**Proje 1: Veritabanı Performans Optimizasyonu ve İzleme**

**Kullanılan veri tabanı:** Microsoft AdventureWorks Data WareHouse 2022

**Link:** <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms#download-backup-files>

Github linki: <https://github.com/virtuososlove> <https://github.com/persuren>

Bu projede istenen adımlar 4 temel başlık altında değerlendirilmiş. 1- Veritabanı İzleme 2- İndeks Yönetimi 3- Sorgu İyileştirme: 4- Veri Yöneticisi Rolleri: Farklı roller için erişim yönetimi.

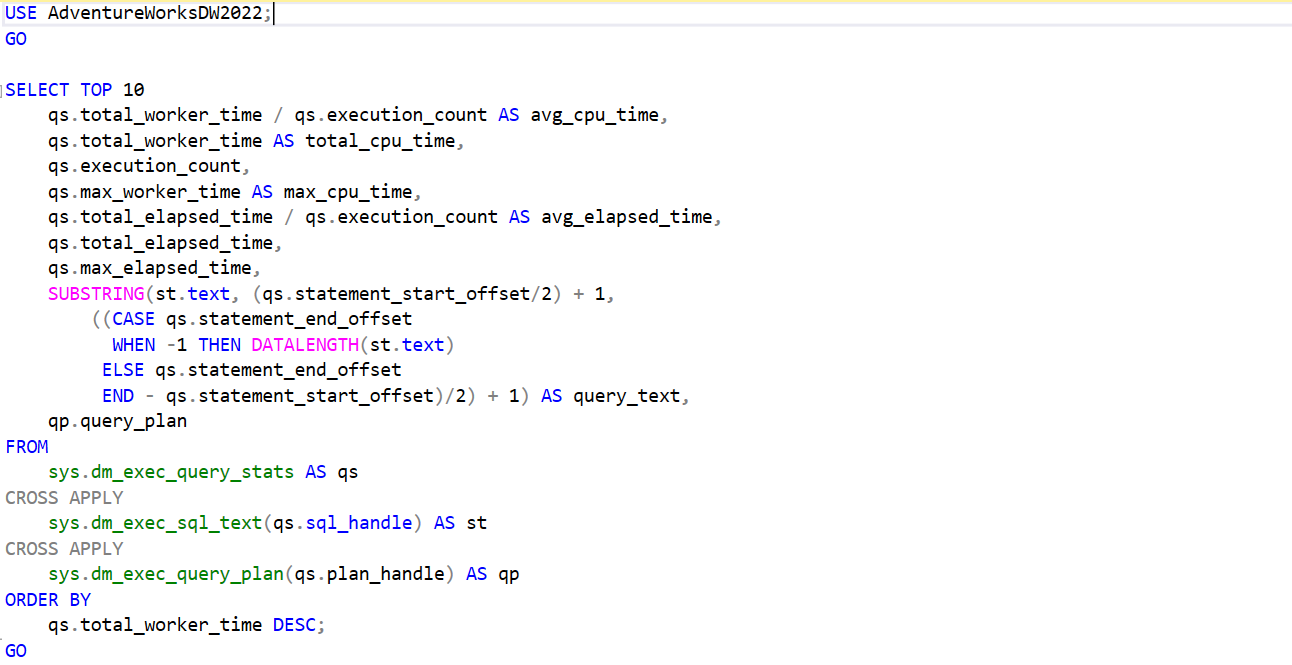
Biz de yaptığımız adımları bu başlıklar üzerinde değerlendirecek olursak.

**Veritabanı İzleme**

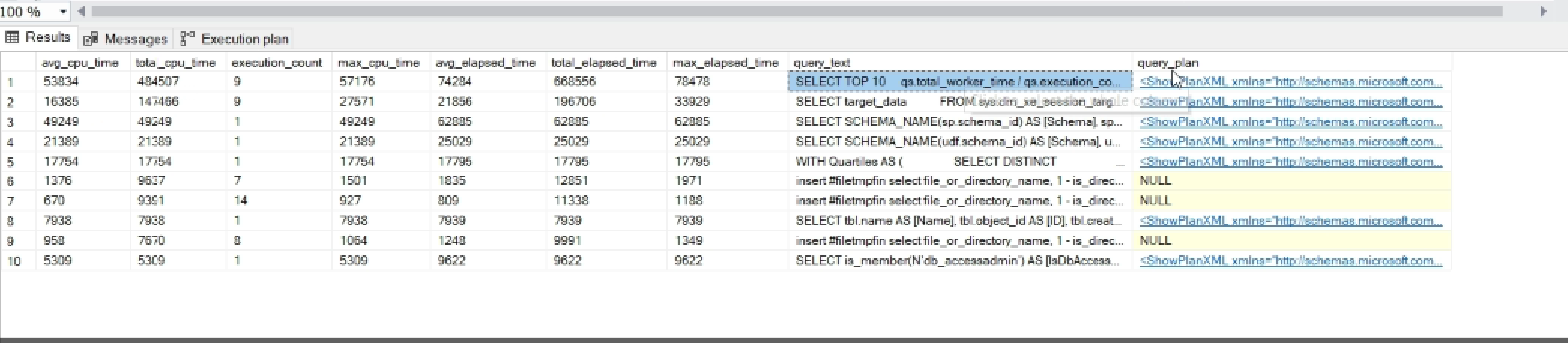
Bu başlık genel olarak birçok sorgu içerisinde Dynamic Management Views (DMV) kullanılarak performansın ölçülmesi, izlenmesi ve hataların tespit edilmesi için gerçekleştirilmiştir.

