**STORED PROCEDURE**

Stored Procedure’ler, veri tabanınızı daha kolay yönetmenizi ve veri tabanı hakkındaki bilgilerin daha kolay görüntülenmesini sağlar. SQL ifadelerinin önceden derlenmiş şekli olarak karşımıza çıkarlar. Veri tabanı içerisinde saklanırlar ve uygulamanızdan bir çağrı yaptığınızda işlevsellik kazanırlar.  Sizin tarafınızdan tanımlanmış belli  şartlara göre program akışı üzerinde etkinlik gösterirler. Stored Procedure’ler mantıksal ifadeler, veri tabanı sorgulama cümleleri içerebilir. Dışarıdan parametreler alabilirler ve geriye değer veya değer kümesi döndürebilirler.  Stored Procedure’ler sayesinde birden çok kez yazmak  zorunda kaldığımız uzun sql cümleciklerini, sadece bir kez yazıp bunu stored procedure içerisinde saklayarak, daha sonraki kullanımlarda da sadece kaydettiğimiz stored procedure’nin ismini referans göstererek kullanabiliriz. Stored Procedure’lerin bu kullanımı, daha önceden yazılıp derlendiği için yazılan uygulamanın hızını arttıracak; aynı ifadeleri bir daha yazma ihtiyacı duyduğumuzda, sadece bu prosedürün ismini referans göstereceğimizden dolayı da, uygulamamızın boyutunu azaltacaktır.

 Yazılım geliştiriciler, veri tabanı kullanan programlarını geliştirirken iki farklı yöntem izleyebilirler. Bunlardan ilki, programcı  veri tabanından bilgi çekerken kullanacağı tüm ifadeleri programının içerisine yerleştirir. Server üzerinde veri tabanı, ve local tarafta da veri tabanı sorgulamalarını içeren uygulama yer alır. Program bir bilgiye ihtiyaç duyduğunda localde tutulan sql ifadesini, veri tabanına gönderir, veri tabanı da verilen talebe uygun cevapları bir recordset (kayıt kümesi) biçiminde kullanıcıya döndürür.  Her hangi bir hata olması durumunda,  program kırıldığında kullanıcı veri tabanından bilgi çekilirken kullanılan sql cümlelerini görebilir. Bu istenmeyen durumdur. Diğer bir durumda kullanıcıların görüntülemeye yetkisi olmadığı tablolar üzerine sorgulama yapmalarıdır. Bu da diğer bir istenmeyen durumdur. Dolayısı ile etkin ve güvenli bir uygulama açısından bu yöntem tercih edilmemelidir. Peki alternatif nedir?  Alternatif, yani yukarıda varlığından bahsettiğimiz ikinci yöntem, veri tabanı üzerinde yapılacak mümkünse tüm, mümkün değilse güvenlik ve/veya hız açısından önemli olan tüm veri tabanı sorgulamalarını stored procedure’ler yoluyla yapmaktır. Bu sayede veri tabanı üzerinde bazı kayıtları görüntüleme yetkisi olmayan kimselerin, yapacakları sorgulamalarda önlenmiş olur.

Stored Procedure’leri kullanmanın diğer bir avantajı ise, ağ trafiğini azaltmalarıdır. Veri tabanı üzerinde bir sorgulama işlemi gerçekleştirirken, sorgulama ifadesini client bilgisayardan, server’a göndermek ağ trafiğini arttırır. Ama stored procedure kullanılmışsa, client sadece stored procedure’nin ismini vererek uzun kod kümesine çağrı gönderecek, tüm işlemler server üzerinde gerçekleşecek ve sadece sonuç ağ üzerinden client’a döndürülecektir. Bu sayede, sql sorgularının ağ trafiği üzerindeki etkisi oldukça azalır.

"Stored Procedure Neden Kullanılmalı?" gibi bir sorunun cevabını şu şekilde verebiliriz. (Bu aynı zamanda Stored Procedure lerin faydaları başlığınıda alacaktır :) )  
  
- Çalışma anı planlama sağlar ve tekrar tekrar kullanılabilir  
- Query lerinize otomatik parametrelendirme getirir  
- Uygulamalar arasında ortak kullanılabilir yapıdadır  
- Güvenli data modifikasyonu sağlar  
- Network bandwidth inden tasaruuf sağlar (daha az network bandwidth kaynak kullanımı)  
- Zamanlanmış görevlere eklenebilir  
- Job olarak tanımlanabilir ve schedule edilebilir  
- Database objelerine güvenli erişim olanağı tanır

Oluşturulma şekillerine göre Stored Procedure’ler dört çeşittir.

