# Proje 6: Veritabanı Yükseltme ve Sürüm Yönetimi

## 1. Giriş

Veritabanı sistemleri, zamanla yeni özellikler, güvenlik güncellemeleri ve performans iyileştirmeleri içeren sürümlerle güncellenmektedir. Bu sürümler arasında geçiş yapmak, hem sistemin güncelliğini korumak hem de gelecekteki yazılım entegrasyonlarına uyumluluk sağlamak açısından kritik öneme sahiptir. Ancak veritabanı yükseltme işlemi, yalnızca teknik bir geçiş değil; aynı zamanda planlama, test, sürüm yönetimi ve olası geri dönüş stratejilerini içeren çok aşamalı bir süreçtir.

Bu proje kapsamında, bir veritabanının eski sürümden daha yeni bir SQL Server sürümüne yükseltilmesi işlemi gerçekleştirilmiş, bu süreç boyunca veri bütünlüğünün korunması, sistem davranışlarının test edilmesi ve yükseltme sonrası şema değişikliklerinin izlenmesi gibi adımlar detaylı bir şekilde uygulanmıştır. Özellikle **DDL (Data Definition Language) trigger** kullanılarak veritabanı yapısındaki değişikliklerin anlık olarak loglanması sağlanmıştır.

**1.1. Projenin Amacı**

Bu projenin temel amacı, bir veritabanının daha yeni bir SQL Server sürümüne geçişini planlamak, uygulamak ve bu süreçte meydana gelebilecek değişiklikleri denetim altına almaktır. Ayrıca, yükseltme sonrası sistemin düzgün çalışıp çalışmadığını doğrulamak ve gerekirse eski sürüme dönüş planı hazırlamak da projenin kapsamına dâhildir.

Alt hedefler aşağıdaki şekilde özetlenebilir:

* **Yedekleme ve geri yükleme stratejileri** ile veri kaybı riskini azaltmak
* **Sürüm farklarının tespiti ve etkilerinin analizi**
* **Şema (schema) değişikliklerinin izlenmesi** için DDL trigger yapılarını kullanmak
* **Yükseltme sonrası test sorguları ile sistem davranışını kontrol etmek**
* **Geri dönüş senaryosu hazırlayarak yükseltme işlemini güvenli hâle getirmek**

Bu hedeflerle birlikte, gerçek bir veritabanı yöneticisinin (DBA) izlemesi gereken profesyonel süreçler uygulamalı olarak deneyimlenmiştir.

**1.2. Kullanılan Veritabanı ve Ortam**

Bu projede Microsoft tarafından sağlanan örnek veritabanlarından biri olan **AdventureWorks2022** kullanılmıştır. Bu veritabanı; satış, müşteri, ürün, sipariş ve personel gibi çok sayıda ilişkilendirilmiş tablo içeren karmaşık bir yapıya sahiptir. Bu nedenle sürüm geçişleri ve şema değişikliklerinin izlenmesi açısından oldukça uygundur.

**Kullanılan Ortamlar:**

| **Bileşen** | **Bilgi** |
| --- | --- |
| Veritabanı | AdventureWorks2022 (orijinal olarak .bak dosyasından restore edilmiştir) |
| SQL Server Sürümü | SQL Server 2022 Express Edition |
| Yönetim Aracı | Microsoft SQL Server Management Studio (SSMS) |
| İşletim Sistemi | Windows 10/11 |

**Ek Araçlar:**

* **SQL Server Profiler** (sürüm karşılaştırmalarında kullanılabilir)
* **T-SQL Scriptleri** (DDL Trigger ve test işlemleri için)
* **Yedekleme dosyası (.bak)** kullanılarak yeni sürüme geçiş gerçekleştirilmiştir.

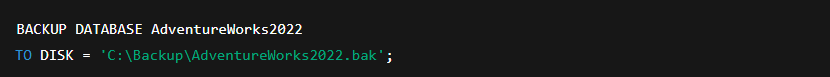
## 2. Veritabanı Yükseltme Planı

Veritabanı yükseltme işlemi; yalnızca teknik bir aktarma süreci değil, aynı zamanda riskleri minimize eden, veri bütünlüğünü koruyan ve performansı güvence altına alan stratejik bir süreçtir. Bu bölümde, AdventureWorks2022 veritabanının daha yeni bir SQL Server sürümüne yükseltilmesi için uygulanan adımlar detaylı olarak ele alınmıştır.

**2.1. Yedekleme Stratejisi**

Veritabanı yükseltme sürecinde ilk ve en kritik adım, **veritabanının eksiksiz bir yedeğinin alınmasıdır**. Olası bir hata, uyumsuzluk ya da veri kaybı durumunda sistemin önceki sürüme geri dönebilmesi için sağlam bir yedekleme planı gereklidir.

**Kullanılan T-SQL Komutu:**

Not: C:\Backup\ dizini mevcut değilse manuel olarak oluşturulmalı ya da SQL Server’ın yazma yetkisi olan bir dizin kullanılmalıdır.

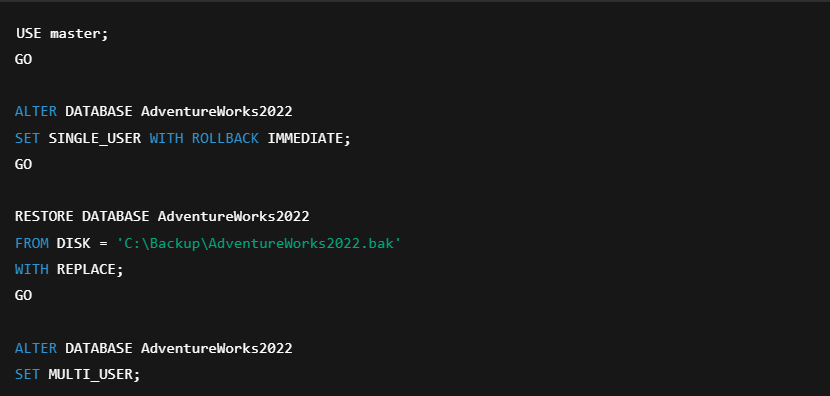
**Açıklama:**

* BACKUP DATABASE: Veritabanının tamamını .bak uzantılı bir yedek dosyasına aktarır.
* Bu yedek, hem yükseltme öncesi arşiv olarak saklanmış hem de test amaçlı geri yükleme için kullanılmıştır.

**2.2. Yeni Sürüm Ortamına Geri Yükleme (Restore)**

Yedek dosyası alındıktan sonra, bu dosya daha yeni bir SQL Server sürümüne **geri yüklenerek (restore edilerek)** yükseltme işlemi gerçekleştirilmiştir. Bu işlem sırasında, hedef veritabanının kullanılmadığından emin olunmalı ve WITH REPLACE komutu ile eski yapı üzerine yazılmalıdır.

