kayıt sayısı arttıkça scan kullanmak verim olarak istenen sonuçları getirmeyecektir. Aslında Index'leme işlemini kitaplardaki içindekiler bölümüne benzetebiliriz. Diyelim ki biz kitabımızda alt sorguları arıyoruz. Her sayfaya tek tek bakmak yerine sadece içindekiler bölümüne bakılıp ilgili bölüm bulunup direkt o bölümün sayfası açılabilir. Şimdi 2 temel Index türünü inceleyelim.

CLUSTERED INDEX

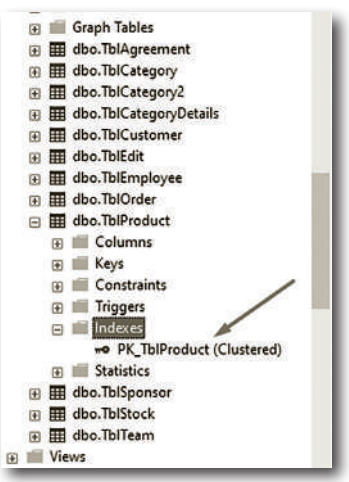
Clustered kelimesini Türkçeye **kümelenmiş** olarak çevirebiliriz. Kümelenmiş Index'lere günlük hayattan bir örnek verelim. Üniversitenizde kitap ihtiyacınız oldu ve kütüphaneye gittiniz. Bu kütüphanede bulunan kitaplar herhangi bir koşula göre sıralanmamışlar. Gelişigüzel bir şekilde raflara dizilmiş. Diyelim kütüphanede 10.000 tane kitap var. Siz SQL ile ilgili şu an elinizde tuttuğunuz **SQL Eğitim Kitabı** adlı eserimizi arıyorsunuz. Her sırada bir tane dolap her dolapta da 100 tane kitap olduğunu varsayalım. Yani bu kütüphanede toplamda 100 tane dolap var. Sizin bir dolapta bulunan kitaplara bakmanız ortalama 5 dakika sürsün. Ve yine diyelim ki aradığınız SQL kitabı 26. dolapta. Siz 26. dolaba gelene kadar 25 * 5'den tam 125 dakika yani 2 saatten fazla bir süreyi kitap aramakla geçireceksiniz.

Düşünsenize ayda 4 tane kitap almak istediğinizi. Bu kütüphaneye gelip saatlerce arama yapmanız gerekecek. Eğer kütüphanedeki kitaplar en başından itibaren alfabetik olarak dizilseydi bu sorunlar ortaya çıkamayacaktı. Mesela A

harfindeki kitaplar 1. dolaba, B harfindeki kitaplar 2. dolaba şeklinde bir dizme işlemi yapılsaydı siz direkt S harfinin bulunduğu dolaba gidip yalnızca 1-2 dakika içinde aradığınız kitabı bulabilecektiniz. İşte **Clustered Index** kavramı tam da bu oluyor. İlk başta yaptığımız tek tek tüm dolaplara ve kitaplara bakma işlemi ise scan oluyor. Normalde kütüphanede 1 tane dolap olsa ve bu dolapta 30 tane kitap olduğunu varsayarsak o zaman Index kullanmaya ihtiyacınız olmayacak. Zira 30 tane kitap içinde istediğimiz kitabı bulmak oldukça kolay olacaktır. Ancak 10.000 tane kitap için aynı durum söz konusu değildir.

