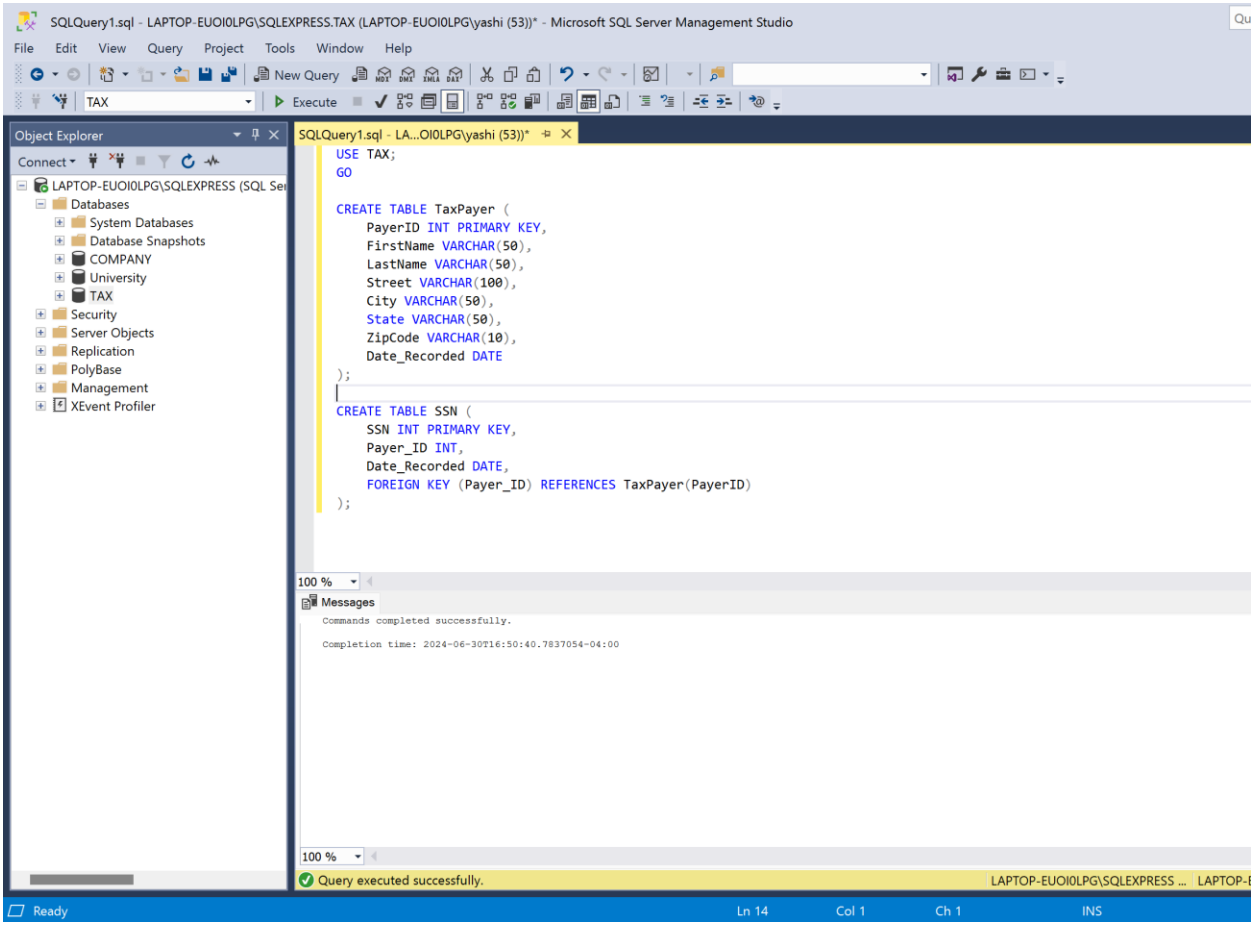
# CIS 552: DATABASE DESIGN

Submitted by – Yashika Patil

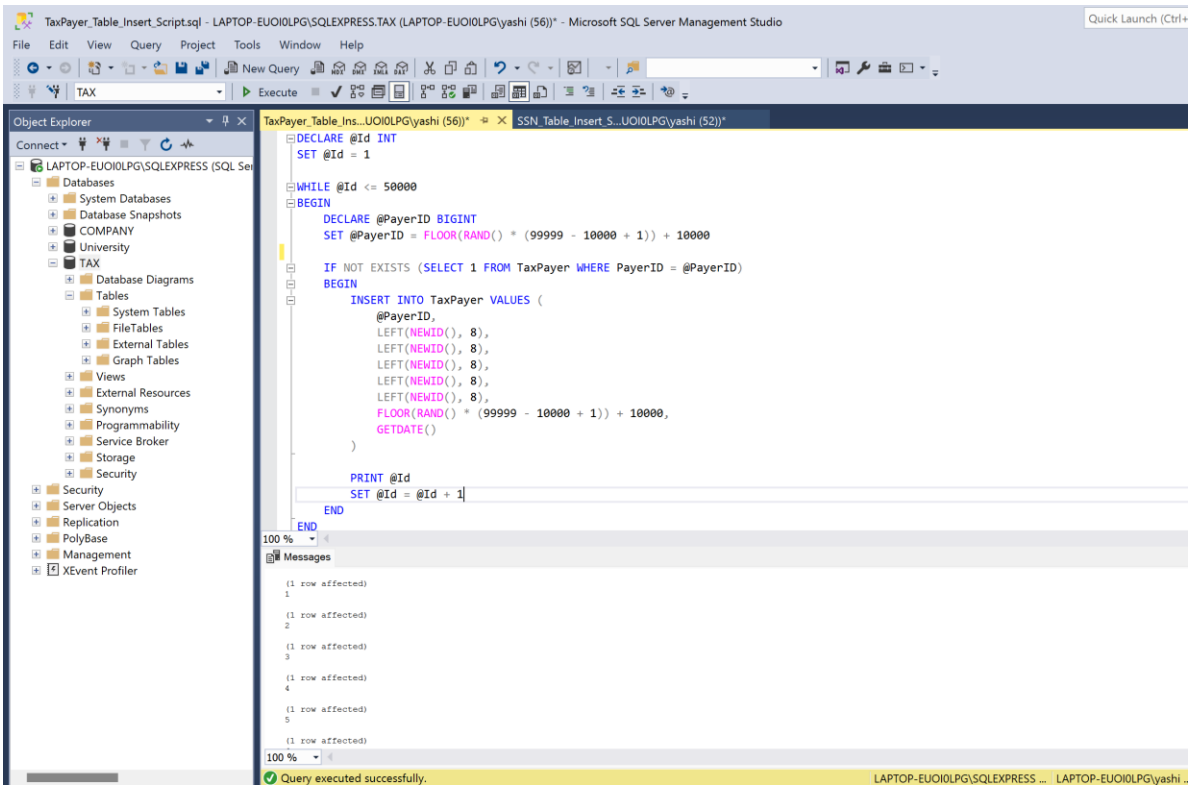
## Lab - 4

## Task 1: Performance Tuning of Queries using Indexes

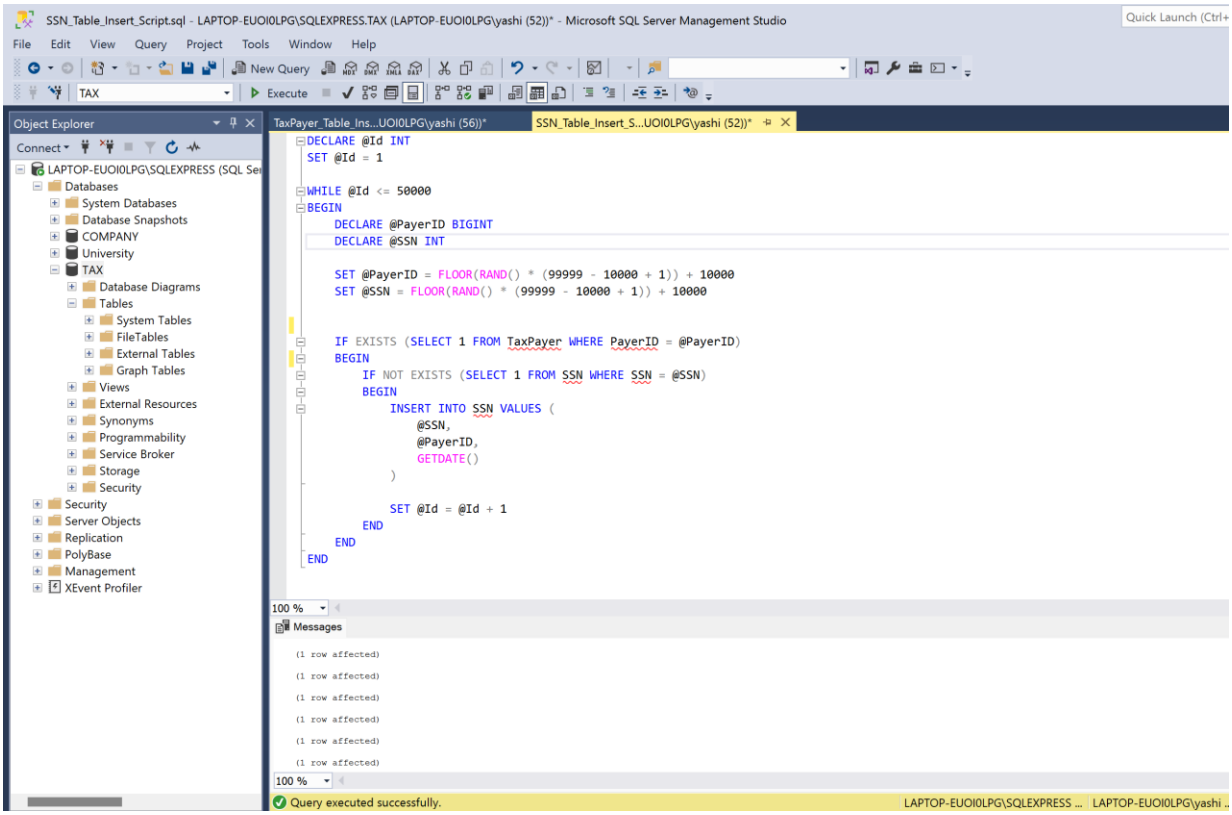
1. **Database and Table Generation:** I generated a database called TAX and created tables according to the provided schema, ensuring all columns and constraints were included.




2. **Data Population:** I populated the tables with 50,000 rows each using the provided SQL queries from the Lab 4 manual, TaxPayer_Table_Insert_Script.sql and SSN_Table_Insert_Script.sql,
    a. Execution of both the queries: ‘TaxPayer_Table_Insert_Script.sql’