**Sorgu İyileştirme ve Performans Analizleri**

İlk olarak, veritabanında en fazla CPU zamanı harcayan sorguları tespit ederek başlamak mantıklı olacaktır. Bu, optimizasyona nereden başlayacağımızı belirlememize yardımcı olacak.



Bu sorgu, sys.dm\_exec\_query\_stats DMV'sini kullanarak sorgu istatistiklerini çeker ve en yüksek toplam CPU süresine sahip ilk 10 sorguyu listelenmesini sağlar.

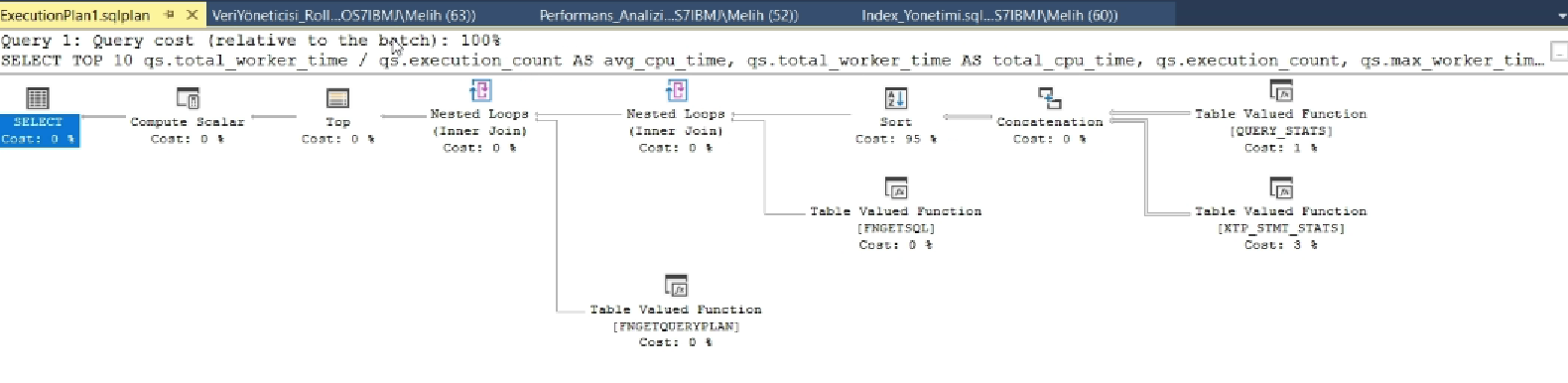


Mevcut sorguyu çalıştırdığımızda yukarıdaki gibi bir sonuç elde ediyoruz. İlk sıradaki sorgu, mevcut değerlere bakıldığında diğerlerine göre belirgin şekilde daha fazla kaynak (CPU ve süre) harcadığı anlaşılıyor.

Sorgu tipini ve query\_plan başlığı altındaki Execution Planı incelediğimizde performansı hakkında bir takım bilgiler elde edebiliyoruz. Execution Plandaki en yüksek maliyetli (%50) işlem "Clustered Index Scan". Bu genellikle, sorgunun ilgili tabloyu büyük ölçüde taraması gerektiği anlamına gelir çünkü WHERE koşuluna veya JOIN'lere uyan daha verimli bir indeks bulunamamıştır.

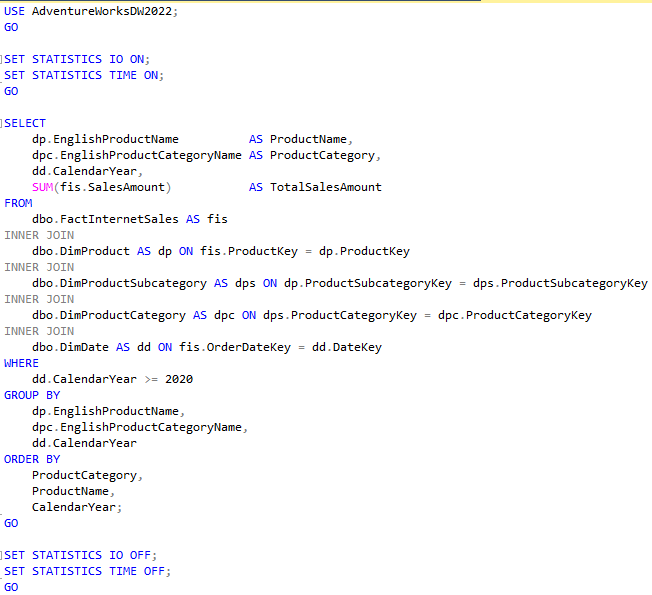
Query\_text kısmındaki sorgu tipine bakıldığında ise, SQL Server Management Studio'nun (SSMS) veritabanındaki fonksiyonlar gibi nesneleri listelemek için kullandığı bir metadata sorgusu olduğu görülüyor. sys.all\_objects, sys.sql\_modules gibi sistem tablolarını sorgulaya yarıyor.

Dolayısıyla bu tarz yapıdaki sistem sorgularını optimize etmek pek mümkün bir işlem değildir.