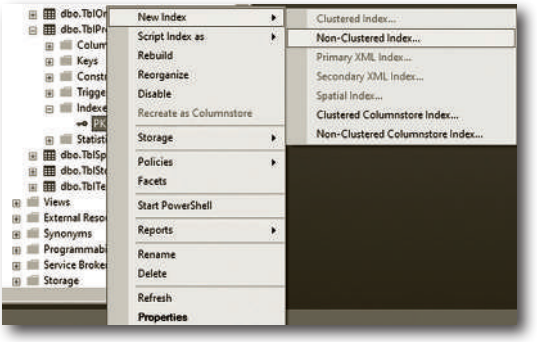
NON CLUSTERED INDEX

Bir tablo için sadece bir tane Clustered Index tanımlanabilir. Non Clustered ise bir tabloda birden fazla Index'in tanımlandığı yapılardır. Aynı kütüphanede siz sadece kitap adına göre değil aynı zamanda yazara göre de arama yapmak istiyorsunuz. Örneğin Stefan Zweig'ın kitaplarını görmek istiyorsunuz. Burada hem yazar hem de kitap adına göre yönlendirme yapılacak. Satranç kitabı S harfinde iken Korku kitap K harfinde, Olağan Üstü Bir Gece kitabı ise O harfinde olacaktır. Yani Non Clustered ile siz direkt ilgili kitaba değil kitabın konumuna erişim sağlayıp sonra o konumda ilgili kitaba ulaşacaksınız. Tıpkı bir kitabın en sonunda bulunan kelime Index'inde olduğu gibi. Kelime Index'i sayesinde herhangi bir kelimenin hangi sayfada geçtiğini görebilirsiniz. Aslında tablomuzda kullandığımız otomatik artan ve ID değeri bir Index'leme işlemidir. Bunun için tablomuzun detaylarına bakabiliriz.

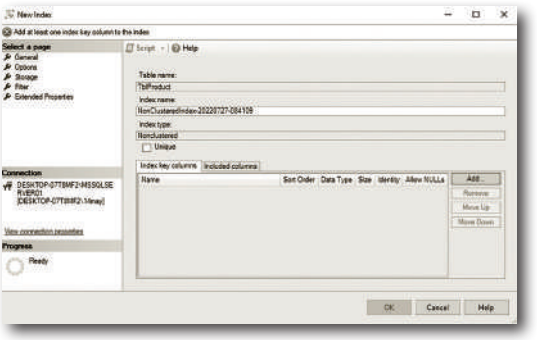


Non Clustered Index oluşturmak için tablomuzun **Indexes** alanı üzerinde sağ tuşla tıklayalım.

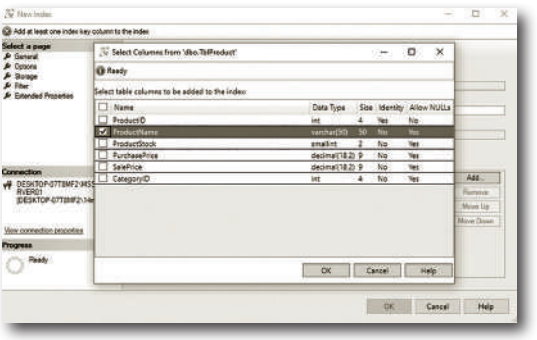
New Index seçeneğinden aşağıda görülen resimdeki alana gelelim.



Karşımıza çıkan pencereden **Add** seçeneğine tıklayalım.



Burada ürün adını seçelim.



Görüldüğü gibi artık birden fazla alana göre Index'leme işlem yapabileceğiz. Index işlemini sorgularla da oluşturabiliriz.

21

Yedek Alma ve Attach

Bu Bölümde

Bu bölümde yedek alma işleminin farklı yöntemlerle nasıl yapılacağını ve yedeği alınan bir veritabanının tekrar nasıl veritabanına dahil edileceğini öğreneceğiz. Backup, Script ve Log dosyaları ile ayrı ayrı yedek alma ve alınan yedeklerin tekrar SQL'e nasıl dahil edileceğini göreceğiz.

NASIL YEDEK ALINIR?