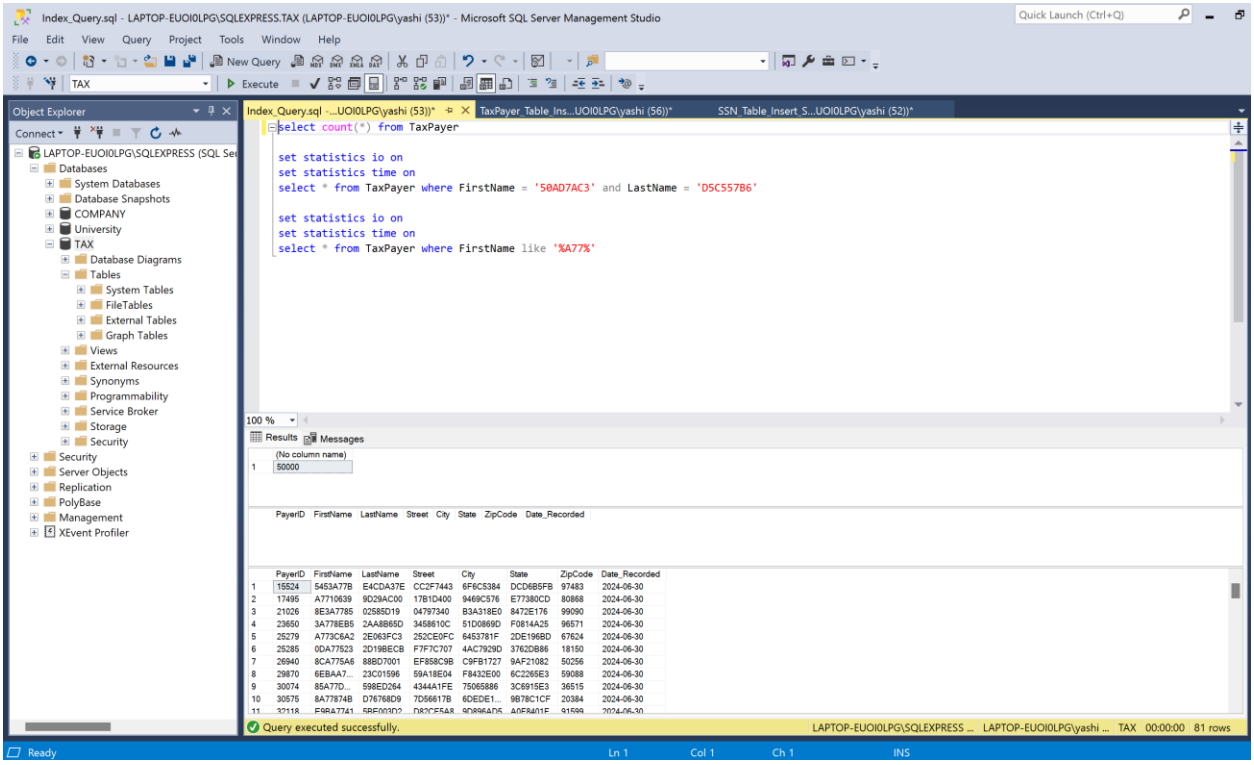



b. Execution of both the queries: ‘SSN_Table_Insert_Script.sql'