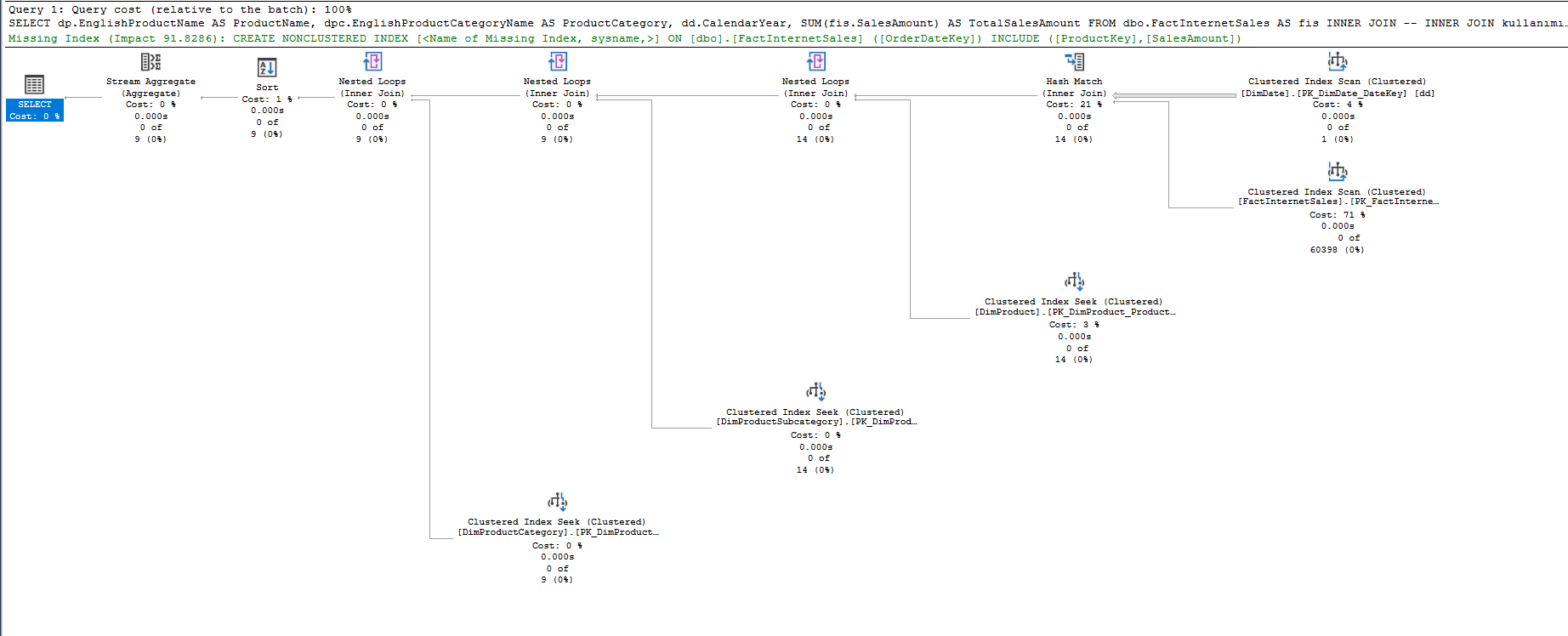


Bahsi geçen Execution Plan

Performans analizi için farklı işlemler deneyelim. FactInternetSales tablosunu ve ilişkili boyut tablolarını kullanan tipik bir analitik sorgu çalıştırarak performans ölçümü sağlamayı deneyebiliriz.



Bu sorgu çalıştırıldıktan sonra Messages kısmındaki sonuçlar ve Execution Plan incelenerek performans hakkında çeşitli bilgiler elde edilebilir.



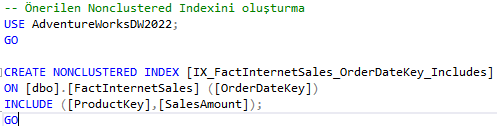
Sorgunun çalıştırılmasıyla elde edilen Execution Plan

Messages kısmı incelendiğinde logical reads değerinin 1013 olduğu görülüyor. Bu, tablonun sorgu için bir kez tarandığını ve 1013 veri sayfasının okunduğunu gösteriyor. Bu, genellikle tablonun tamamının veya büyük bir kısmının okunduğu anlamına gelir ve genellikle performans için istenmeyen bir durumdur.

Execution Plan incelendiğinde, plandaki en yüksek maliyetli operatör, FactInternetSales tablosu üzerindeki Clustered Index Scan. Mevcut bir indeks olan PK\_FactInternetSales sorgunun ihtiyaçlarını karşılamak için tamamen taranmak zorunda kalmış.