**Kullanılan Komut Dizisi:**

 **Açıklama:**

* SET SINGLE\_USER WITH ROLLBACK IMMEDIATE: Restore işlemini engelleyen açık oturumları sonlandırır.
* WITH REPLACE: Var olan veritabanı üzerine yazılmasını sağlar.
* İşlem tamamlandıktan sonra veritabanı tekrar çok kullanıcıya açılır.

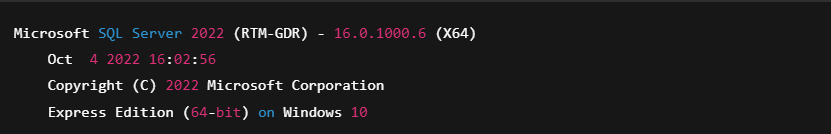
**2.3. Yükseltme Öncesi ve Sonrası Versiyon Kontrolü**

Yükseltme işleminin başarıyla tamamlandığını doğrulamak için, hem önceki hem de yeni ortamda SQL Server sürümü sorgulanmıştır.

**Kullanılan Komut:**

Bu komut, SQL Server'ın kurulu sürümünü ve yapı numarasını döndürür. Böylece geçilen ortamın gerçekten daha yeni bir sürüm olup olmadığı doğrulanabilir.

**Örnek Çıktı:**



## 3. Sürüm Yönetimi ve Şema Değişikliklerinin Takibi

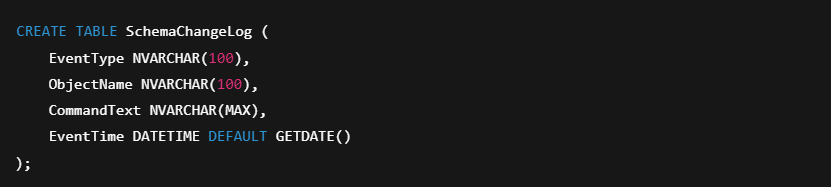
Veritabanı yapısındaki değişikliklerin takibi, özellikle çok kullanıcılı ortamlarda veri bütünlüğünü ve sistem kontrolünü sağlamak açısından büyük önem taşır. Bu tür değişiklikleri izlemek, versiyonlama ve sürüm yönetimi açısından kritik bir adımdır. SQL Server’da bu tür işlemleri otomatik olarak kaydetmenin etkili bir yolu **DDL (Data Definition Language) trigger** kullanmaktır.

Bu proje kapsamında AdventureWorks2022 veritabanı üzerinde yapılan şema değişiklikleri, özel olarak oluşturulmuş bir DDL trigger sayesinde bir log tablosunda kayıt altına alınmıştır.

**3.1. DDL Trigger ile Şema Değişikliği Günlüğü Oluşturma**

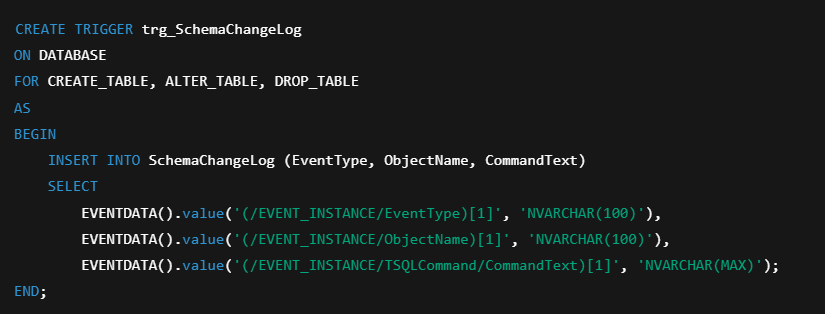
İlk adım olarak, şema değişikliklerini tutacak bir **log tablosu** ve ardından bu tabloya veri yazacak bir **DDL trigger** oluşturulmuştur. Bu yapı sayesinde veritabanında CREATE, ALTER, DROP gibi yapısal değişiklikler olduğunda bunlar otomatik olarak kaydedilir.

**Oluşturulan Log Tablosu:**

 **Açıklama:**

* EventType: Gerçekleşen işlem türü (örneğin ALTER\_TABLE)
* ObjectName: Değiştirilen tablonun/nesnenin adı
* CommandText: Gerçekleştirilen SQL komutunun kendisi
* EventTime: İşlem zamanı

**DDL Trigger Oluşturma:**

 **Açıklama:**

* ON DATABASE: Tetikleyicinin veritabanı genelinde geçerli olduğunu belirtir.
* EVENTDATA(): SQL Server'da gerçekleşen olay hakkında XML formatında veri döndürür.
* Bu tetikleyici sayesinde CREATE, ALTER ve DROP işlemleri otomatik olarak SchemaChangeLog tablosuna kaydedilir.

**3.2. Şema Değişikliklerinin Kaydedilmesi ve Görüntülenmesi**

DDL trigger’ının doğru çalıştığını test etmek amacıyla, veritabanında bir şema değişikliği işlemi yapılmış ve bunun SchemaChangeLog tablosuna yansıyıp yansımadığı kontrol edilmiştir.

**Test Amaçlı Şema Değişikliği:**

Bu komut, Person tablosuna yeni bir sütun ekler ve bir DDL olayı tetikler.

**Log Kayıtlarını Görüntüleme:**



**Beklenen Çıktı:**

| **EventType** | **ObjectName** | **CommandText** | **EventTime** |
| --- | --- | --- | --- |
| ALTER\_TABLE | Person | ALTER TABLE ... ADD MiddleInitial | 2025-05-29 23:45:12 |

## 4. Test ve Geri Dönüş Planı

Veritabanı yükseltme işleminin tamamlanmasının ardından, yeni ortamda sistemin kararlı şekilde çalıştığını doğrulamak gerekir. Bu doğrulama testleri sayesinde veri bütünlüğü korunur, işlevsellik kontrol edilir ve sistemsel hataların önüne geçilir. Ayrıca herhangi bir olumsuz durumda geri dönüş (rollback) yapılabilmesi için bir kurtarma planı önceden hazırlanmalıdır.

**4.1. Yükseltme Sonrası Test İşlemleri**

Veritabanı SQL Server 2022 sürümüne restore edildikten sonra, temel fonksiyonların çalıştığını test etmek için çeşitli sorgular ve yapısal kontroller uygulanmıştır.