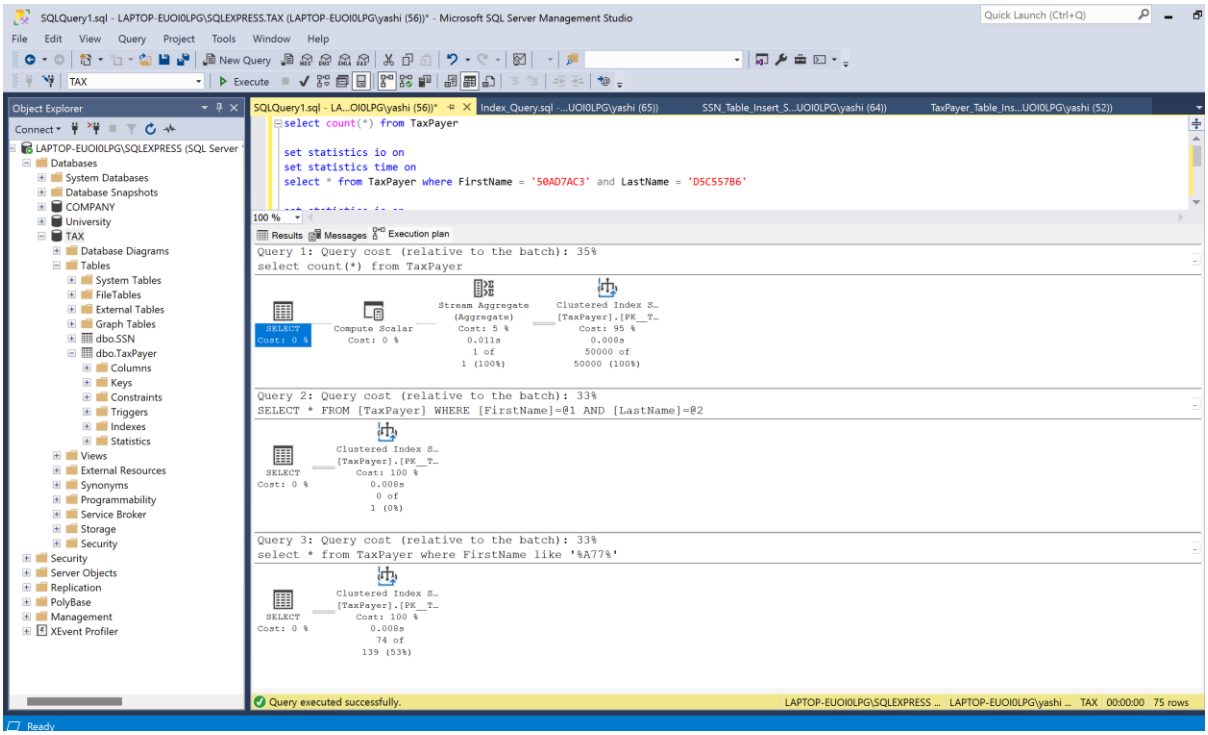


### 3. Without non-clustered index performance analysis:

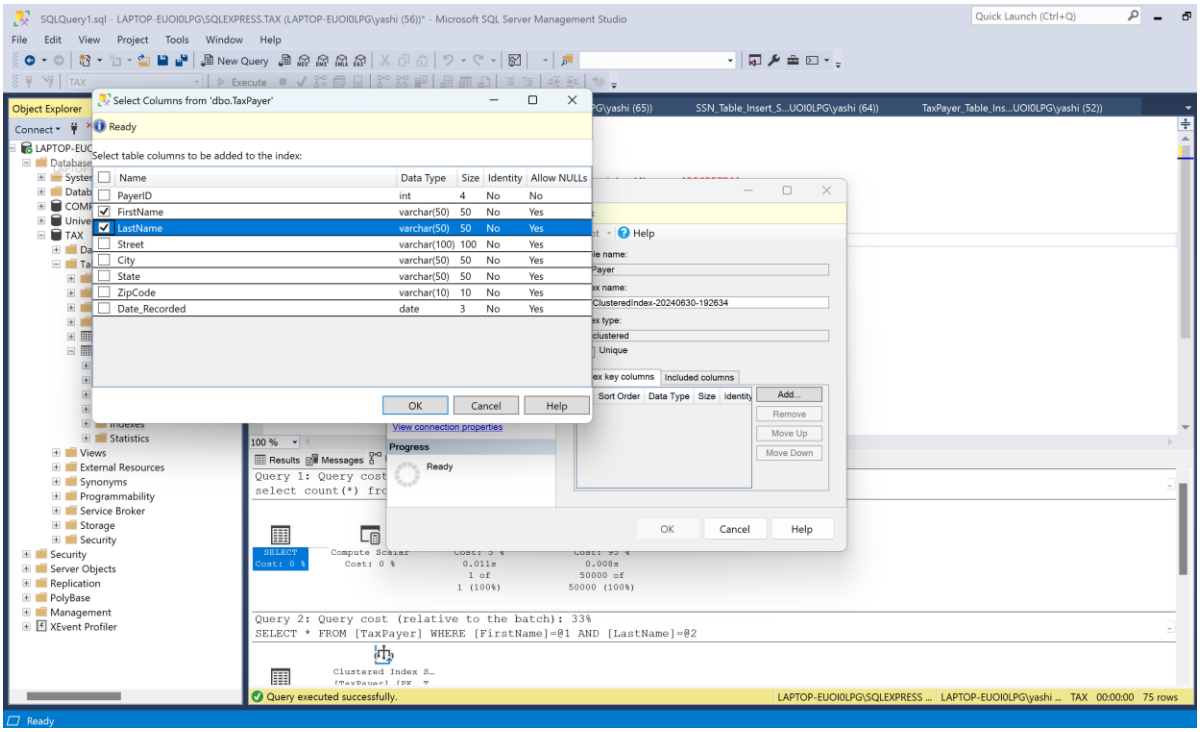
I executed the Index_query.sql script to observe the initial query performance without any indexes.



