

Mustafa Girgin
21290649

<https://github.com/antelcha/sqlprojeler>

Veritabanı Güvenliği ve Erişim Kontrolü

HumanResources Şeması İçin Güvenli Kullanıcı Tanımlama

AdventureWorks veritabanı üzerinde temel güvenlik önlemlerini adım adım uygulayarak, HumanResources şemasına özel bir kullanıcı tanımladım. Buradaki hedefim, yalnızca ihtiyaç duyulan en temel yetkileri vererek, sistemin hem işlevselliğini hem de güvenliğini korumaktı.

1. Sunucu Seviyesinde Login Oluşturulması

İlk olarak, sisteme kimlerin bağlanabileceğini kontrol altına almak gerekiyor. Bu nedenle, SQL Server Authentication kullanarak HR_Login isimli bir kimlik tanımladım. Bu login henüz herhangi bir veritabanında işlem yapma yetkisine sahip değil; sadece sunucuya erişim izni var.

```
1  
2 CREATE LOGIN HR_Login  
3 WITH PASSWORD = 'HrLogin1!';  
4  
5
```

2. Veritabanı Seviyesinde User Tanımlanması

Sunucuya bağlanma yetkisi olan HR_Login'in, AdventureWorks veritabanı içerisinde işlem yapabilmesi için burada bir kullanıcıya (user) dönüştürülmesi gerekiyordu. Bu yüzden önce veritabanını seçtim, ardından **HR_User** adında bir kullanıcı oluşturdum. Henüz herhangi bir

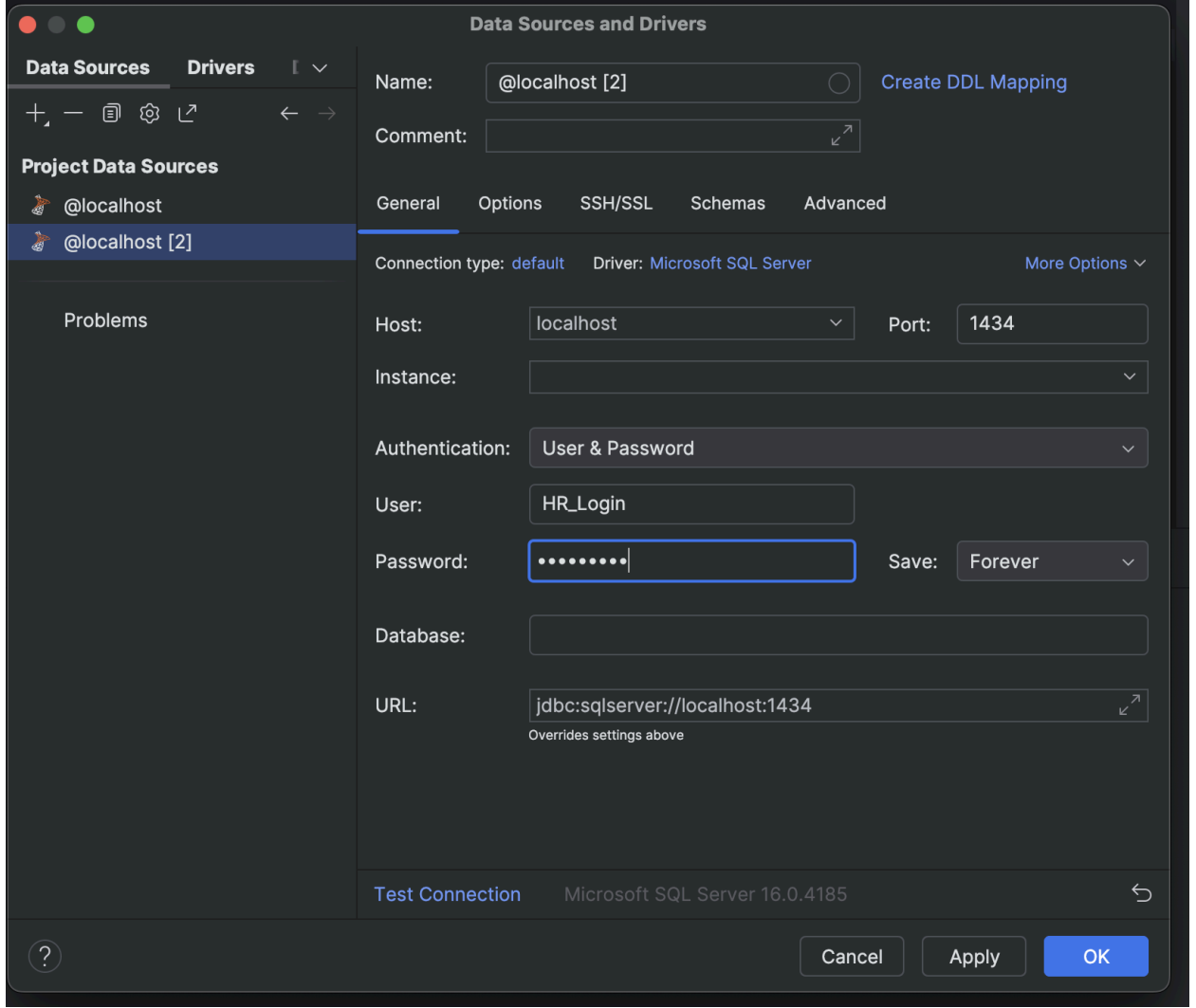
tabloya erişimi yok.

```
1  USE AdventureWorks;
2  CREATE USER HR_User FOR LOGIN HR_Login;
3  GO
4
5  |
```