Planın en üstünde yeşil renkle yazan mesaj: Missing Index (Impact 80.8396): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[FactInternetSales] ([OrderDateKey]) INCLUDE ([ProductKey],[SalesAmount]) SQL Server Query Optimizer

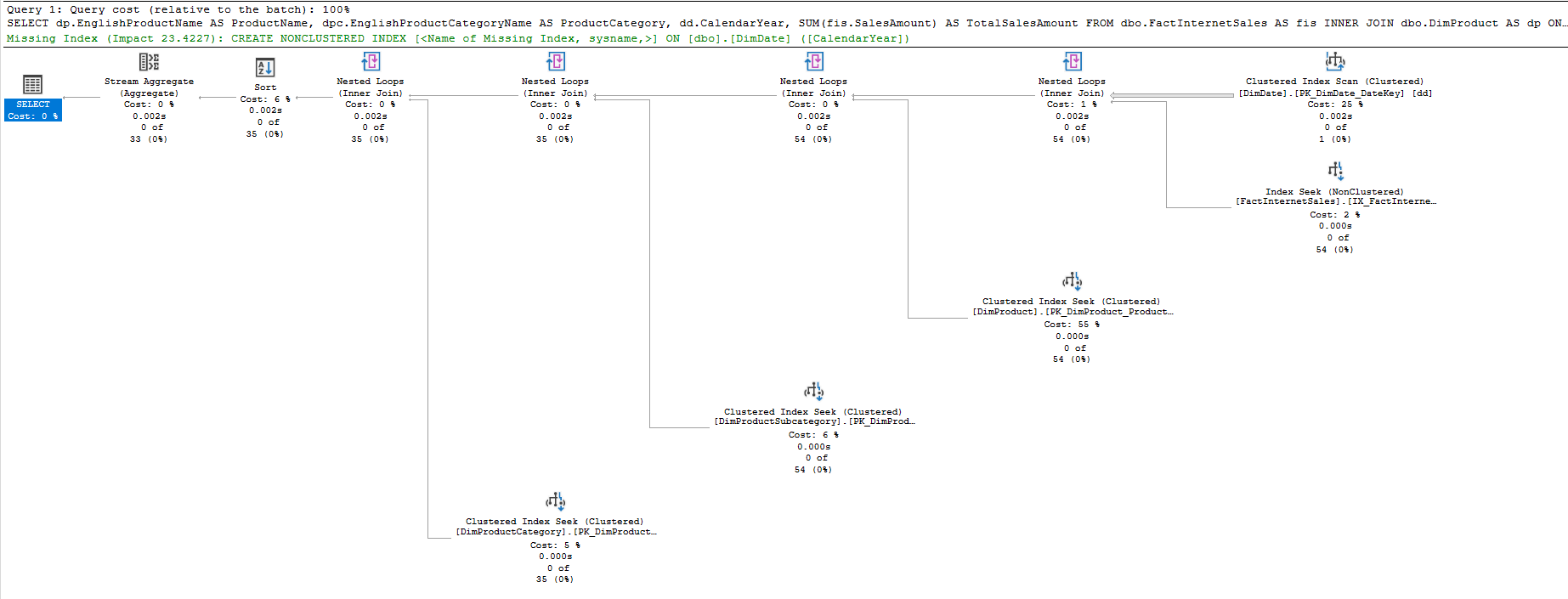
Bu sorgunun performansını %80.8 oranında iyileştirebileceğini düşündüğü bir indeks öneriyor. Öneri, FactInternetSales tablosunda OrderDateKey sütunu üzerine bir NONCLUSTERED indeks oluşturulması ve bu indekse ProductKey ile SalesAmount sütunlarının da dahil edilmesi yönünde.



Bu sorguyu çalıştırarak bize öneri olarak verilmiş indexi oluşturmuş oluyoruz. Önceki sorguyu tekrar çalıştırarak yine Execution Plan ve Messages kısmını inceleyerek performans artışını doğrulayabiliyoruz.

Önceki ve Sonraki Performansların Karşılaştırılması

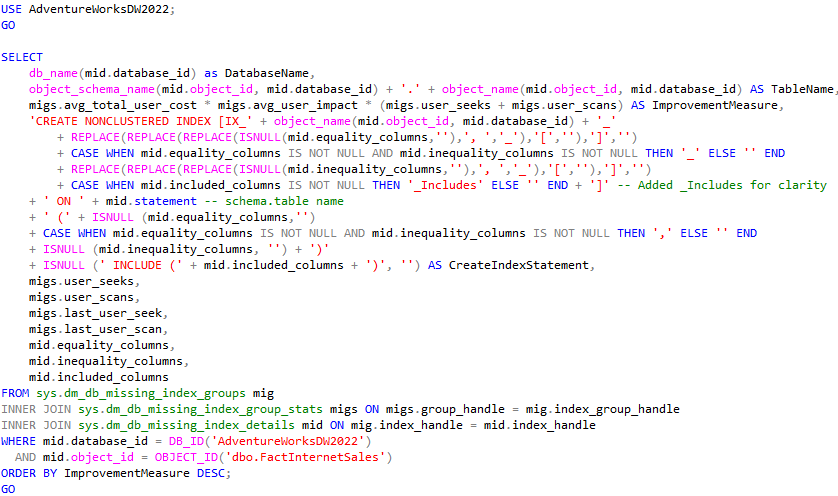
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metrik** | **İndeks Öncesi** | **İndeks Sonrası** | **Değişim** | **Sonuç** |
| **Zaman:** |  |  |  |  |
| CPU Süresi | 0 ms | 16 ms | +16 ms | (Önemsiz Artış) |
| Geçen Süre | 20 ms | 19 ms | -1 ms | (Benzer) |
| **Mantıksal Okumalar (IO):** |  |  |  |  |
| FactInternetSales | 1013 sayfa | 301 sayfa | -712 sayfa (%70.3 Azalma) | İYİLEŞME |
| DimProduct | 188 sayfa | 2 sayfa | -186 sayfa (%98.9 Azalma) | İYİLEŞME |
| Worktable (tempdb) | 1942 sayfa | 0 sayfa | -1942 sayfa (%100 Azalma) | İYİLEŞME |
| Toplam Mantıksal Okumalar | ~3236 sayfa | ~396 sayfa | ~ -2840 sayfa (%87.8 Azalma) | İYİLEŞME |
| **Execution Plan:** |  |  |  |  |
| FactInternetSales Erişimi | Clustered Index Scan (%72) | NonClustered Index Seek (%35) | Scan -> Seek | İYİLEŞME |
| En Maliyetli Operatör | Clustered Index Scan (%72) | Clustered Index Seek (DimDate) (%35) | Maliyet Azaldı ve Kaydı | İYİLEŞME |
| Eksik İndeks Önerisi | Var (Etki ~%80) | Yok | Öneri Giderildi | İYİLEŞME |
| Worktable Kullanımı | Var (Hash Join/Agg) | Yok (Stream Agg) | Kullanım Kalktı | İYİLEŞME |