**Test Edilen Adımlar:**

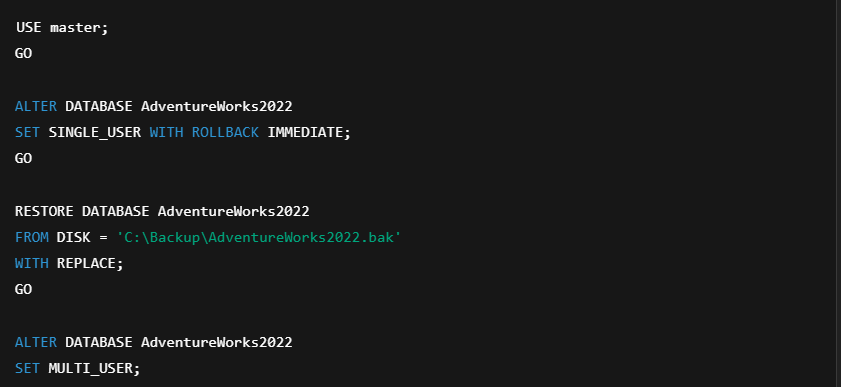
1. **Veri Bütünlüğü Testi**:
   * Örnek sorgular çalıştırılarak tabloların eksiksiz şekilde taşınıp taşınmadığı test edilmiştir.
2. **Yapısal Uyum Testi**:
   * Trigger, View, Stored Procedure gibi nesnelerin düzgün çalıştığı kontrol edilmiştir.
3. **Sürüm Uyumluluğu Testi**:
   * @@VERSION komutu ile SQL Server sürümünün güncel olup olmadığı doğrulanmıştır.
4. **DDL Trigger Doğrulaması**:
   * Şema üzerinde bir değişiklik yapılarak trigger'ın loglama işlevi başarıyla gerçekleştirdiği test edilmiştir.

**4.2. Geri Dönüş (Rollback) Senaryosu**

Yükseltme işlemi her zaman sorunsuz sonuçlanmayabilir. Uyumsuzluk, veri kaybı, uygulama tarafında hatalar veya kullanıcıdan gelen olumsuz geri bildirimler gibi durumlar için **önceden bir geri dönüş planı oluşturulmalıdır**.

Bu projede, yükseltme öncesi alınan .bak yedeği sayesinde istenildiğinde eski sürüme dönülebilecek yapı oluşturulmuştur.