SQL'de yedek alma işlemi için birden fazla yöntem bulunmaktadır. Bunlar içinden 3 tanesini kullanacağız. Bunlar;

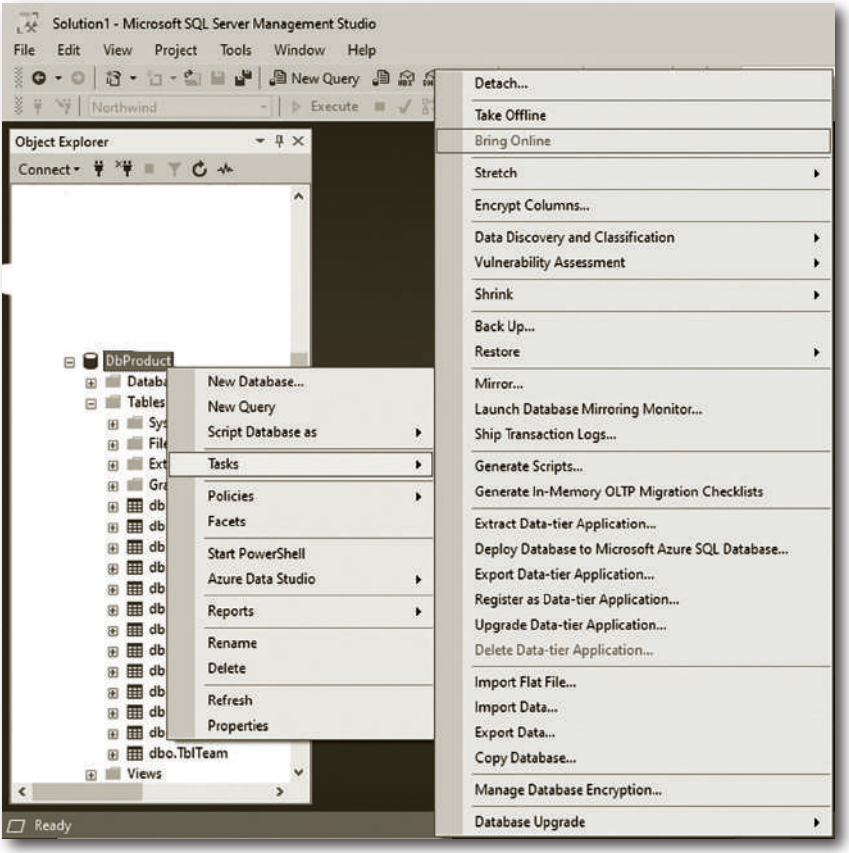
» Backup Dosyası ile Yedek Alma

» Script ile Yedek Alma

» Mdf Ldf Log Dosyaları ile Yedek Alma

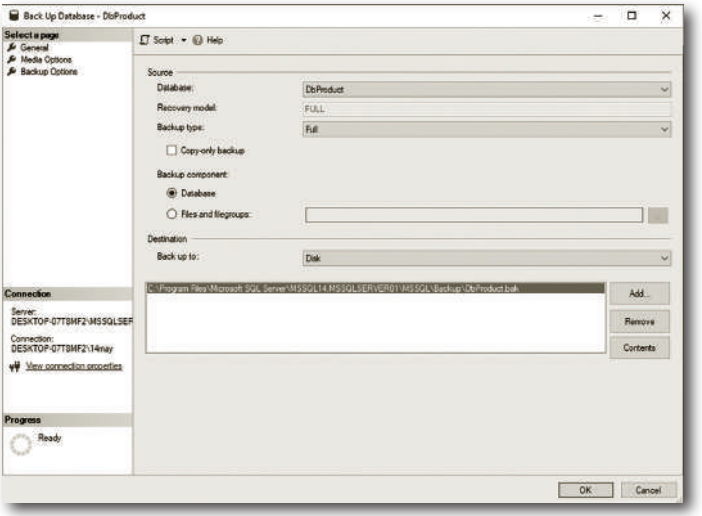
BACKUP İLE YEDEK ALMA

İlk olarak Backup ile başlayalım. Backup yedeğini alıp bu yedeği farklı bir bilgisayarda açabiliriz. Kitabımız boyunca kullandığımız ürün yönetim veritabanımız üzerinde sağ tuşla tıklayalım. **Tasks** kısmından **Back Up...** seçeneğine gelelim.