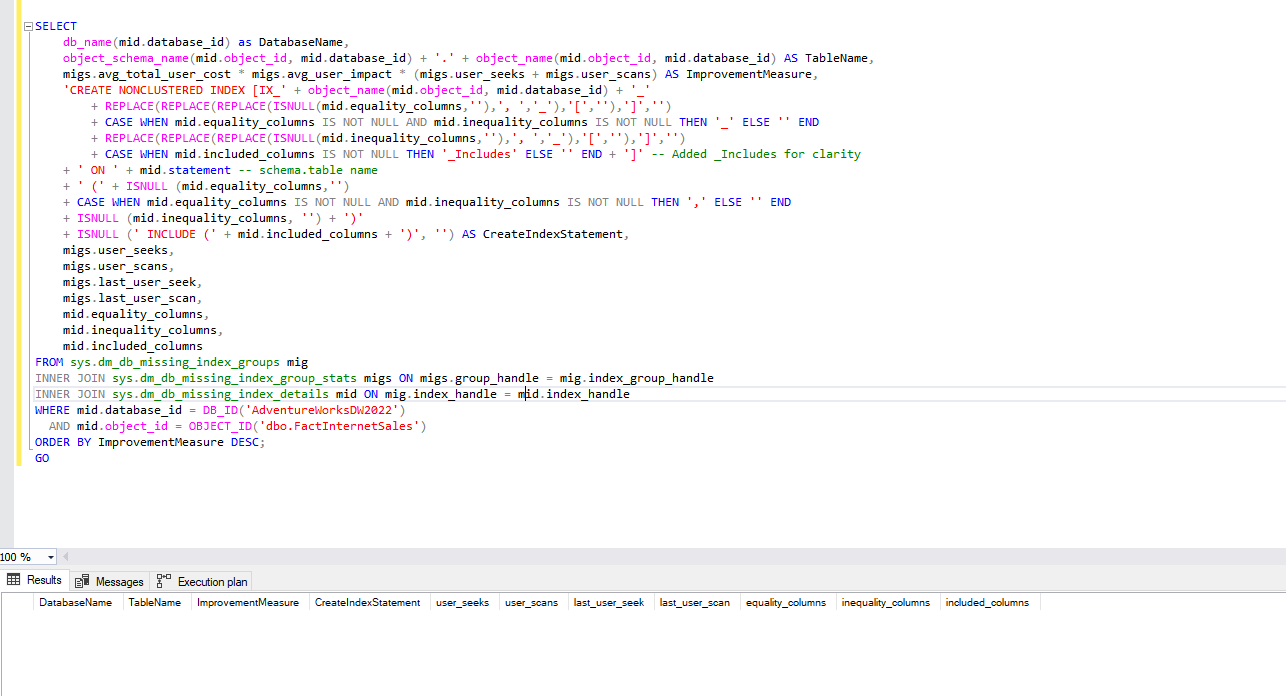


İndeks eklenmesi sonrası Execution Plan

**İndeks Yönetimi**

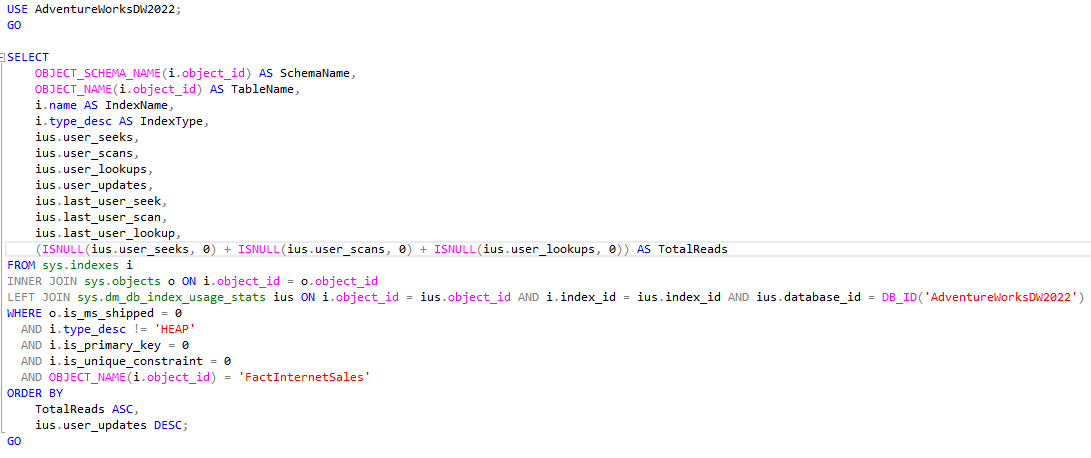
Proje hedeflerimizden biri olan İndeks Yönetimi kapsamında, veritabanındaki önemli ve büyük tablolardan biri olan FactInternetSales için SQL Server'ın önerdiği eksik indeksler olup olmadığını kontrol edelim. Aşağıdaki sorgu çalıştırılarak eksik indeks olup olmadığı kontrol edilebilir.