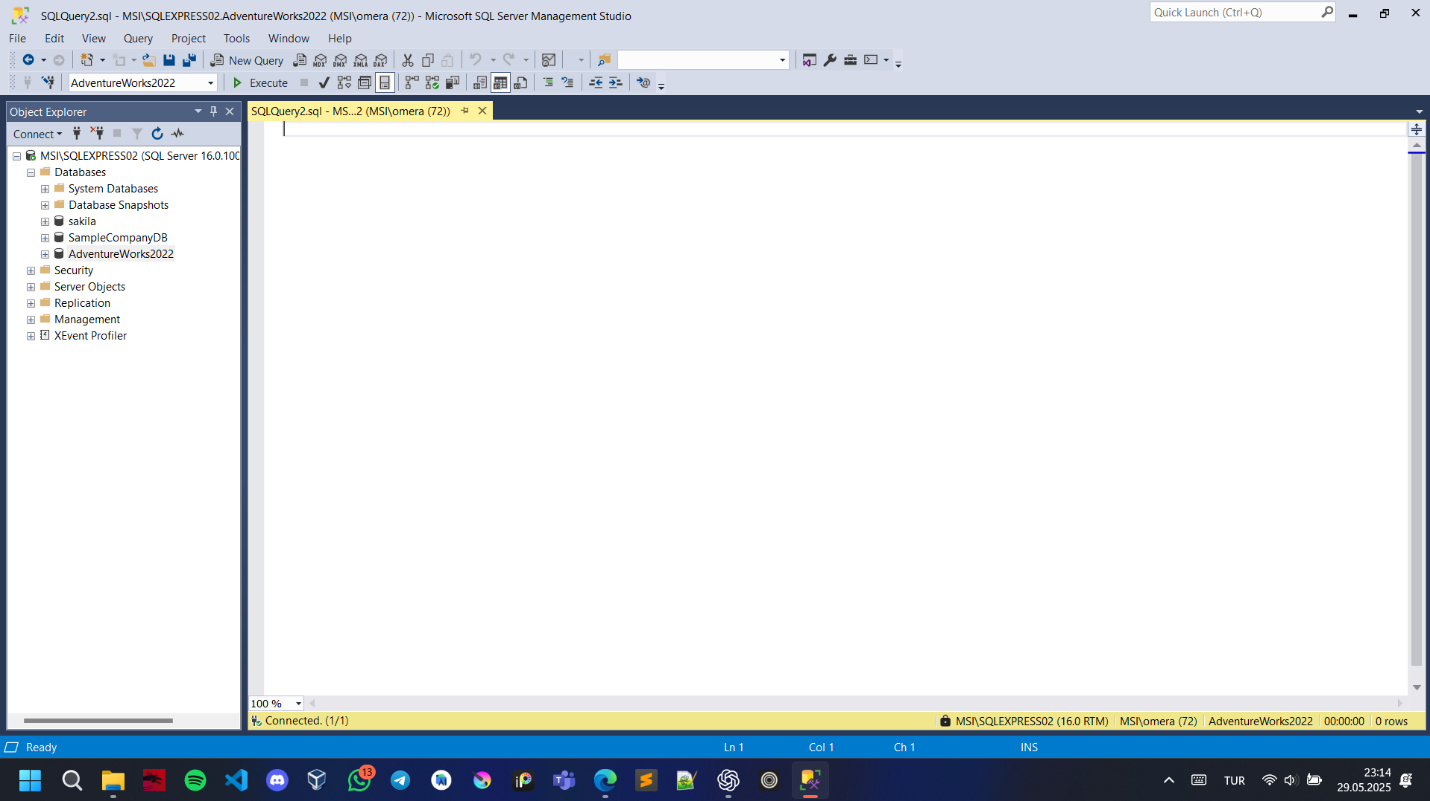
**Geri Dönüş Komutu:**

 **Açıklama:**

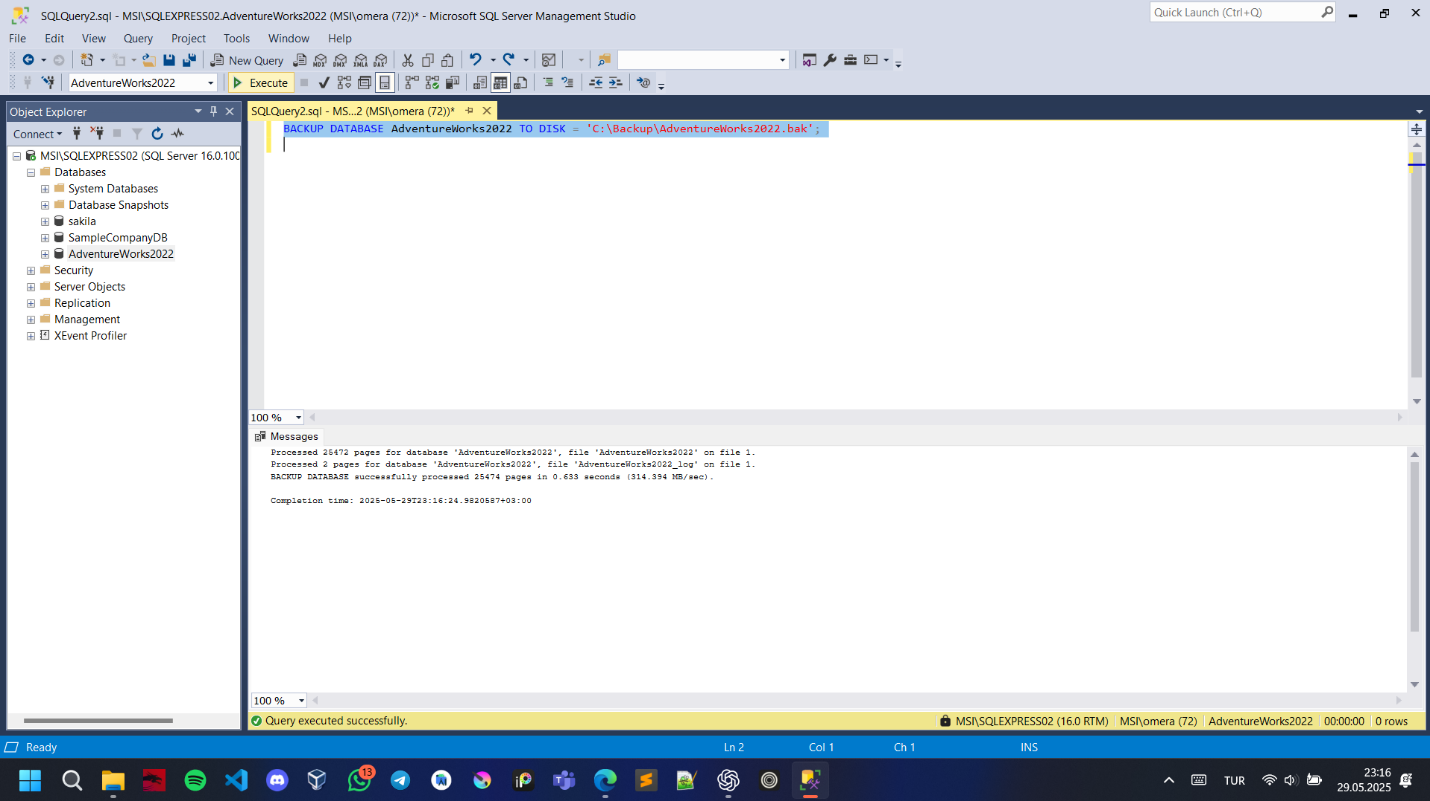
* SET SINGLE\_USER: Geri yükleme sırasında bağlantı hatalarının önüne geçer.
* WITH REPLACE: Var olan veritabanını üzerine yazar.
* Bu işlemler sonucunda sistem, yükseltme öncesindeki stabil sürüme döndürülmüş olur.

## 5.Ekran Görüntüleri

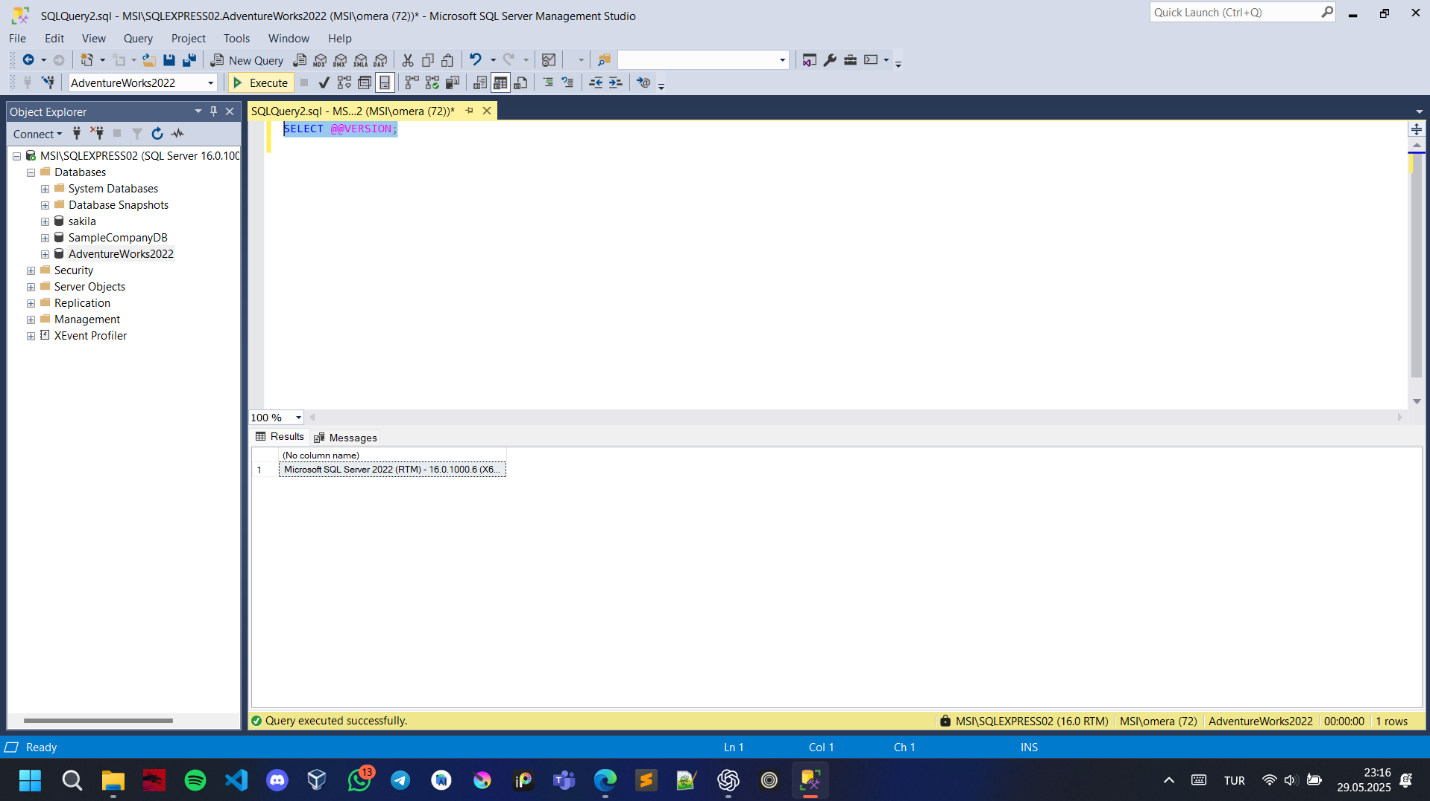
1)



