**Name – Davender Singh**

**Student ID – 125700211**

**Course – DBW624V1A**

**Professor – Les King**

To enhance the performance and scalability of a data warehouse that is expected to grow significantly, a variety of strategies can be employed. Here are five different kinds of performance enhancements, along with their descriptions and SQL/Database commands:

**1. Indexing**

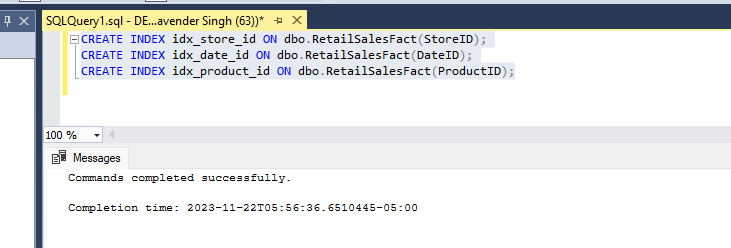
**Description and Benefits:** Creating indexes on columns that are frequently used in JOIN, WHERE, and ORDER BY clauses can significantly speed up query performance by allowing the database engine to find data more efficiently.

**SQL Command:**

CREATE INDEX idx\_store\_id ON dbo.RetailSalesFact(StoreID);

CREATE INDEX idx\_date\_id ON dbo.RetailSalesFact(DateID);

CREATE INDEX idx\_product\_id ON dbo.RetailSalesFact(ProductID);



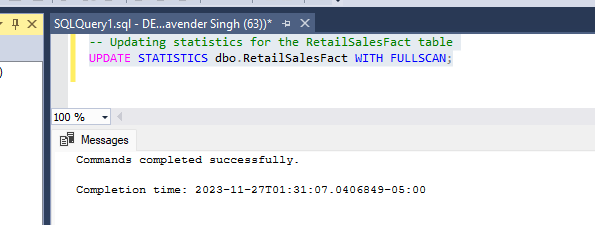
**2. Optimizing and Updating Statistics**

**Description and Benefits:** Statistics are used by the SQL Server query optimizer to determine the most efficient way to execute a query. Keeping statistics up-to-date ensures the optimizer has accurate data distribution information.

**SQL Command:**

-- Updating statistics for the RetailSalesFact table

UPDATE STATISTICS dbo.RetailSalesFact WITH FULLSCAN;



**3. Implementing Data Compression**

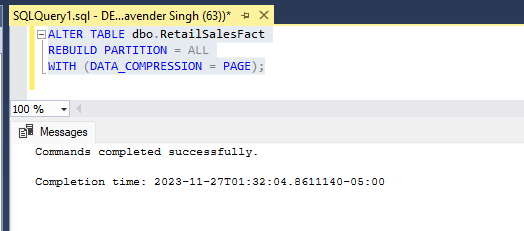
**Description and Benefits:** Data compression can reduce the storage footprint and improve I/O performance by reducing the number of pages needed to store the data.

**SQL Command:**

ALTER TABLE dbo.RetailSalesFact

REBUILD PARTITION = ALL

WITH (DATA\_COMPRESSION = PAGE);



**4. Creating Indexed Views**

**Description and Benefits:** Indexed views can pre-calculate and store complex aggregations and joins. They can significantly improve the performance of queries that can utilize these precomputed results.

**SQL Command:**

CREATE VIEW SalesSummary

WITH SCHEMABINDING

AS

SELECT

StoreID,

SUM(ISNULL(SalesRevenue, 0)) AS TotalSales,

COUNT\_BIG(\*) AS TotalRows

FROM

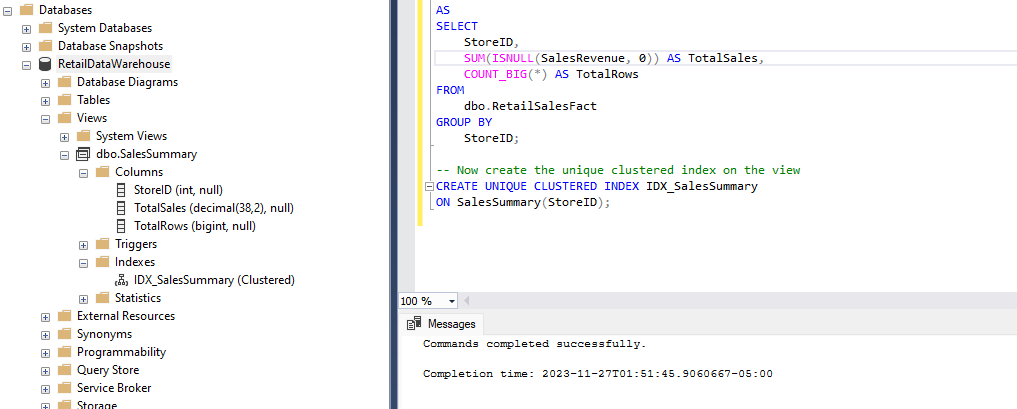
dbo.RetailSalesFact

GROUP BY

StoreID;

CREATE UNIQUE CLUSTERED INDEX IDX\_SalesSummary

ON SalesSummary(StoreID);



**5. Asynchronous Statistics Update**

**Description and Benefits:** Asynchronous statistics updates allow queries to continue using existing statistics even when those statistics are out-of-date, while SQL Server updates them in the background. This avoids query delays associated with synchronous statistics updates, which can be particularly beneficial for large tables.

**SQL Command:**

ALTER DATABASE RetailDataWarehouse

SET AUTO\_UPDATE\_STATISTICS\_ASYNC ON;