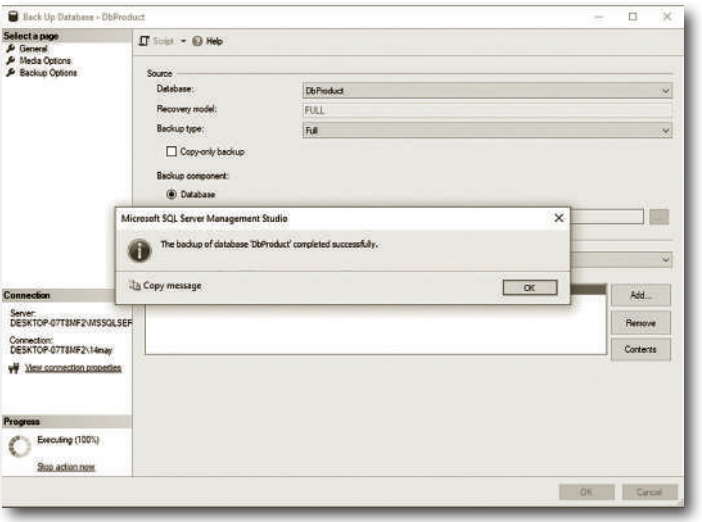


Back Up... seçeneğine tıkladığımız zaman karşımıza aşağıdaki gibi bir pencere

gelecektir. Burada Backup yani yedeğin alınacağı konum seçilebilir. Eğer konum kısmında herhangi bir değişiklik yapmazsak **default** yani varsayılan değer olarak **C** sürücümüzde SQL'in kurulu olduğu dizindeki Backup klasörü içine kaydolacaktır.



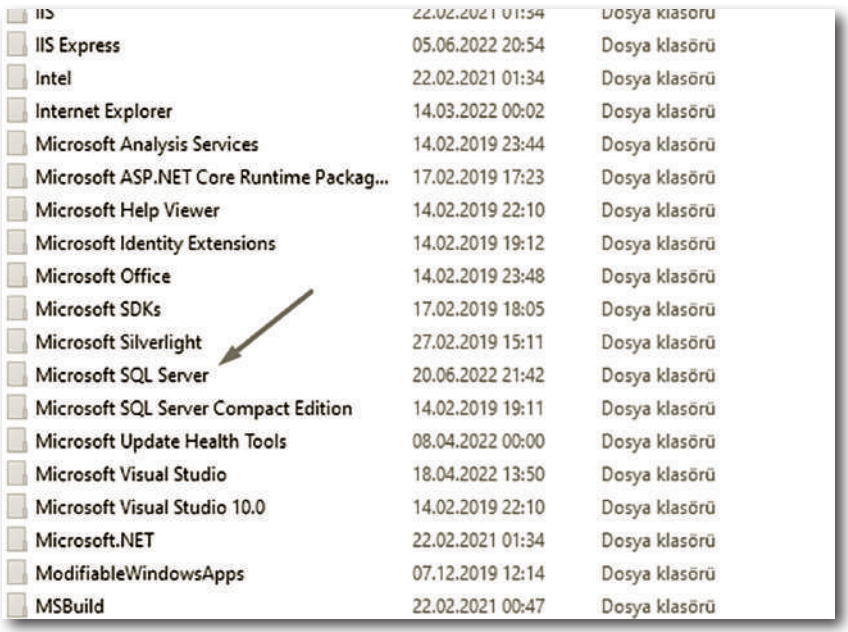
Ok butonuna tıklayalım. **OK** seçeneğine tıkladığımız zaman **Backup** dosyasının başarılı bir şekilde alındığı mesajı karşımıza gelecektir.



Şimdi Backup dosyamızın bulunduğu dizini açalım.

Bilgisayarımızda C sürücümüzü açalım. Burada **Program Files** kısmına çift tıklayalım.

Burada **Microsoft SQL Server'ı** seçelim.