**System Stored Procedure’ler**; SQL Server üzerinde  **master** isimli veri tabanında saklanan prosedürlerdir. Default olarak Server’ı kurduğunuzda yüklenirler, herhangi bir veri tabanı üzerinde işlem yaparken kullanılabilirler. Diğer bir deyişle, herhangi bir veri tabanına direk olarak  bağlı değillerdir. Bunlarda yine kendi içlerinde birkaç gruba ayrılırlar. Mesela SQL Server’ların kaydı sırasında kullanılan Active Directory Procedure’ler; ODBC uygulamalarını sistem tabloları üzerindeki değişikliklerden izole eden ve ODBC veri işleme prosedürlerini barındıran,Catalog Procedure’ler; kursör değişkenin fonksiyonlarını yerine getiren, Cursor Procedure’ler, Full-Text Search Procedure’ler, Database Maintenance Plan Procedure’ler, Security Procedure’ler, SQL Mail Procedure’ler, XML Procedure’ler, Web Assistant Procedure’ler.

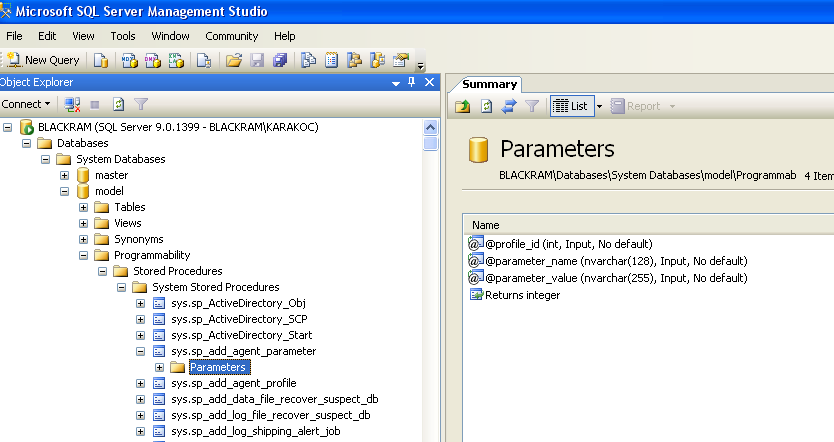
**Extended Stored Procedure:** Genellikle “dll” dosya şeklinde derlenmiş prosedürlerdir. T-SQL dışındaki dillerde yazılırlar. Adları “xp\_” ile başlar. Ama “sp\_” ile başlayan prosedürler de vardır. Master veri tabanında tutulurlar ve tam adları yazılarak kullanılır. “master.dbo.xp\_cmdshell” gibi.

**CLR Stored Procedure:** Herhangi bir CLR programlama dilini kullanarak Stored Procedure oluşturulabilir. TSQL’in desteklemediği kod yazımlarında bu programlama dilleriyle kodlar yazılır. Güvenlik ve denetim gibi konularda CLR Stored Procedure’ler SQL Server için avantaj sağlar.

**Kullanıcı Tanımlı Stored Procedure :** local veya global olarak oluşturulabilirler, bir tane sayı işareti (#) ile başlayan local prosedürler, bir anda sadece bir kullanıcı tarafından kullanılabilirler. İki tane sayı işareti ile başlayan (##) global prosedürler ise herkes tarafından görülebilir. Programcının yazdığı prosedürlerdir. Geçerli oldukları yere göre üç tiptir.

* Geçici
* Yerel
* Uzak

**Local Stored Procedure’ler;** Server’dan bağımsız olarak, bir veri tabanı üzerinde işlem yapabilen biz, yazılım geliştiricilerin oluşturduğu prosedürlerdir. Bu konumuzda bizim asıl ilgileneceğimiz, prosedür tipi budur.  Bu prosedürler, SQL Server üzerinde oluşturuldukları veri tabanının altında Programmability🡪Stored Procedures dizininin altında saklanırlar.