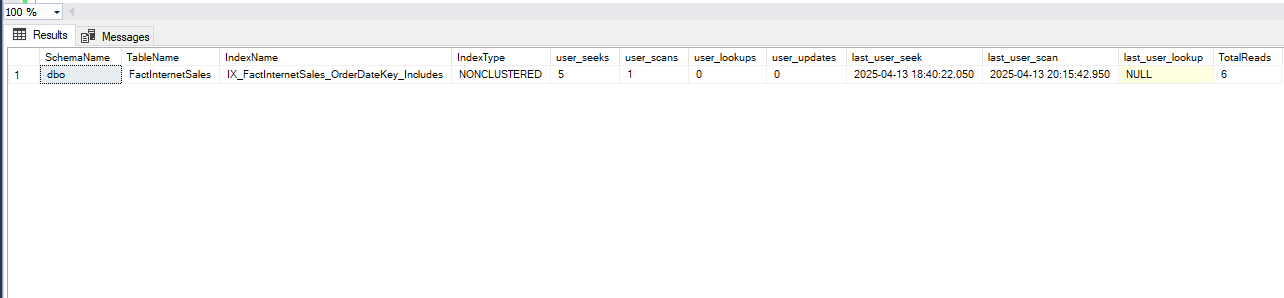


Sorgu çalıştırıldığında boş bir dönüş alındığı görülüyor. Sorgunun herhangi bir sonuç döndürmemesi, SQL Server'ın FactInternetSales tablosu için şu anda eksik olarak değerlendirdiği ve önemli performans artışı sağlayacağına inandığı bir indeks önerisi olmadığı anlamına geliyor.

Kullanılmayan indeksleri belirlemek için aşağıdaki sorguyu çalıştırabiliriz.



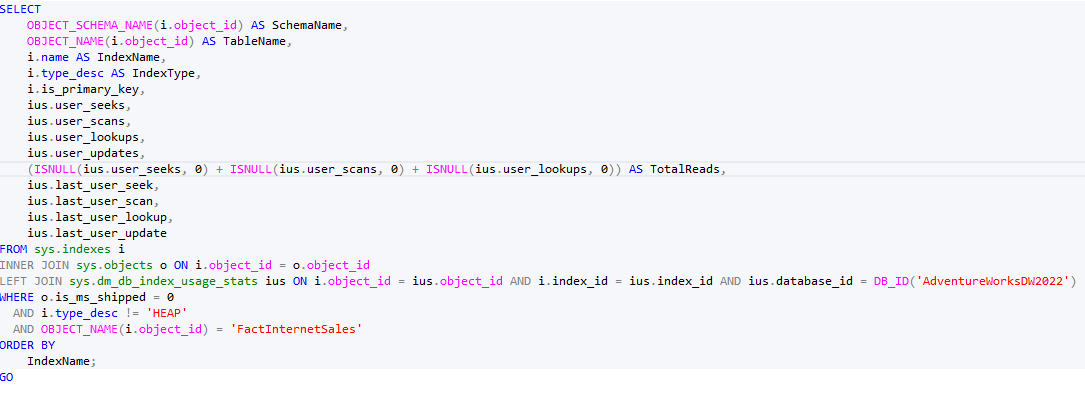
Bu sorguyu çalıştırdığımızda primary key olmayan sadece bir indeks olduğunu görüyoruz.



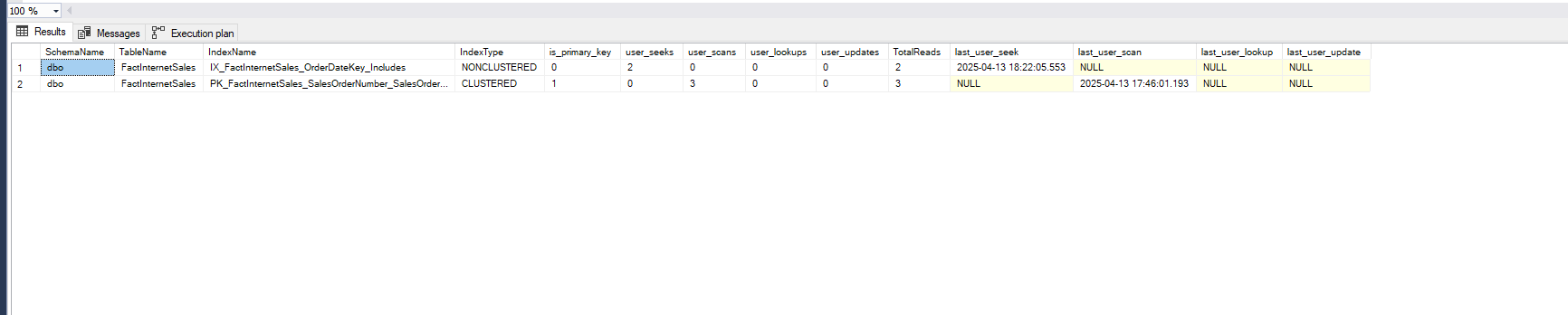
Sorgu sonucunda FactInternetSales sistemde kullanım primary olmayan indeksin sistem kaydının olduğunu doğruladı. Dolayısıyla, kullanım eksikliğine dayanarak kaldırılacak bir indeks bulunmuyor.