Benim bilgisayarımda birden fazla SQL Server sürümü olduğu için sizin ekranlarınıza göre biraz daha fazla klasör bulunacaktır. Burada kullandığımız sürüme ait log dosyaları en alttaki klasördedir.

23

Örnek Proje: Kitaplık Veritabanı

Bu Bölümde

Bu bölümde kitabımızda öğrendiğimiz kavramların neredeyse tamamını tek bir veritabanında toplayıp yeni bir veritabanı ve tablolar oluşturacağız. Bu tabloları gelecek bölümde C# dili ile somut bir projede kullanacağız. Projemiz bir kitaplık yönetim modülü olacak. Projemizde üyeler, kitaplar, yazarlar, yayınevleri, emanetler, gecikmeler gibi tablolar kullanacağız.

Giriş

Geldik kitabımızın son bölümüne. Yaklaşık 6 ay süren çalışmamızın son kısmını kitabımızın tamamını kapsayacak bir örnek olmasını istedik. Böylece kitabımızı kapatırken neredeyse bütün bölümlerin tekrarı tek bir veritabanı üzerinden yapılabilir. Bu veritabanımız bir kütüphane veritabanı olsun. Kütüphane veritabanımız içinde kullanacağımız başlıklar şunlar olacaktır;

» DDL Komutları
» DML komutları
» Select Sorguları
» Operatörler
» View
» Prosedür
» Tetikleyici
» Alt Sorgular
» Gruplandırmalar
» Birleştirmeler
» İlişkiler ve daha pek çok başlığı bu örneğimizde kullanacağız.

Kütüphane veritabanımızda kullanacağımız tablolar şöyle olacaktır;

» Kitaplar
» Yazarlar
» Üyeler
» Emanetler
» Üye İşlemleri
» Yayınevi

Toplamda ilk etapta 6 farklı tablomuz olacak. Her sorgumuzu bir uygulama olarak ele alacağız. Örnekleri yapmayı ihmal etmeyin. Öyleyse başlayalım.

Sorgu 1: DDL üzerinden veritabanımızı oluşturalım.

```
Create Database DbBookStore
```

Veritabanımıza BookStore yani kitapçı ismini verdik. Dilerseniz Library ismini de kullanabilirsiniz. Şimdi tablolarımızı oluşturalım. İlk olarak yazarlar tablomuzla başlayalım.

Sorgu 2: Yazar tablomuzu oluşturalım.

```
Use DbBookStore
Create Table TblAuthor
(
AuthorID int primary key identity(1,1),
AuthorName varchar(50),
AuthorSurname varchar(50)
)
```

Yazar tablomuzu yazar ID, yazar adı ve yazar soyadı olarak 3 sütundan oluşan bir tablo şeklinde oluşturduk. Kitap tablomuzla devam edelim.

Sorgu 3: Kitap tablomuzu oluşturalım.

```
Use DbBookStore
Create Table TblBook
(
BookID int primary key identity(1,1),
BookName varchar(100),
BookAuthor int,
BookPage varchar(3),
BookPublisher int
)
```

Kitap tablomuzda kitabımızın ID değeri, adı, yazarı, sayfa sayısı ve yayınevi bilgisini tutacak olan sütunlarımızı tanımladık. Yazarı ve yayınevini ilişkili tablo üzerinden alacağımız için veri türünü int yaptık.

Sorgu 4: Yayınevi tablomuzu oluşturalım.

```
Use DbBookStore
Create Table TblPublisher
(
PublisherID int primary key identity(1,1),
PublisherName varchar(50),
)
```

Yayınevi tablomuza 2 tane sütun ekledik. Bu sütunlardan ilki yayınevinin ID bilgisi diğeri yayınevinin adını tutacak olan sütunlarımızdır. Tablolarımızı oluşturmaya devam edelim.