3. Yetkilendirme (GRANT)

HR_User'ın asıl amacı HumanResources şemasındaki verilere erişim sağlamak olduğu için, bu kullanıcıya yalnızca SELECT (okuma) yetkisi verdim. Böylece kullanıcı verileri görüntüleyebilecek, ancak değiştiremeyecek. Yetkilendirme sürecini **GRANT SELECT ON SCHEMA::HumanResources TO HR_User** şeklinde tamamladım.

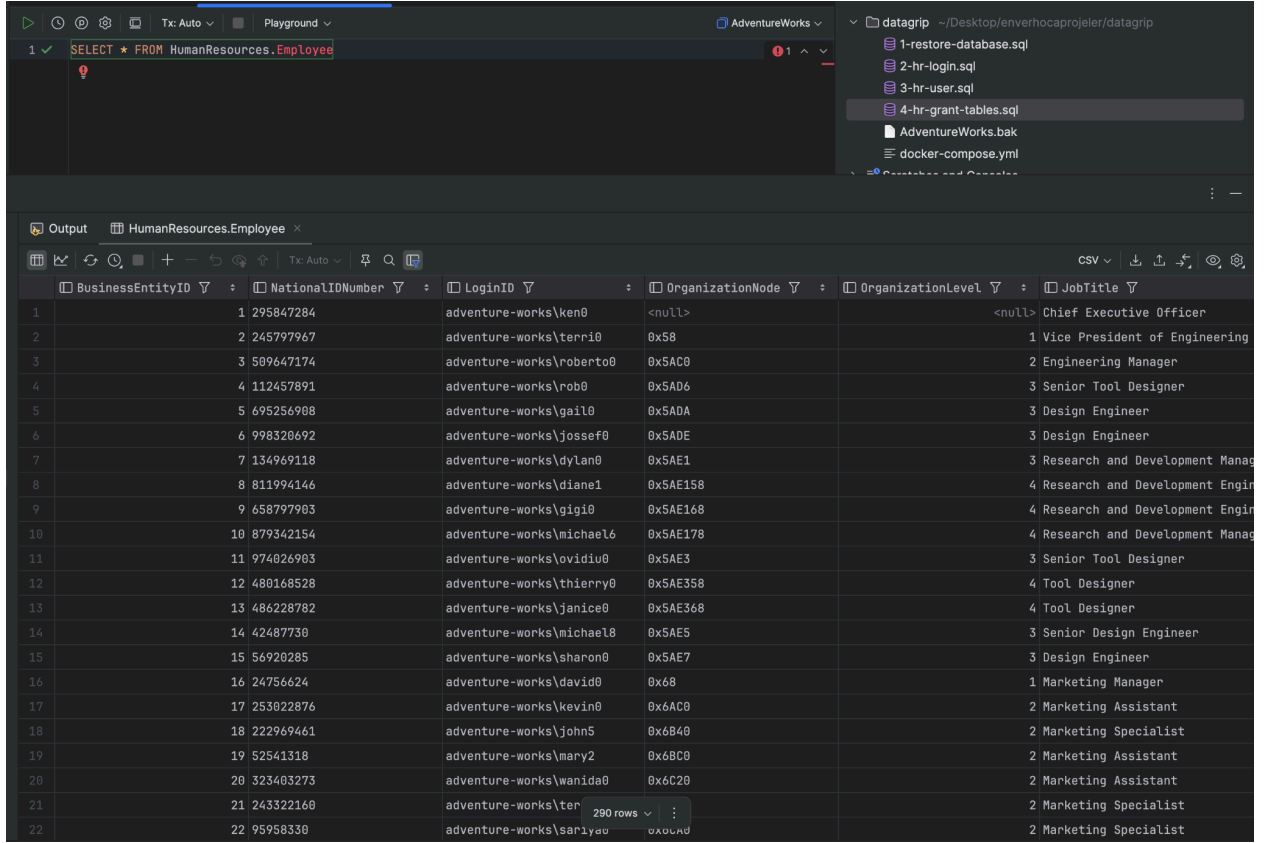
```
1  GRANT SELECT ON SCHEMA :: HumanResources TO HR_User;
2  GO
3
4
```