**Çalıştırılma Aşamaları**

Bir Stored Procedure’ün çalıştırılma aşamaları herhangi bir Query’nin çalıştırılma aşamalarıyla aynıdır.

* Stored Procedure’ün bileşenleri parçalara ayrıştırılır (Parsing).
* Veri tabanı içerisinde table, view gibi başka nesnelere gönderme yapan referanslar varsa, geçerli olup olmadıkları (nesnenin ve iznin olup olmaması) kontrol edilir.
* Kontrollerden geçen Stored Procedure’ün adı sysobjects tablosuna, kodları ise syscomments tablosuna kaydedilir (Compilig).
* Bu işlemlerle birlikte derleme işlemi yapılır. Normalizasyon işlemleri olarak da anılan bu işlemler sonucunda, ağaç şeması elde edilir. Bu şema da sysprocedures tablosunda saklanır.

Stored Procedure herhangi bir anda çağrıldığında, ilk kez çalışıyorsa bu işlemler gerçekleştirilir. İlk defa çağrılmıyorsa, kontrol, sorgulama ağacı oluşturma işlemleri yapılmaz ve oldukça hızlı bir şekilde Stored Procedure’ün derlenmiş hali çalışır. Bundan dolayı Stored Procedure’ler derlenen nesnelerden biri olarak bilinir.

Bir veri tabanının altında bir çok Stored Procedure kayıtlı. Bunlardan tipi “**system**” olanlar, sistem tarafından otomatik olarak oluşturulmuş, başka bir deyişle master altındaki sistem prosedürlerinden örnek olarak alınmış prosedürlerdir. Tipi “**User**” olan prosedürler ise, kullanıcı (database administrator- veri tabanı yöneticisi veya yazılım geliştirici) tarafından yazılmış olanlar prosedürlerdir. Prosedürün ‘**owne**r’ı diğer bir deyişler, Stored Procedure’nin sahibi, **dbo** (database owner- veri tabanının sahibi)’dur.

Herhangi bir stored procedure üzerinde mouse’un sağ tuşuna basıp, stored procedure’nin özelliklerine (properties) bakacak olursak bu prosedürün ne iş yaptığını görebiliriz. Eğer Stored Procedure’nin tipi ‘system’ ise  kodlar üzerinde bir değişiklik yapamayız.

**Stored Procedure Oluşturmak:** Stored Procedure’ün oluşturulma şekli aşağıdaki gibidir.

CREATE PROC [ EDURE ] prosedür\_adi

AS

T-SQL ifadeleri

GO

Oluşturacağınız Stored Procedure’ler ile sistemin Stored Procedure’lerinin karışmaması için prosedür adlarının önüne “U” harfini ekleyiniz. Böylece, oluşturduğunuz Stored Procedure’leri sistem Stored Procedure’lerinden ayırt etmiş olursunuz.

Bir Stored Procedure CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER ve CREATE VIEW ifadelerini içeremez. Ancak, her nesneden veri alabilir.

**Örnek**: Bir sınıfta bulunan öğrencilerin, öğrenci numaraları, adları, soyadları ve final notları bir tabloda tutulmaktadır.



Final notları 55’ten büyük olan notları gösterecek bir Stored Procedure oluşturmak istediğiniz düşününüz.

* Veri tabanınızın Programmability klasörünün solundaki + işaretine tıkladığınızda açılan Stored Procedures klasörü üzerinde farenizle sağ tıklayınız.
* Yeni bir Stored Procedure oluşturabilmeniz için New Stored Procedure komutunu tıklatınız.

Komutu tıkladığınızda Stored Procedure’ün genel yapısını bulunduran bir Query sayfası açılacaktır.



* Stored Procedure’ün yapısı için gerekli yazım kuralı verilmişti. Ekrana gelen Query sayfasında size gerekli olmayacak satırları silebilir veya Query’nin tamamını silip kendiniz Stored Procedure’ü oluşturabilirsiniz.
* Final notu 55’ten büyük olan notları görüntüleyecek satırları aşağıdaki gibi yazabilirsiniz.