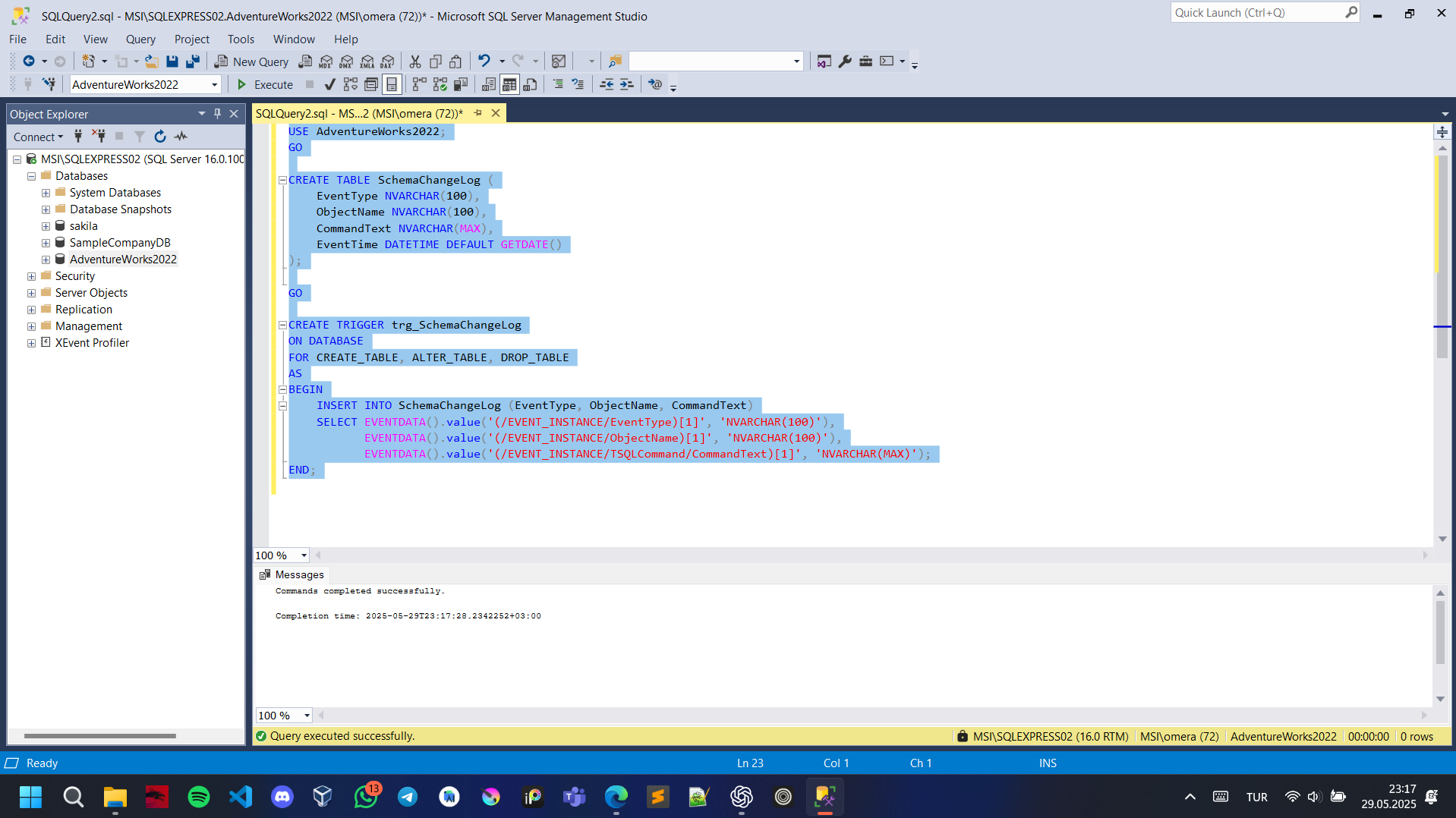
2)



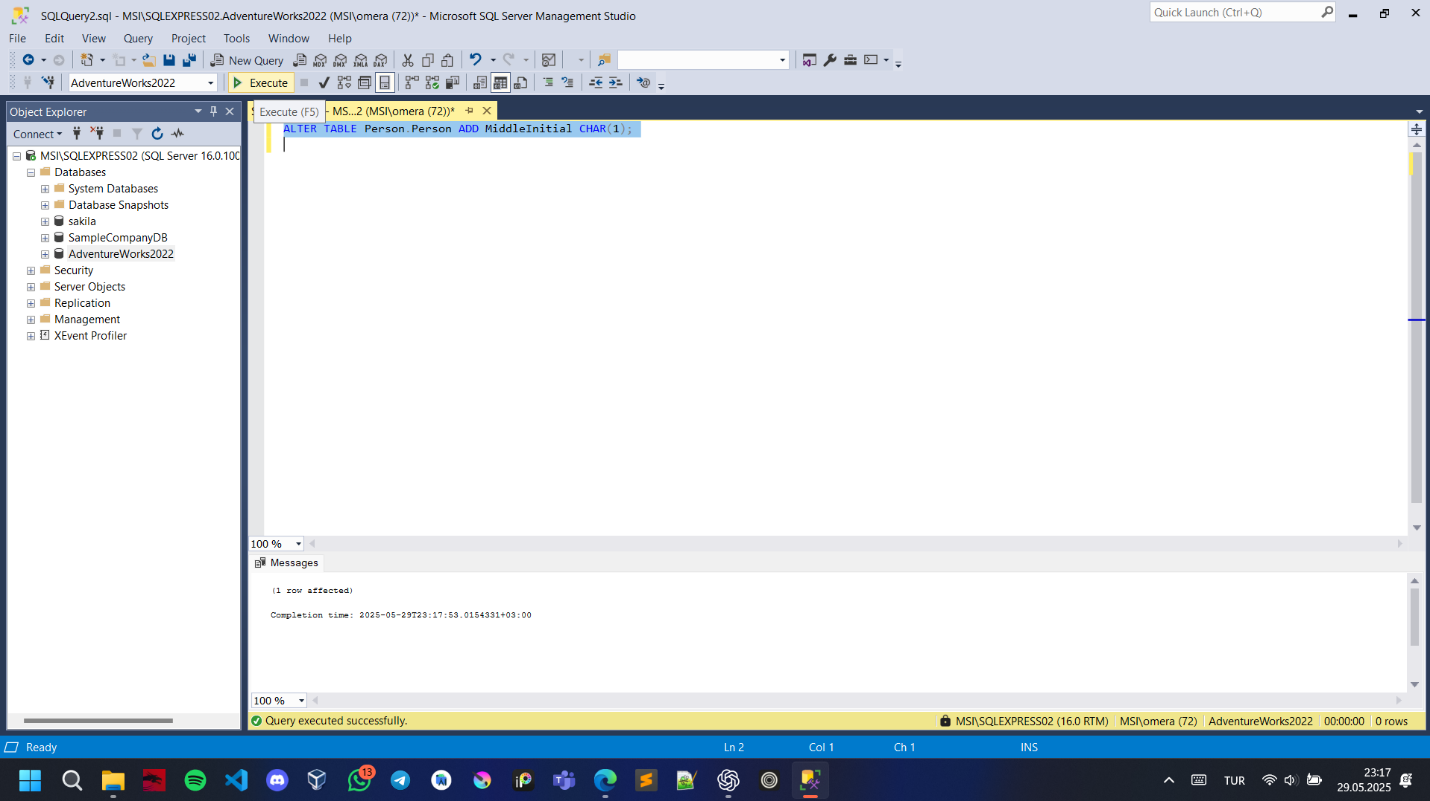
3)