4. Yapılandırmanın Test Edilmesi

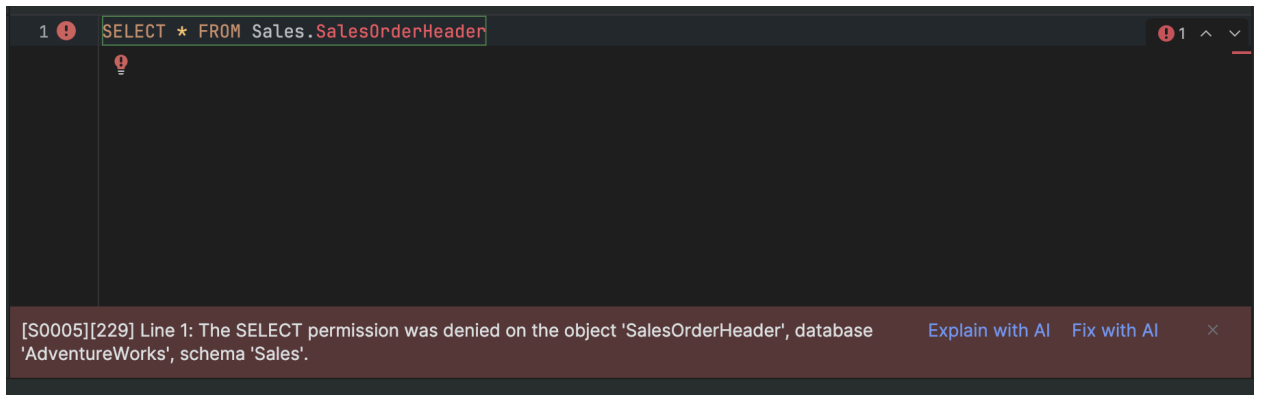
Tanımladığım yapıların beklendiği gibi çalıştığından emin olmak için, DataGrip üzerinden HR_Login ile bağlantı kurdum ve bazı testler yaptım:

1. **Yetkili erişim testi:** HumanResources şemasındaki bir tabloya (örneğin Employee) SELECT sorgusu gönderdim, veri başarılı şekilde geldi.