Sorgu 5: Üyeler tablomuzu oluşturalım.

```
Use DbBookStore
Create Table TblMember
(
MemberID int primary key identity(1,1),
MemberName varchar(50),
MemberSurname varchar(50),
MemberUsername varchar(50),
MemberPassword varchar(50),
MemberDepartment int
)
```

Üyeler tablomuza üye ID, üyemizin ad ve soyad bilgilerini tutacak olan `name`, `surname` sütunlarını ekledik. Kitabımızın gelecek bölümünde SQL'in nasıl C# dili ile kullanıldığını anlatacağız. Orada oluşturduğumuz sistem kullanıcı adı ve şifreyle açılacak. Bu sebeple üyelerimiz için bir kullanıcı adı ve şifre sütunu tanımladık. Son olarak üyemizin hangi bölümde öğrenim gördüğünü tutacak olan department sütunu dahil ettik. Bölümleri ayrı bir tablo olarak tutacağımız için bu kısmı ilişki içine alacağımız varsayarak int olarak tanımladık.

24

C# ile SQL Kullanımı

Bu Bölümde

Kitabımızın bu son bölümünde C# ile SQL veritabanını beraber kullanacağız. Geçen bölümde oluşturduğumuz veritabanını C# dilinde bir form tasarımı yapıp somut olarak uygulayacağız.

Bu bölümde ayrıca C# dilini, form uygulamaları geliştirmeyi, sınıf ve nesne yapısını, C# araçlarını, form tasarımını, özellikler penceresini, kullanıcı adı ve şifre ile giriş yapmayı, üyeye özel panel oluşturmayı ve daha pek çok başlığı öğreneceğiz.

GİRİŞ

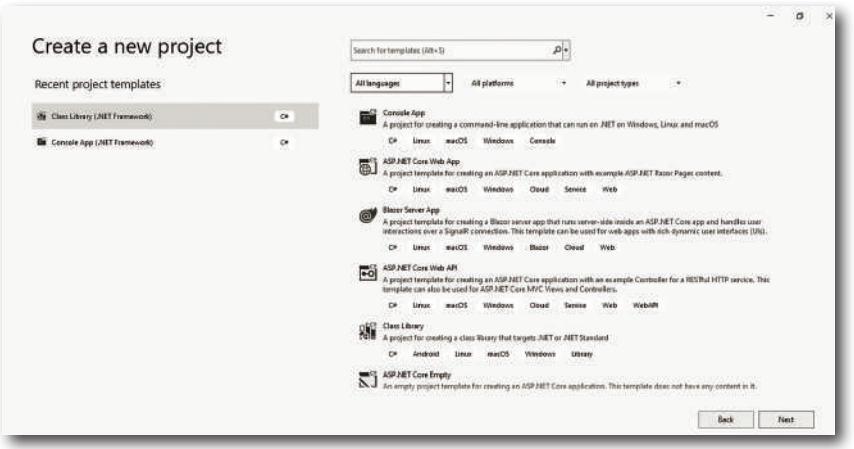
Özüne baktığınız zaman bu bir SQL kitabı ancak sizlere hem fikir vermesi hem de veriyi işlenmiş olarak nasıl kullanacağımızı gösterebilmek adına SQL'de öğrendiğimiz bu konuları C#'da bir uygulama üzerinden nasıl çalıştıracağımızı göstermek istedim. Umarım kitabımızın bu son bölümü de sizler için faydalı olur. Visual Studio kurulumunu anlatmaya gerek duymadım zira webde kolaylıkla bulabilirsiniz. Eğer kurulumda sorun yaşarsanız bana sosyal medya hesaplarımdan ulaştığınız takdirde size destek olacağım. Visual Studio 2022 sürümüyle çalışacağız. İlk olarak C# dilinden kısaca bahsedelim.