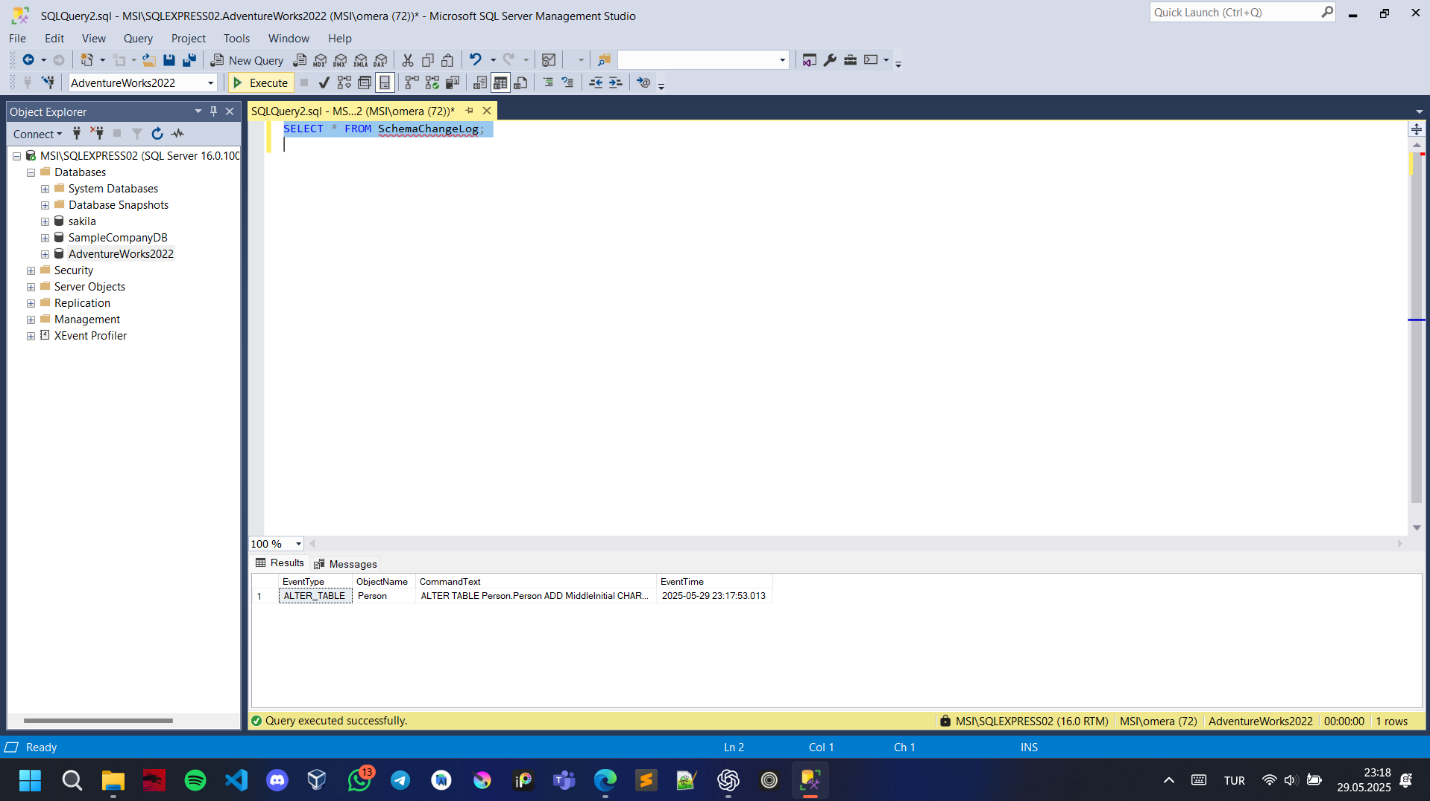
4)