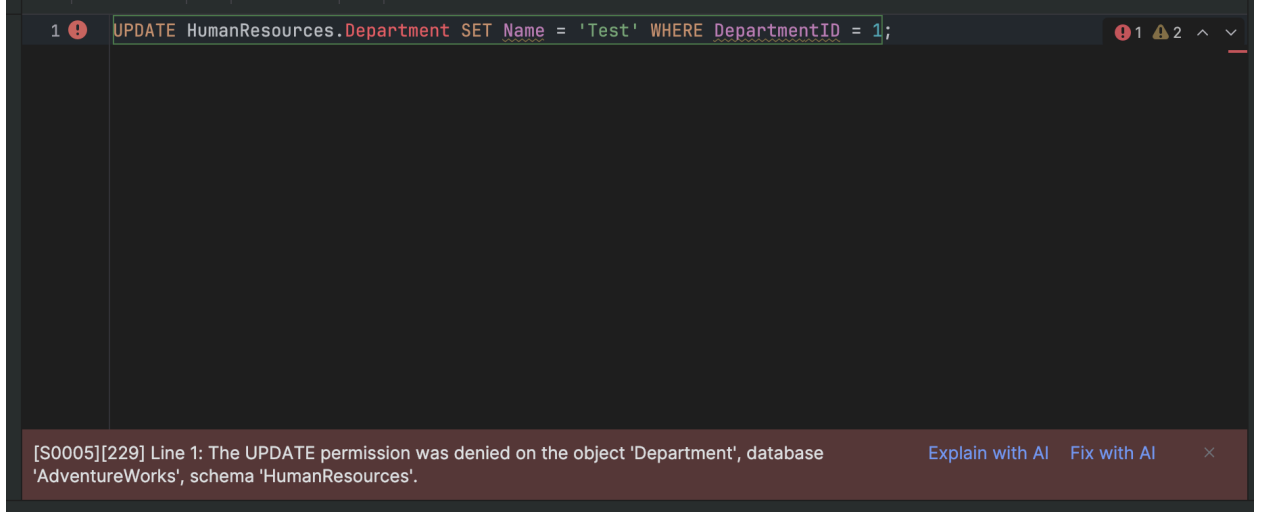


	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle
1	1	295847284	adventure-works\ken0	<null>	<null>	Chief Executive Officer
2	2	245797967	adventure-works\terri0	0x58		1 Vice President of Engineering
3	3	509647174	adventure-works\roberto0	0x5AC0		2 Engineering Manager
4	4	112457891	adventure-works\rob0	0x5AD6		3 Senior Tool Designer
5	5	695256908	adventure-works\gail0	0x5ADA		3 Design Engineer
6	6	998320692	adventure-works\jossef0	0x5ADE		3 Design Engineer
7	7	134969118	adventure-works\dylan0	0x5AE1		3 Research and Development Manag
8	8	811994146	adventure-works\diane1	0x5AE158		4 Research and Development Engin
9	9	658797903	adventure-works\gigi0	0x5AE168		4 Research and Development Engin
10	10	879342154	adventure-works\michael6	0x5AE178		4 Research and Development Manag
11	11	974026903	adventure-works\ovidiu0	0x5AE3		3 Senior Tool Designer
12	12	480168528	adventure-works\thierry0	0x5AE358		4 Tool Designer
13	13	486228782	adventure-works\janice0	0x5AE368		4 Tool Designer
14	14	42487730	adventure-works\michael8	0x5AE5		3 Senior Design Engineer
15	15	56920285	adventure-works\sharon0	0x5AE7		3 Design Engineer
16	16	24756624	adventure-works\david0	0x68		1 Marketing Manager
17	17	253022876	adventure-works\kevin0	0x6AC0		2 Marketing Assistant
18	18	222969461	adventure-works\john5	0x6B40		2 Marketing Specialist
19	19	52541318	adventure-works\mary2	0x6BC0		2 Marketing Assistant
20	20	323403273	adventure-works\wanida0	0x6C20		2 Marketing Assistant
21	21	243322160	adventure-works\ter			2 Marketing Specialist
22	22	95958330	adventure-works\sariyao			2 Marketing Specialist

- Yetkisiz erişim denemesi (okuma):** Sales şemasındaki bir tabloya sorgu gönderdim. “SELECT permission was denied...” hatasıyla karşılaştım.



- Yetkisiz erişim denemesi (yazma):** HumanResources.Employee tablosuna INSERT ya da UPDATE sorgusu göndermeyi denedim. Beklendiği gibi işlem reddedildi.

A screenshot of a SQL query execution window. The query bar at the top contains the text: `UPDATE HumanResources.Department SET Name = 'Test' WHERE DepartmentID = 1;`. Below the query bar, a large empty space represents the results area. At the bottom, a red error message bar displays the text: `[S0005][229] Line 1: The UPDATE permission was denied on the object 'Department', database 'AdventureWorks', schema 'HumanResources'.` To the right of the error message, there are two links: `Explain with AI` and `Fix with AI`, followed by a close button (X).

Bu testler, kullanıcıya verdiğim yetkilerin yalnızca gereken işlemlerle sınırlı olduğunu gösterdi.