Son olarak tüm indekslerin kullanım istatistiklerini görüntülemek mantıklı olabilir. Bu sayede kullanılmayan indeksleri silerebiliriz.

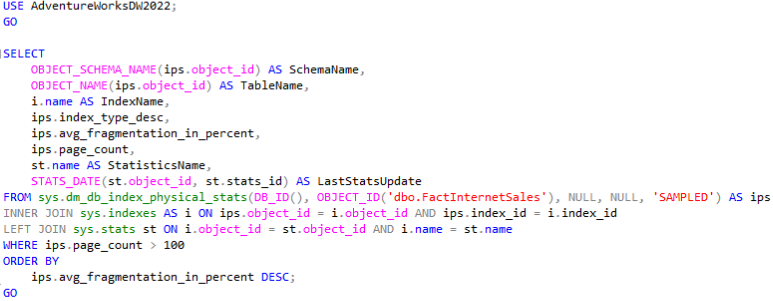
Daha önceki adımda kullanılmayan indeksleri ararken Primary Key olanları hariç tutmuştuk, çünkü PK'ler genellikle tablonun temel erişimi için kritik öneme sahiptir. Ancak, tablodaki tüm indekslerin kullanım istatistiklerine bakmak, bize genel tablo erişim aktivitesi hakkında bir fikir verebilir ve belki gözden kaçan bir durumu ortaya çıkarabilir. Aşağıdaki sorgu bize tüm indeksleri veriyor.



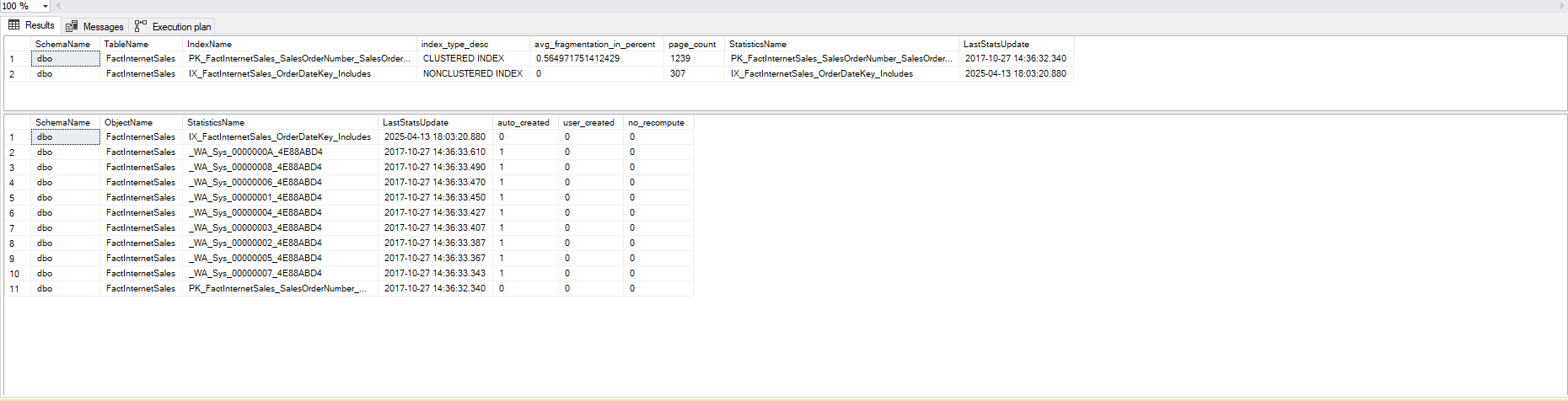
Sonuca bakıldığında her iki indeksin de kullanıldığını yani sistem kaydının bulduğunu görüntüleyebiliyoruz. Bu da hiçbir indeksin silinemeyeceğini gösteriyor.



Bu konuda başka yapabileceğimiz bir şey var mı diye düşünürsek. FactInternetSales tablosundaki indekslerin parçalanma durumunu ve istatistiklerin en son ne zaman güncellendiğini kontrol edebiliriz.



Bu sorgu FactInternetSales tablosundaki indekslerin ortalama parçalanma yüzdesini ve sayfa sayısını gösterir.



PK\_FactInternetSales için p arçalanma %0.56. Bu çok düşük bir oran.

Sayfa Sayısı (page\_count): 1238

İstatistik Güncelleme (LastStatsUpdate): 2017-10-27 14:30.

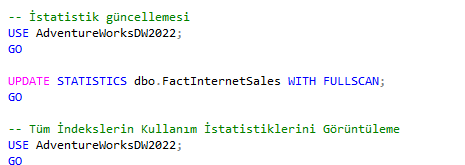
IX\_FactInternetSales

Parçalanma: %0. İndeks yeni oluşturulduğu için parçalanma yok, bu beklenen bir durum.