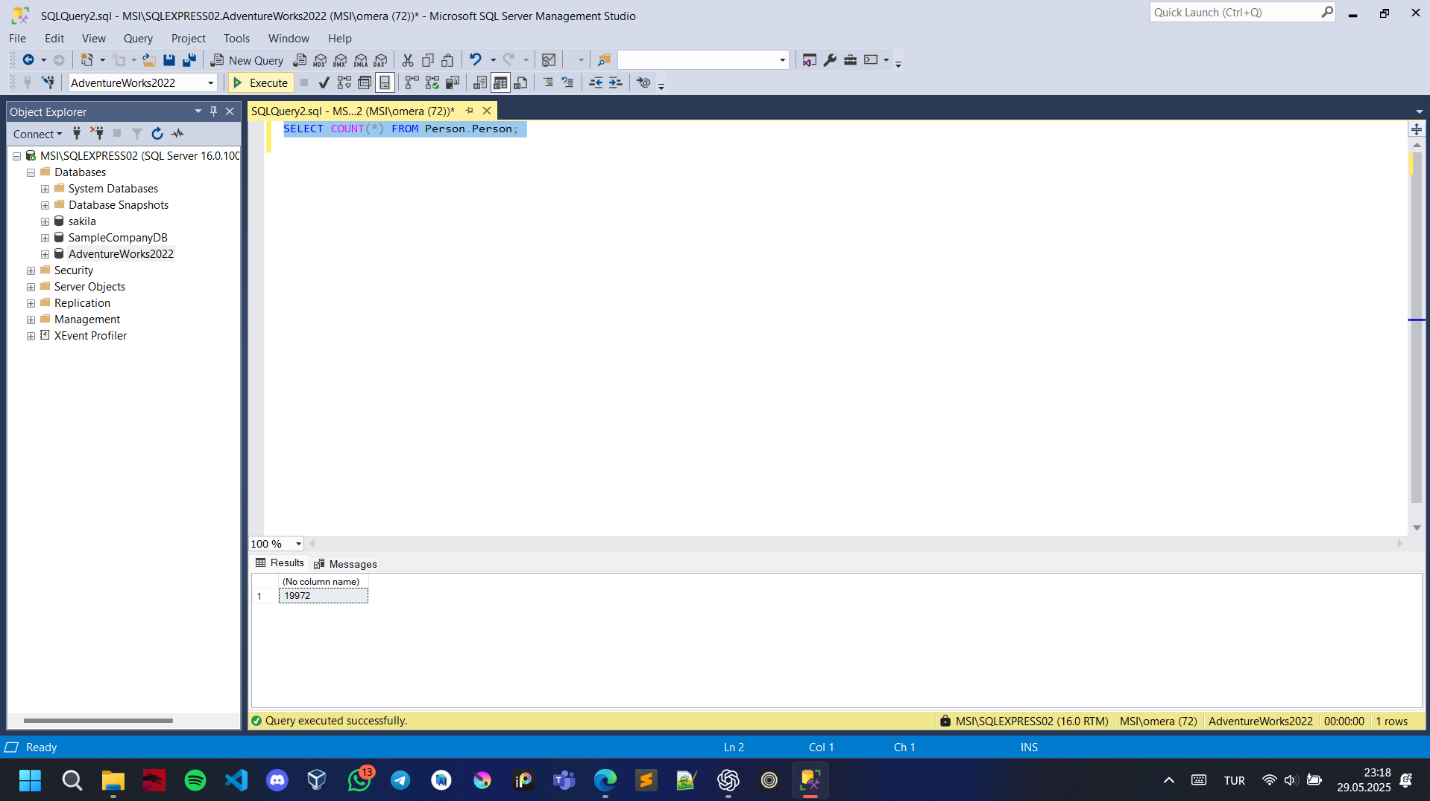
5)



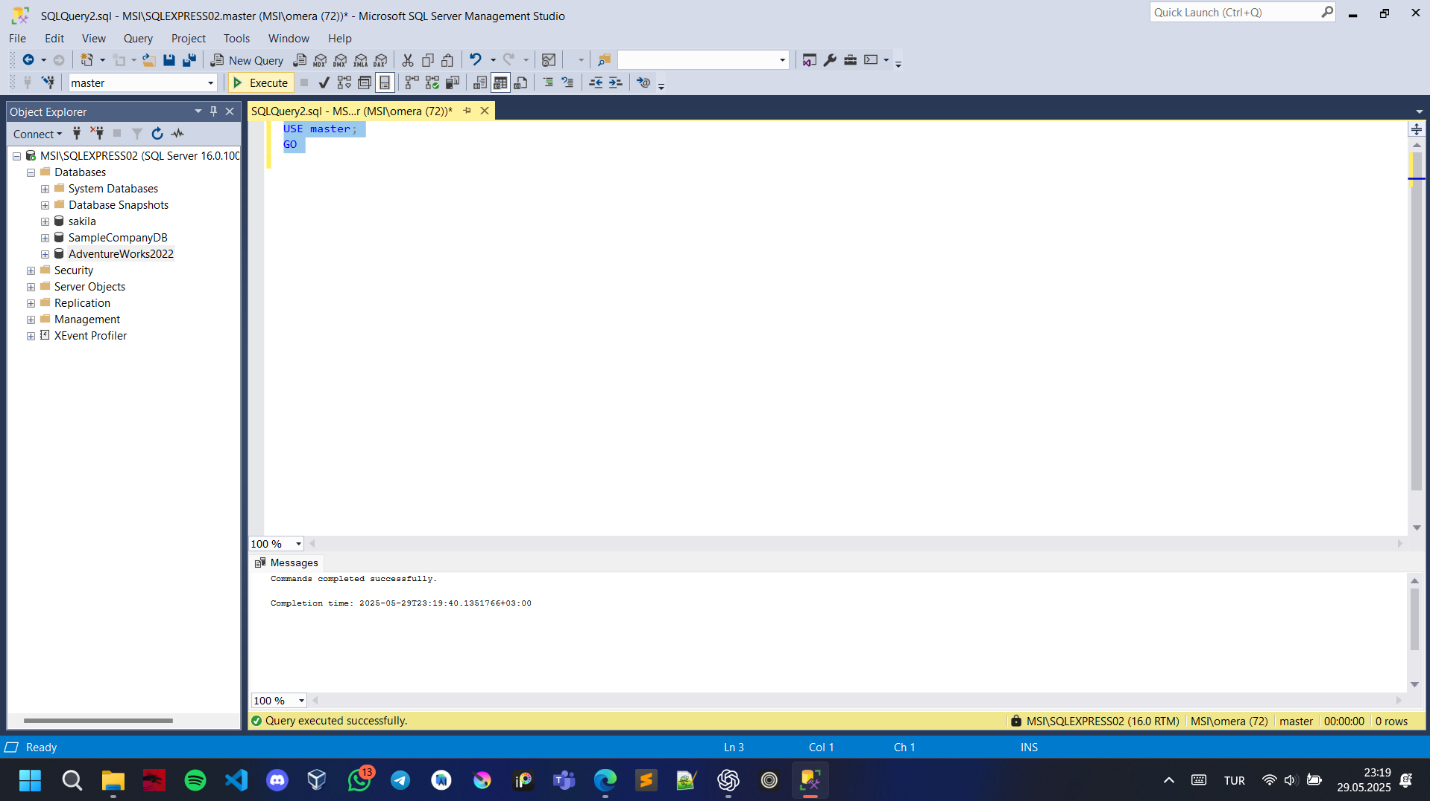
6)