SQL Injection Zafiyeti ve Korunma Yöntemleri

1. Giriş

Bu bölümde, SQL Injection zafiyetinin ne olduğunu ve bu tür saldırılara karşı nasıl önlem alınabileceğini örneklerle ele alıyorum. SQL Injection, kullanıcıdan alınan verilerin yeterince kontrol edilmemesi durumunda kötü niyetli kişilerin sistemde yetkisiz SQL komutları çalıştırmasına olanak tanıyan ciddi bir güvenlik açığıdır.

2. Zafiyetli Senaryo Oluşturulması

AdventureWorks veritabanında HumanResources.Employee tablosu üzerinde, JobTitle'a göre filtreleme yapan zafiyetli bir stored procedure yazdım. Bu prosedürde kullanıcı girdisi doğrudan sorguya eklendiği için injection'a açık hale geliyor. Kodda string birleştirme yöntemiyle SQL komutu oluşturuluyor.

```
1 ✓ USE AdventureWorks;
2 GO
3
4 -- Eğer prosedür zaten varsa, önce sil (tekrar tekrar çalıştırabilmek için)
5 ✓ IF OBJECT_ID('dbo.usp_GetEmployeeByJobTitle_Vulnerable', 'P') IS NOT NULL
6     DROP PROCEDURE dbo.usp_GetEmployeeByJobTitle_Vulnerable;
7 GO
8
9 ✓ CREATE PROCEDURE dbo.usp_GetEmployeeByJobTitle_Vulnerable
10     @JobTitle NVARCHAR(50)
11 AS
12 BEGIN
13     DECLARE @SQL NVARCHAR(MAX);
14
15     -- Kullanıcı girdisi (@JobTitle) doğrudan sorguya ekleniyor - ZAFİYET BURADA
16     SET @SQL = N'SELECT BusinessEntityID, NationalIDNumber, LoginID, JobTitle
17         FROM HumanResources.Employee
18         WHERE JobTitle LIKE '%' + @JobTitle + '%''';
19
20     PRINT N'Çalıştırılacak Güvensiz Sorgu: ' + @SQL; -- Çalıştırılacak sorguyu görmek için
21
22     EXEC sp_executesql @SQL; -- Dinamik SQL'i çalıştır
23 END
24 GO
25 |
```

```
AdventureWorks> USE AdventureWorks
[2025-04-21 14:47:40] [S0001][5701] Changed database context to 'AdventureWorks'.
[2025-04-21 14:47:40] completed in 6 ms
AdventureWorks> IF OBJECT_ID('dbo.usp_GetEmployeeByJobTitle_Vulnerable', 'P') IS NOT NULL
    DROP PROCEDURE dbo.usp_GetEmployeeByJobTitle_Vulnerable;
[2025-04-21 14:47:40] completed in 6 ms
AdventureWorks> CREATE PROCEDURE dbo.usp_GetEmployeeByJobTitle_Vulnerable
    @JobTitle NVARCHAR(50)
AS
BEGIN
    DECLARE @SQL NVARCHAR(MAX);

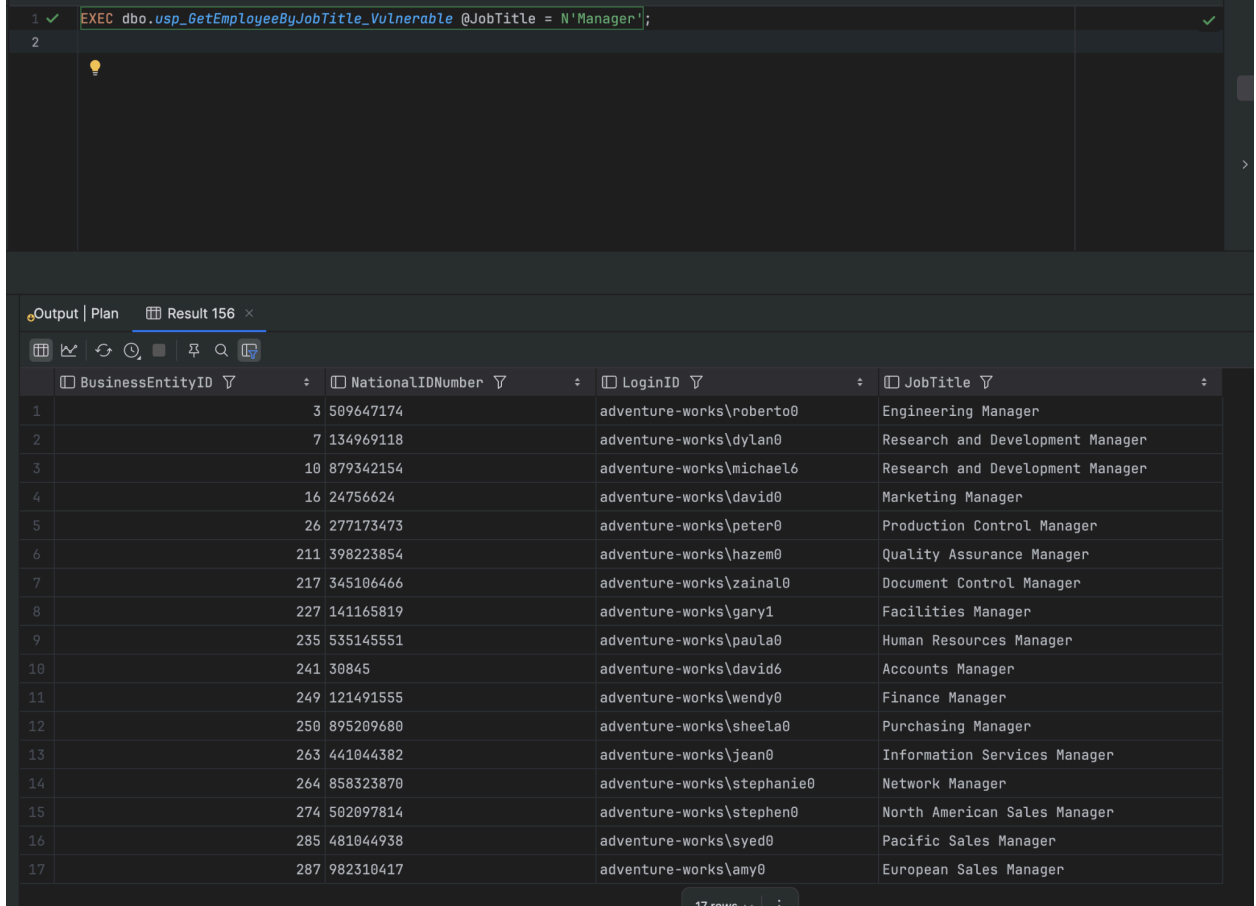
    -- Kullanıcı girdisi (@JobTitle) doğrudan sorguya ekleniyor - ZAFİYET BURADA
    SET @SQL = N'SELECT BusinessEntityID, NationalIDNumber, LoginID, JobTitle
        FROM HumanResources.Employee
        WHERE JobTitle LIKE '%' + @JobTitle + '%''';

    PRINT N'Çalıştırılacak Güvensiz Sorgu: ' + @SQL; -- Çalıştırılacak sorguyu görmek için

    EXEC sp_executesql @SQL; -- Dinamik SQL'i çalıştır
END
[2025-04-21 14:47:40] completed in 33 ms
```