* F5 tuşuna basarak oluşturduğunuz Stored Procedure’ün doğruluğunu sınayınız ve hata varsa düzeltiniz.
* Oluşturduğunuz Stored Procedure’ü görmek için Stored Procedures üzerinde fareyle sağ tıklayarak Refresh komutunu veriniz.
* Oluşturulan Stored Procedure’ü çalıştırabilmek için veri tabanınız üzerinde fare sağ tuşuyla açılan menüden New Query komutunu veriniz.

* Tablonuzdaki verilerin tamamını ve final notu 55’ten büyük olan öğrencileri görebilmek için aşağıdaki gibi ilgili kodları yazınız.



* EXEC komutu oluşturduğunuz Stored Procedure’ü çalıştıracak olan komuttur. EXEC’ten sonra yazılan ifade ise oluşturduğunuz Stored Procedure’ün adıdır.
* Yazdığınız bu Query’i çalıştırarak (F5) sonucu görebilirsiniz.

**NOCOUNT Parametresi**

Sorgunun çalıştırılması sonucunda kaç kaydın bu işlemden etkilendiğini gösterir veya göstermez. SET NOCOUNT ON şeklinde sorguya yazılırsa kaç kaydın bu işlemden etkilendiği gösterilmeyecektir. SET NOCOUNT OFF şeklinde bir sorgu yazımındaysa bu işlemden kaç kaydın etkilendiği gösterilecektir. Varsayılan olarak bu parametre OFF konumundadır. Yani etkilenen kayıtlar gösterilmektedir. Yukarıdaki Stored Procedure oluşturma örneğinde sonuçların gösterildiği Results sekmesinin yanında bulunan Messages sekmesinde etkilenen kayıtları görebilirsiniz. Eğer, etkilenen kayıtların gösterilmesini istemiyorsanız Stored Procedure’ü oluştururken SET NOCOUNT ON satırını eklemelisiniz.



Etkilenen kayıtların gösterilmesi

**Stored Procedure’ün Kullanım Hakkı**

Stored Procedure’ü oluşturduktan sonra çalıştırılabilmesi için kullanıcılara izin verilmiş olması gerekir. Eğer, bir kullanıcının izni varsa o Stored Procedure’ü kullanabilecektir.

İzin verme yetkileri DENY ve GRANT ile sağlanmaktadır. DENY ile belirtilen kullanıcı Stored Procedure’ü kullanamayacaktır. GRANT ile belirtilen kullanıcıysa Stored Procedure’ü kullanabilecektir.

DENY ON UNotlar TO guest1

GRANT ON UNotlar TO ali

**Değişiklik Yapmak🡪**Stored Procedure’lerde değişiklik yapmak ALTER ile gerçekleşir.

ALTER PROC [EDURE] prosedür\_adı

AS

T-SQL ifadeleri

GO

Bir Stored Procedure’de değişiklik yapabilmek için önce kaynak kodunun alınması ve bir Query ekranına kopyalanıp düzenlenmesi gerekir.

**Örnek:**Stored Procedure oluşturmayla ilgili yapılan örnekte final notları 55’ten büyük olan notları göstermektedir. Bu prosedürü final notları 55 ve 55’ten küçük olan final notları şeklinde değiştirelim.

Oluşturulmuş bir Stored Procedure’de değişiklik yapabilmenin yani kaynak kodlarını alabilmenin iki yolu vardır:

1. Management Studio’da oluşturulmuş Stored Procedure üzerinde fareyle sağ tıklanarak açılan menüden Modify komutunu vermek,
2. “sp\_helptext prosedür\_adı” komutunu kullanarak elde edilen kodları kopyalayıp yeni bir Query’e yapıştırmaktır.



**Management Studio ile Değişiklik Yapmak**

Oluşturduğunuz “UNotlar” adındaki Stored Procedure üzerinde fareyle sağ tıklatınız ve açılan menüden Modify komutunu veriniz.

Modify komutunu verince kodlar bir Query sayfası şeklinde ekrana gelecektir. Üzerinde gerekli değişiklikleri yapınız ve Execute (F5) ediniz.





Düzenlenmiş Stored Procedure

Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinde görebilirsiniz.



Yeni bir Query ekranı açınız ve Stored Procedure’ün çalışması için gerekli kodu yazınız.



UNotlar Stored Procedure’ü işletilecek ve sonuçlar Results penceresinde size gösterilecektir.