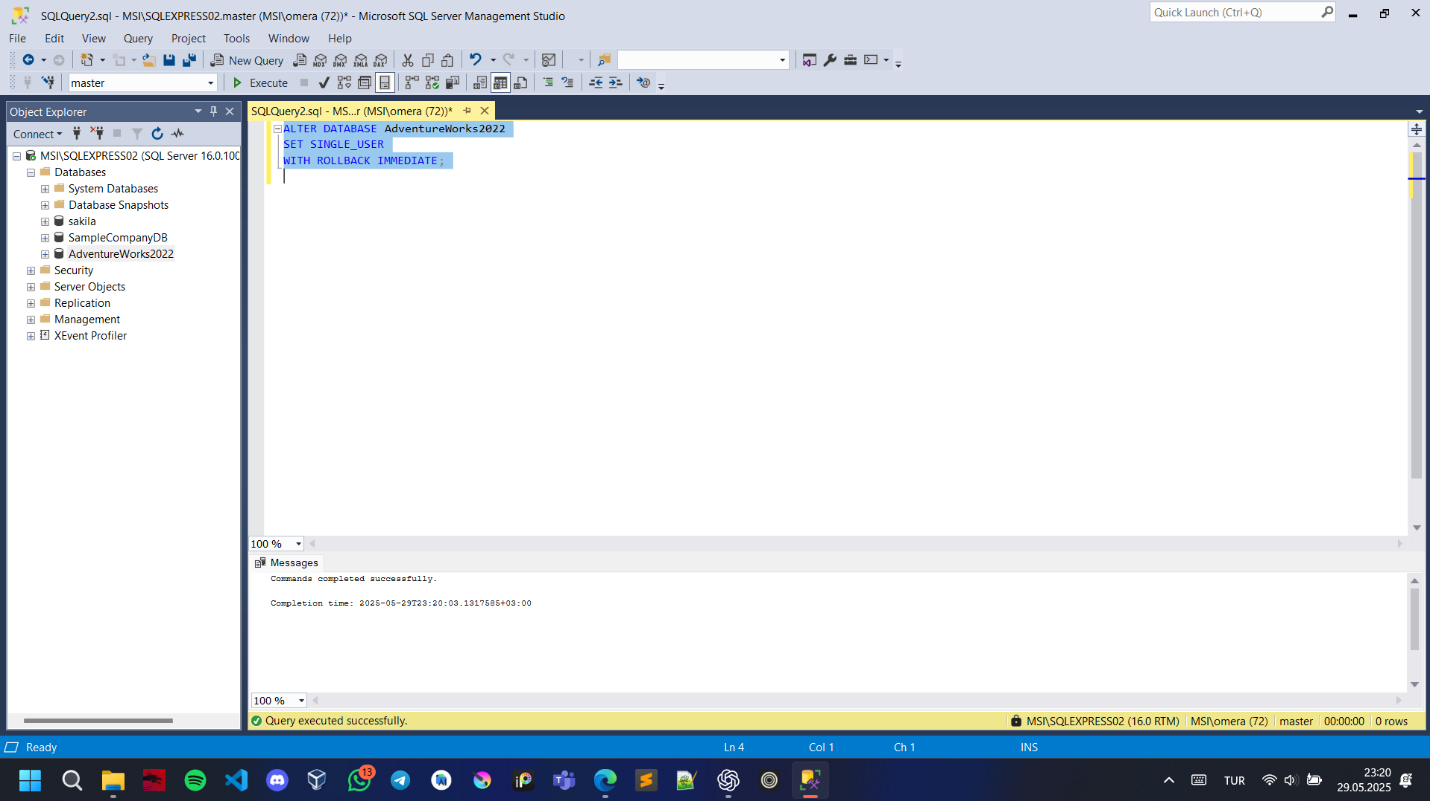
7)



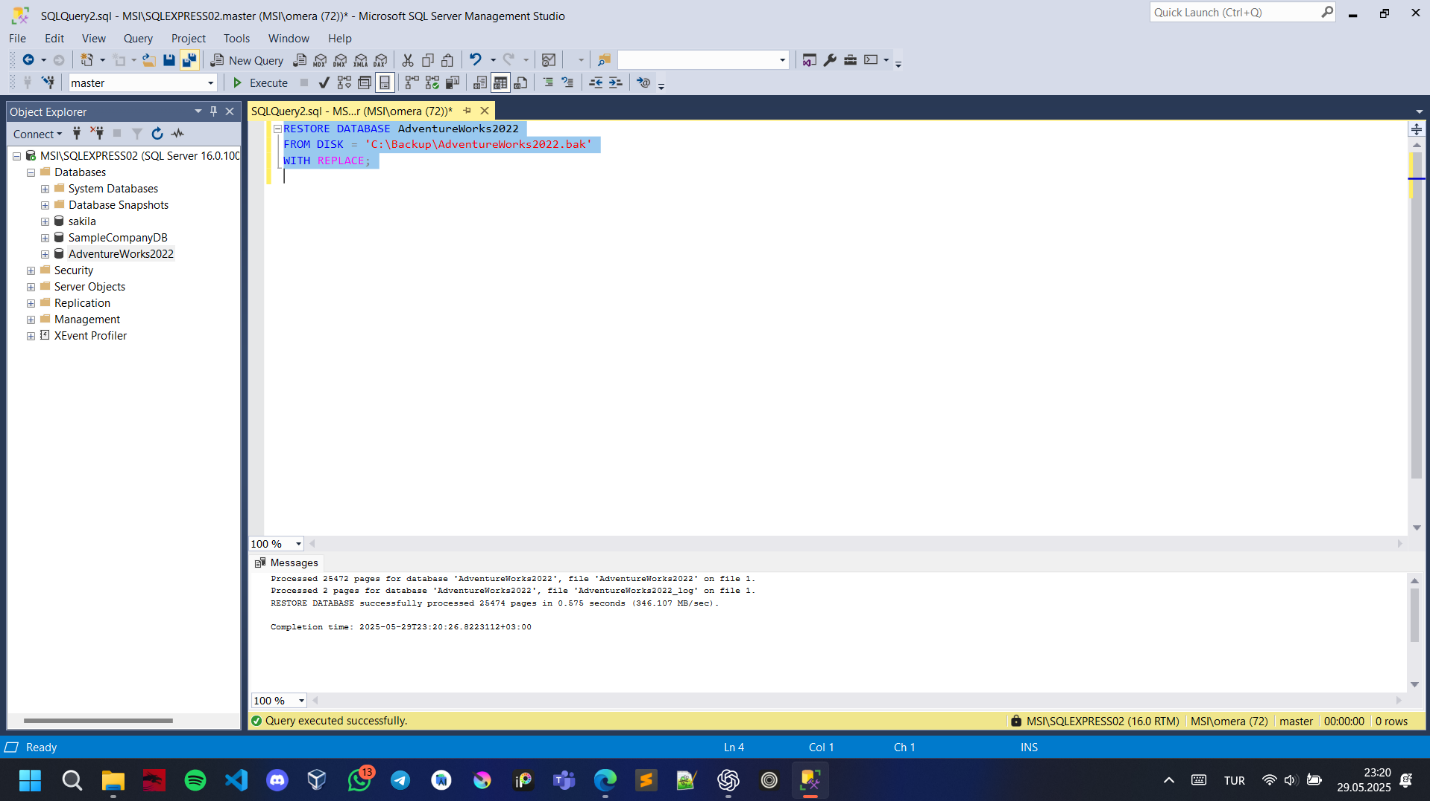
8)



9)



10)



11)