3. Zafiyetli Senaryonun Test Edilmesi (Normal Kullanım)

İlk olarak prosedürü 'Manager' gibi geçerli bir girdiye göre çalıştırdım. Beklendiği gibi sadece bu unvana sahip çalışanlar listelendi. Aynı zamanda PRINT komutu ile oluşan SQL sorgusunu da gözlemladim.



EXEC dbo.usp_GetEmployeeByJobTitle_Vulnerable @JobTitle = N'Manager';

BusinessEntityID	NationalIDNumber	LoginID	JobTitle
3	509647174	adventure-works\roberto0	Engineering Manager
7	134969118	adventure-works\dylan0	Research and Development Manager
10	879342154	adventure-works\michael6	Research and Development Manager
16	24756624	adventure-works\david0	Marketing Manager
26	277173473	adventure-works\peter0	Production Control Manager
211	398223854	adventure-works\hazem0	Quality Assurance Manager
217	345106466	adventure-works\zainal0	Document Control Manager
227	141165819	adventure-works\gary1	Facilities Manager
235	535145551	adventure-works\paula0	Human Resources Manager
241	30845	adventure-works\david6	Accounts Manager
249	121491555	adventure-works\wendy0	Finance Manager
250	895209680	adventure-works\sheela0	Purchasing Manager
263	441044382	adventure-works\jean0	Information Services Manager
264	858323870	adventure-works\stephanie0	Network Manager
274	502097814	adventure-works\stephen0	North American Sales Manager
285	481044938	adventure-works\syed0	Pacific Sales Manager
287	982310417	adventure-works\amy0	European Sales Manager