AMACIMIZ

Bu projede amacımız C# diliyle somut bir proje geliştirmektir. C# Microsoft tarafından geliştirilen ve dünyanın en çok kullanılan 5 programlama dilinden birisidir. C# ile geliştirme yapan geliştiricilerin ciddi bir kısmı veritabanı olarak MSSQL kullanmaktadır. Biz de burada MSSQL veritabanı alt yapısıyla C# dilini kullanarak örnek bir proje geliştireceğiz. Projemizde veritabanı olarak BookStore veritabanımızı kullanacağız. O halde sözü daha fazla uzatmadan Visual Studioyu açalım ve uygulamamızı geliştirmeye başlayalım.

Visual Studioyu açtığınız zaman karşınıza yukarıdaki gibi bir ekran gelecektir. Burada daha önce oluşturduğunuz projeleri görebilirsiniz. Sol üst kısımda yer alan **Recent** kutusundan kendi projenizi aratabilirsiniz. C# ve Visual Studio hakkında kitap üzerinden çalışma yapmak isterseniz **C# Eğitim Kitabı**'mızı takip edebilirsiniz. Sağ tarafta yer alan **Create New Project** alanına tıklayalım.

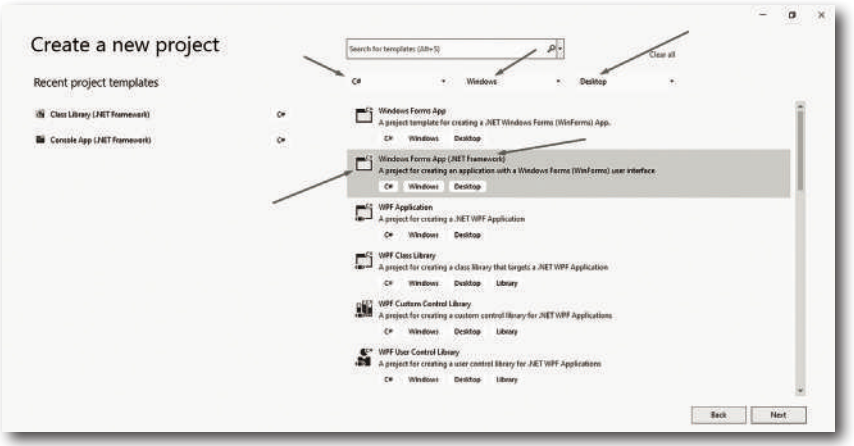
Karşımıza şöyle bir ekran gelecektir.



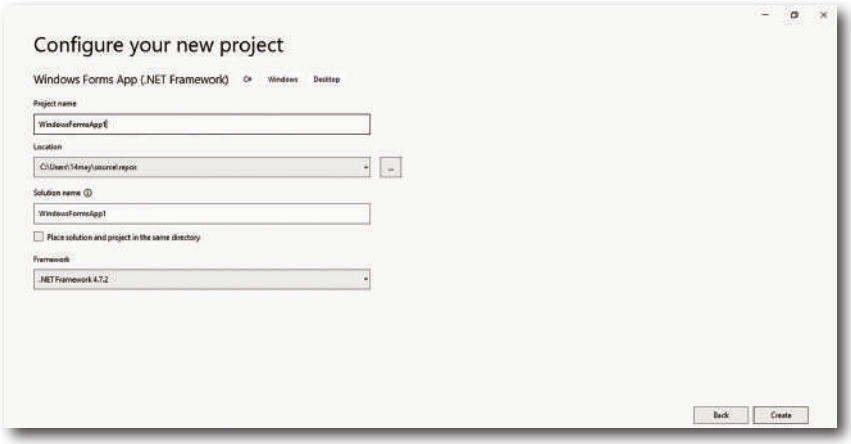
Bu ekranda **All Launguages** kısmından C# dilini seçelim.

All Platforms kısmında **Windows'u** seçelim.