Sayfa Sayısı (page\_count): 387.

İstatistik Güncelleme (LastStatsUpdate): 2025-04-13 18:08.

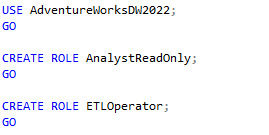
Genellikle %5'in altındaki parçalanma oranları iyi kabul edilir ve müdahale etmeye gerek olmaz.



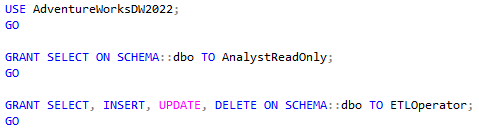
İki indeks arasındaki yıl farkının çok olması sebebiyle eski istatistikleri güncellemek için yukarıdaki sorgu çalıştırılması gerekir. Eski istatistikler, Sorgu İyileştirici'nin sorgular için verimsiz yürütme planları seçmesine neden olabilir.

**Veri Yöneticisi Rolleri**

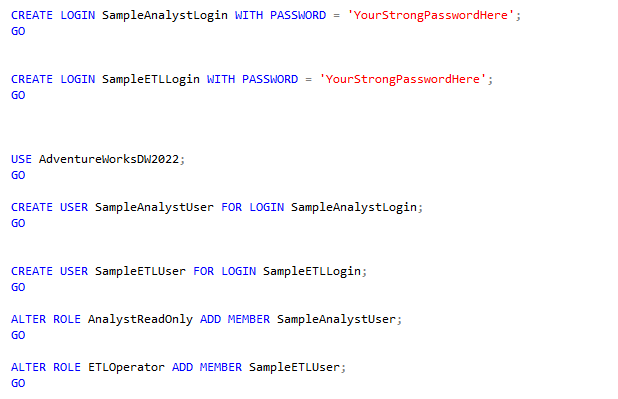
Son adımımız olan veri yöneticisi rollerini gerçekleştirmek için de birtakım sorgular çalıştırabiliriz.



Bu sorgu veritabanımızda AnalystReadOnly ve ETLOperator isimli iki boş rol oluşturur.



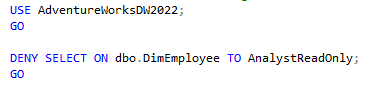
Bu sorgu, AnalystReadOnly rolüne dbo şemasındaki her şey üzerinde sadece SELECT yapma hakkı, ETLOperator rolüne ise SELECT, INSERT, UPDATE, DELETE yapma hakkı tanıyacaktır.



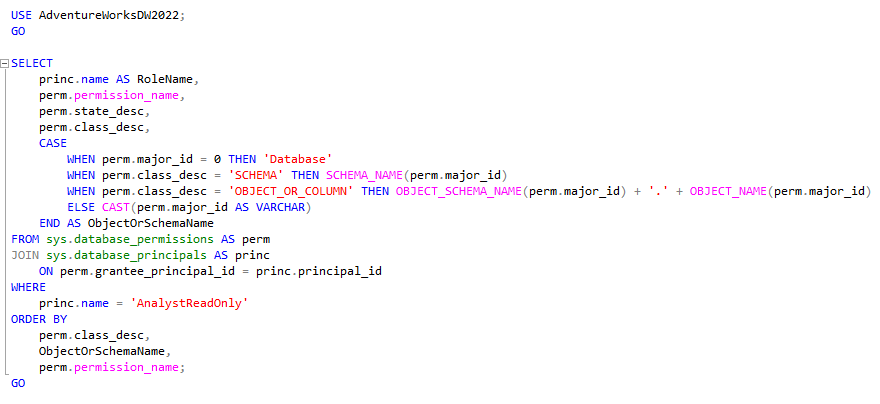
Yukarıdaki sorgunun çalıştırılmasından sonra

SampleAnalystLogin ile sunucuya bağlanan bir kişi, AdventureWorksDW2022 veritabanında SampleAnalystUser olarak tanınacak ve AnalystReadOnly rolünün izinlerine (yani dbo şeması üzerinde sadece SELECT) sahip olacaktır. Bu kullanıcı INSERT, UPDATE, DELETE yapmaya çalıştığında hata alacaktır.

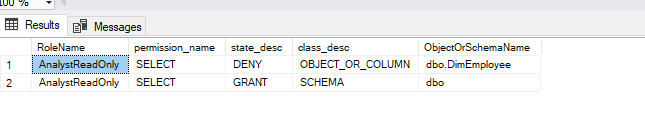
SampleETLLogin ile sunucuya bağlanan bir kişi/uygulama, SampleETLUser olarak tanınacak ve ETLOperator rolünün izinlerine (yani dbo şeması üzerinde SELECT, INSERT, UPDATE, DELETE) sahip olacaktır.



Bu sorgu çalıştırıldıktan sonra, AnalystReadOnly rolüne atanmış bir kullanıcı dbo.DimEmployee tablosundan veri çekmeye çalıştığında artık izin hatası alacaktır, ancak dbo şemasındaki diğer tablolardan veri çekmeye devam edebilecektir.



AnalystReadOnly rolünde hangi izinlerin atandığını görmek için yukarıdaki sorgu çalıştırılabilir.



Yukarıdaki sonuçta da görüldüğü gibi mevcut izinler atadığımız gibi.