4. SQL Injection Denemesi

Ardından, 'X' ' OR 1=1 -- şeklinde bilindik bir injection girdisi ile prosedürü tekrar çalıştırdım. Bu sefer WHERE koşulu geçersiz hale geldi ve tüm kayıtlar listelendi. PRINT çıktısındaki sorgu da bu yapıyı açıkça gösteriyordu. OR 1=1 ifadesi koşulu sürekli doğruya çevirdi, -- ise sorgunun kalanını yorum satırına aldı.

1	✓	EXEC dbo.usp_GetEmployeeByJobTitle_Vulnerable @JobTitle = N'X' OR 1=1 --';	✓
2			

Output Plan Result 157 x				
	BusinessEntityID	NationalIDNumber	LoginID	JobTitle
276	276	191644724	adventure-works\Linda3	Sales Representative
277	277	615389812	adventure-works\jillian0	Sales Representative
278	278	234474252	adventure-works\garrett1	Sales Representative
279	279	716374314	adventure-works\tsvi0	Sales Representative
280	280	61161660	adventure-works\pamela0	Sales Representative
281	281	139397894	adventure-works\shu0	Sales Representative
282	282	399771412	adventure-works\josé1	Sales Representative
283	283	987554265	adventure-works\David8	Sales Representative
284	284	90836195	adventure-works\Tete0	Sales Representative
285	285	481044938	adventure-works\syed0	Pacific Sales Manager
286	286	758596752	adventure-works\Lynn0	Sales Representative
287	287	982310417	adventure-works\amy0	European Sales Manager
288	288	954276278	adventure-works\rachel0	Sales Representative
289	289	668991357	adventure-works\jae0	Sales Representative
290	290	134219713	adventure-works\ranjit0	Sales Representative

5. Güvenli Yöntem: Parametrelili Sorgu

Aynı işlevi gören ancak injection'a karşı güvenli bir stored procedure yazdım. Bu versiyonda kullanıcı girdisi SQL metnine doğrudan dahil edilmek yerine, bir parametre olarak işlendi. Böylece SQL Server bu veriyi yalnızca veri olarak ele aldı, kod olarak değil.


```
1  ✓ USE AdventureWorks;
2  GO
3
4  ✓ IF OBJECT_ID('dbo.usp_GetEmployeeByJobTitle_Secure', 'P') IS NOT NULL
5      DROP PROCEDURE dbo.usp_GetEmployeeByJobTitle_Secure;
6  GO
7
8  ✓ CREATE PROCEDURE dbo.usp_GetEmployeeByJobTitle_Secure
9      @JobTitle NVARCHAR(50)
10 AS
11 BEGIN
12     -- Parametre doğrudan WHERE içinde kullanılıyor, SQL Injection riski yok!
13     -- SQL Server, @JobTitle içindeki özel karakterleri SQL kodu olarak yorumlamaz.
14     SELECT BusinessEntityID, NationalIDNumber, LoginID, JobTitle
15     FROM HumanResources.Employee
16     WHERE JobTitle LIKE '%' + @JobTitle + '%';
17 END
18 GO
19
```