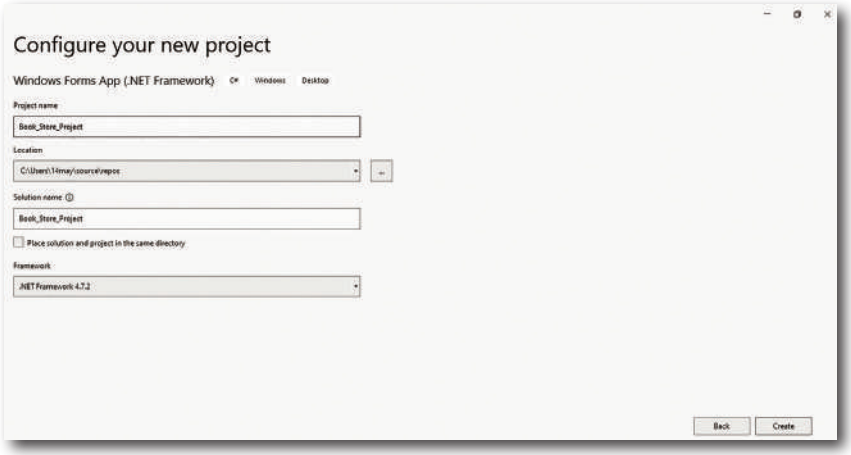
All Project Types kısmından ise **Desktop** yani masaüstü seçeneğini seçelim.



Filtreleme işleminden sonra karşımıza yukarıdaki gibi bir ekran gelecektir. Bu ekrandan **Windows Form App** yazan kısmı seçelim. 2 tane **Windows Form App** seçeneği bulunuyor. Bunlar içinde sağ tarafında **.Net Framework** yazan seçeneği seçelim. Bu kısım önemli. Framework bir geliştirme çatısıdır. Biz burada **.Net Framework** isimli çatı üzerinde geliştirme yapacağız. Seçtikten sonra **Next** butonuna tıklayalım.



Karşımıza yukarıdaki gibi bir ekran gelecektir. Bu ekran üzerinde projemize isim verebiliriz. Projemizi aşağıdaki gibi isimlendirelim.



Book_Store_Project olarak adlandığımız projemizi sağ alt taraftaki **Create** butonu ile oluşturalım. Projemizi oluşturduğumuz zaman karşımıza aşağıdaki gibi bir ekran gelecektir. (bir sonraki sayfada inceleyebilirsiniz)

Son Söz

Kitabın sonuna geldiğiniz için öncelikle sizi en içten dileklerimle kutluyor ve bu istikrarlı adımlarınızın bir ömür boyu devam etmesini temenni ediyorum.

Öğrenme yolculuğu hiç kimse için bitmeyecek bir yolculuktur. Bu yolculukta sizler için bir adım olabildiysem ne mutlu bana. Yaşamınız boyunca güçlü öğrenme arzunuzun devam etmesini, kararlı yapınızla karşınıza çıkabilecek bütün sorunları kolayca aşabilmenizi diliyorum. Yolunuz açık, başarınız daim olsun.

2023, Murat YÜCEDAĞ

İstanbul - Türkiye

Murat YÜCEDAĞ

muratyucedag.com

facebook.com/murattyucedag

twitter.com/murattyucedag

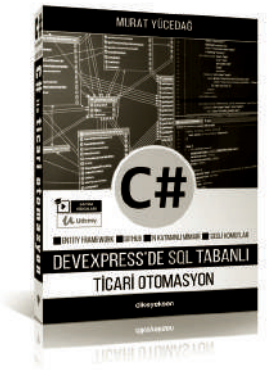
linkedin.com/in/murat-yücedağ-186933149

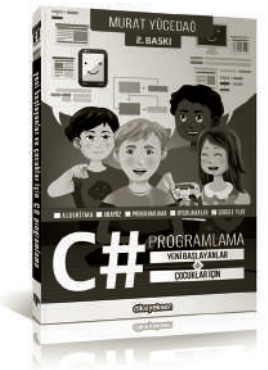
udemy.com/user/murat-yucedag-3

youtube.com/user/YazilimHerYerde

github.com/MuratYucedag

yucedagmurat23@gmail.com





Murat YÜCEDAĞ

Tüm Kitap ve Setler