 **“sp\_helptext” İfadesi ile Değişiklik Yapmak**

 “UNotlar” Stored Procedure’ündeki kodları görebilmek için yeni bir Query’de “sp\_helptext” ifadesini Aşağıdaki gibi kullanınız.

“UNotlar” Stored Procedure’deki kodlar Results penceresinde size gösterilecektir.

Kodları seçerek kopyalayınız ve Query sayfasındaki sp\_helptext ifadesini silerek yapıştırınız.



Kodlardaki CREATE komutunu silerek ALTER komutunu yazınız.



Stored Procedure’ü çalıştırınız (F5). Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinden görebilirsiniz.



Yine, Query sayfasında yazılan Stored Procedure kodlarını siliniz ve sonuçların gösterilmesi için gerekli olan kodları yazınız.



Elde edilen sonuçları Results penceresinde görebilirsiniz.



Bir Stored Procedure’de çok fazla değişiklikler yapılıyorsa EXEC komutuyla çalıştırılırken yeniden derlenmesi istenebilir. Bunun için, RECOMPILE kullanılmalıdır.

**Örnek:** EXEC UNotlar WITH RECOMPILE

**Stored Procedure’ü Silmek**

Var olan bir Stored Procedure’ü silmek için DROP komutunu kullanmak gereklidir. DROP komutundan sonra Stored Procedure’ün sahibinin adı ve Stored Procedure’ün adı yazılmalıdır.

Genel Kullanımı🡪 DROP PROC sahip.prosedür\_adı

**Örnek:** DROP PROC dbo.UNotlar

**Değer Alan Stored Procedure’ler**

Stored Procedure’lerin daha etkin kullanılabilmesi ve işlevsel bir hale gelebilmesi için dışarıdan değer almalarına ihtiyaç duyulur. Bu nedenle girdi parametreleri (Input Parameter) kullanılır. Stored Procedure’nin aldığı değer Query’den gelen değerdir. Gönderilen değeri karşılayacak bir değişken Stored Procedure’de tanımlanmalıdır.

**Örnek:** Bir öğrenciye ait 3 not bilgisi Query’de ilk değerleri atanarak not ortalamaları hesaplanacaktır. Hesaplanan not ortalamasına göre öğrencinin başarılı olup olmadığı Stored Procedure’de belirlenecek ve mesaj olarak yazdırılacaktır. Bunun için, yeni bir sorgu sayfası açınız ve öğrencinin 3 notunun tutulacağı değişkenleri tanımlayıp ilk değerlerini aşağıdaki gibi veriniz.



İlk değerleri verilen değişkenlerin ortalamasını “@ortalama” değişkenine hesaplatıp aşağıdaki gibi aktarınız.



Elde edilen değeri oluşturacağınız Stored Procedure’e EXEC komutuyla aşağıdaki gönderiniz. Böylece hem değer gönderilecek hem de Stored Procedure çalıştırılmış olacaktır.



Hesaplamaların yapılacağı Stored Procedure’ü yazmak için yeni bir Query sayfası açınız. Stored Procedure’ün adını “UHesapla” olarak belirterek ortalamayı karşılayacak değişkeni tanımlayınız ve başarının belirleneceği “if” yapısını yazınız.



Yazdığınız Stored Procedure’ü F5 ile çalıştırarak doğruluğunu kontrol ediniz.



Daha sonra ilk yazdığınız sorgunuza dönerek çalıştırınız. “UHesapla” Stored Procedure’ü çalıştırılacak ve sonucu Messages penceresinde gösterilecektir.



Stored Procedure’e gelen parametrelerin isteğe bağlı olması istenebilir. Bu gibi durumlarda Stored Procedure’de tanımlanan değişkene default değer atanması yapılır. Stored Procedure’e gelen parametreye değer atanmazsa, tanımlanan değişkene atanan default değer işleme tabi tutulur. Stored Procedure’de bir default değer atayacaksanız bu değer, bir sabit olması gerekir.

**Örnek:** Bir sınıftaki öğrencilerin bilgilerinin tutulduğu bir tabloda öğrenci adlarının içerisinde “a” harfi geçen öğrencileri gösteren bir Stored Procedure şöyle yazılmalıdır.





“Ogr\_Ara” Stored Procedure’ü

Stored Procedure’de tanımlanan “ara” değişkeni Query’den bir değer gelmese de NULL değerini alacak ve işlem gerçekleşecektir.