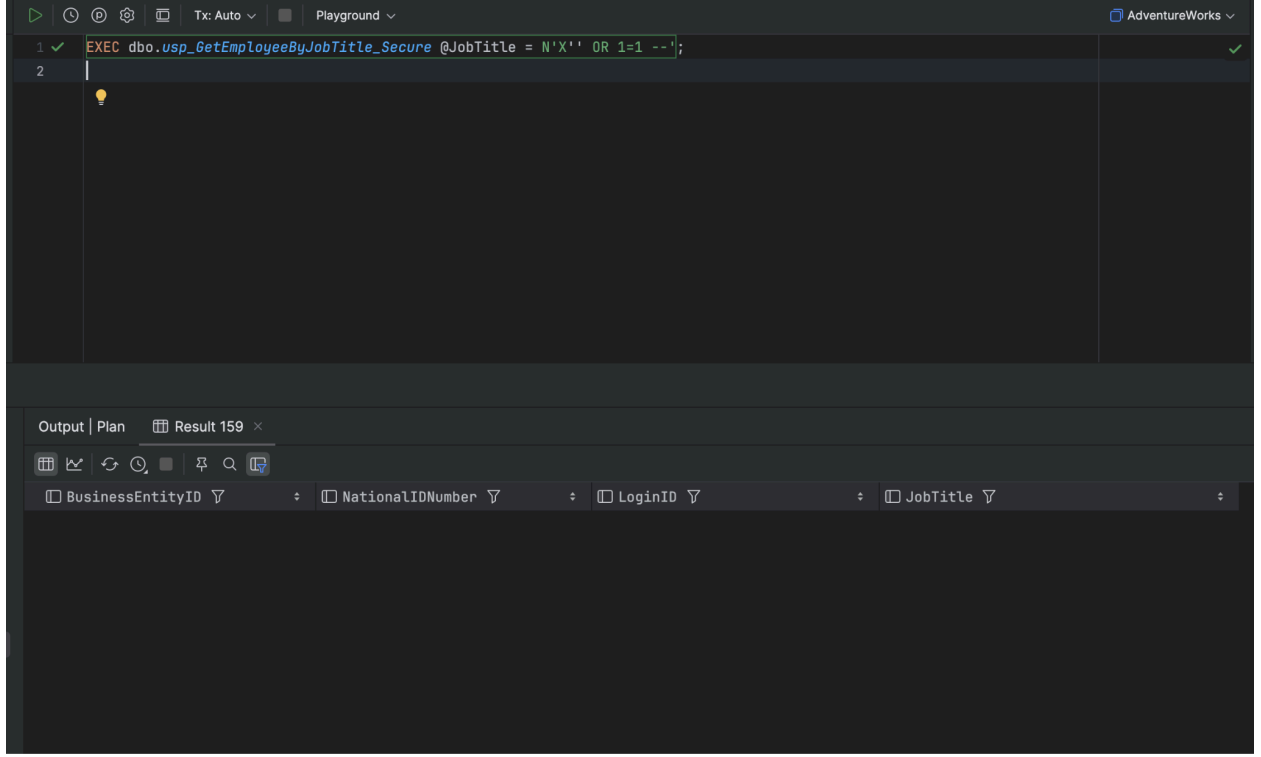
Output | Plan Result 157

Output Plan

```
      DROP PROCEDURE dbo.usp_GetEmployeeByJobTitle_Secure;
[2025-04-21 14:51:11] completed in 6 ms
AdventureWorks> CREATE PROCEDURE dbo.usp_GetEmployeeByJobTitle_Secure
      @JobTitle NVARCHAR(50)
      AS
      BEGIN
          -- Parametre doğrudan WHERE içinde kullanılıyor, SQL Injection riski yok!
          -- SQL Server, @JobTitle içindeki özel karakterleri SQL kodu olarak yorumlamaz.
          SELECT BusinessEntityID, NationalIDNumber, LoginID, JobTitle
          FROM HumanResources.Employee
          WHERE JobTitle LIKE '%' + @JobTitle + '%';
      END
[2025-04-21 14:51:11] completed in 30 ms
```

6. Güvenli Prosedürle Test

Bu güvenli prosedürü de yine aynı zararlı girdiyle test ettim. Sonuç olarak sistem bu girdiyi kod gibi değil, doğrudan arama kriteri olarak değerlendirdi ve doğal olarak eşleşen kayıt bulunamadı. Beklendiği gibi, veri sızıntısı yaşanmadı.



Bu bölümde, SQL Injection'ın ne kadar tehlikeli olabileceğini ve bu riskin nasıl önlenebileceğini net şekilde göstermiş oldum. Uygulama geliştirirken kullanıcıdan alınan her veriye kuşkuyla yaklaşmak gerekiyor. Parametrelili sorgular ya da benzer güvenlik önlemleri bu tür saldırılara karşı en temel savunma hattını oluşturuyor.