Query’den öğrenci adlarının içerisinde “a” harfi olan öğrencileri görmek için aşağıdaki gibi kod satırını yazınız ve çalıştırınız.



Ogr\_Ara ‘a’ şeklindeki yazım ile de Stored Procedure çalışacaktır. Sonuçları Results penceresinde görebilirsiniz.



Stored Procedure’e değer gönderilmeseydi Messages penceresinde “Command(s) completed successfully.” mesajını görecektiniz.

**Değer Alıp-Veren Stored Procedure’ler**

Değer alıp-veren saklı prosedürler, Query’den gönderilen değerleri alıp istenilen işleme tabi tuttuktan sonra elde edilen sonucu tekrar Query’e gönderen prosedürlerdir. Bu işlem için OUTPUT parametresi kullanılır.

**Örnek** Tanımlanan ve ilk değerleri verilen iki sayıyı Stored Procedure’e göndererek toplamlarını Stored Procedure’de yapıp sonucu gönderildiği yerde görüntüleyen kodları yazmak için aşağıda verilen adımları uygulayınız.

Yeni bir Query sayfası açınız. Stored Procedure’ün adını “UTopla” olarak belirleyiniz. Query’den gelecek değerler için kullanılacak iki sayı değişkeni, toplamları ve geriye değer döndürmesi için toplam değişkenini tanımlayınız.



Gelen sayıları “tpl” değişkeninde toplatınız.



Oluşturduğunuz prosedürü çalıştırınız ve hata olmadığından emin olunuz.

Yeni bir Query sayfası açınız. Toplam için kullanılacak iki sayı değişkeni, toplamları için toplam değişkenini tanımlayınız ve ilk değerlerini atayınız.



EXEC komutuyla, Stored Procedure’e göndereceğiniz değişkenleri ve sonucu alacak “toplam” değişkenini de OUTPUT parametresi ile yazınız.



Stored Procedure’den dönen değeri de Results penceresinde görmek için SELECT komutuyla yazdırınız.



Query’nizi F5 ile çalıştırınız. Sonucu Results penceresinde göreceksiniz.



Query’den gönderilen “sayi1” ve “sayi2” değişkenlerini Stored Procedure’de “s1” ve “s2”, “toplam” değişkenini de “tpl” değişkeni karşılamaktadır.

“Toplam” ve “tpl” değişkenleri OUTPUT parametresiyle tanımlandığından sonuç “tpl” değişkeninden “toplam” değişkenine gönderilmektedir.

**RETURN Deyimi**

Prosedürden tek bir tamsayı değer döndürmek için bir başka seçenek olarak RETURN ifadesi kullanılabilir. Bu şekilde OUTPUT parametresini hem prosedür içinde hem de prosedürün çağrıldığı yerde tanımlamak zorunda kalınmadan doğrudan değer döndürülebilir. Bir uygulamada hem RETURN hem de OUTPUT parametresi aynı anda kullanılabilir.

**Örnek:** Değer alıp veren Stored Procedure’ler örneğini RETURN deyimi ile de yapılabilir. Örneği adım adım uygulayınız.

Yeni bir Query sayfası açınız. Stored Procedure’ün adını “URet\_Topla” olarak belirleyiniz. Query’den gelecek değerler için kullanılacak iki sayı değişkeni için Stored Procedure’de iki değişkeni tanımlayınız.



Yapılacak işlem sonucunda elde edilen değerin Query’e gönderilmesi için RETURN deyimiyle beraber yapılacak işlemi belirtiniz.



Oluşturduğunuz Stored Procedure’ü çalıştırarak hata olmadığından emin olunuz. Hata yoksa Stored Procedure oluşturulmuş demektir.

Yeni bir Query sayfası açınız. Kullanılacak değişkenleri belirleyiniz ve ilk değerlerini atayınız.



Birinci ve ikinci sayıyı URet\_Topla Stored Procedure’üne gönderecek ve dönen değeri toplam değişkenine aktaracak EXEC satırını yazınız.



Sonucun Results penceresinde görüntülenmesi için gerekli kod satırını yazınız



Query’nizi çalıştırarak hata olmadığından emin olunuz. Eğer hata yoksa elde edilen sonucu görebilirsiniz.