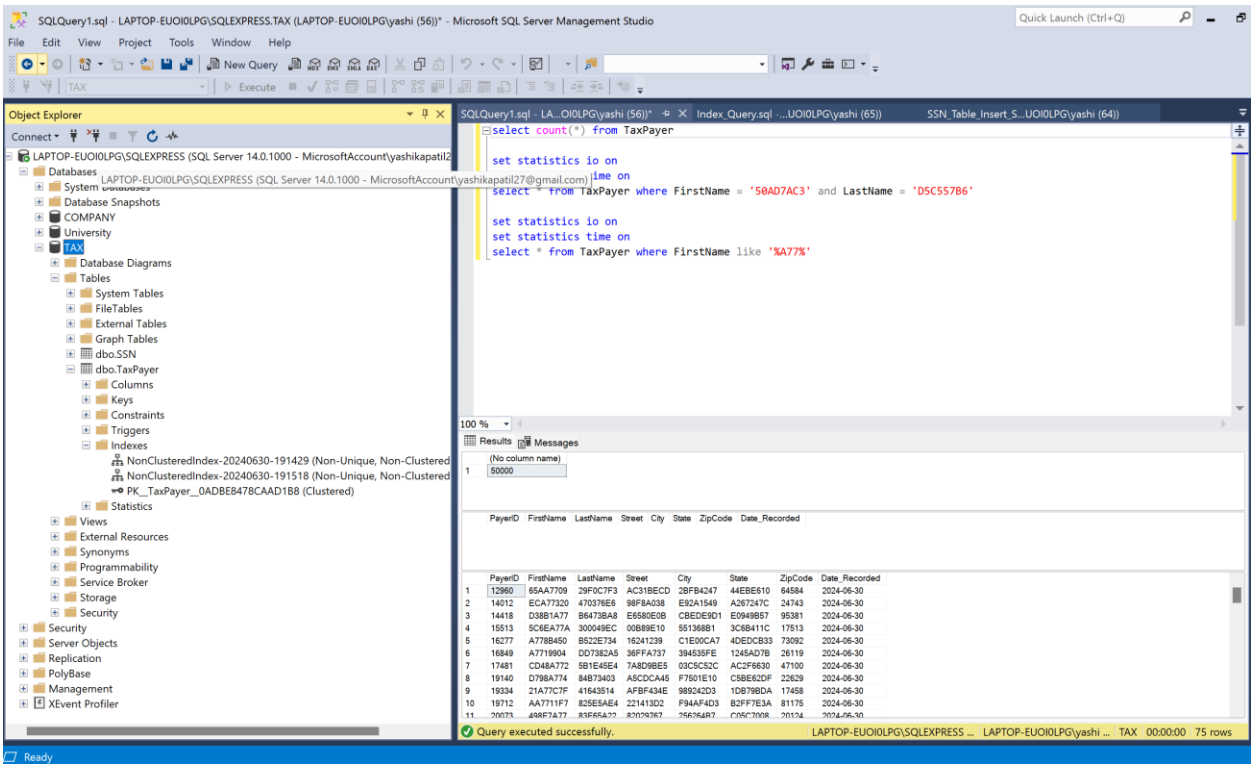

**Execution plan without non-clustered:** The execution plan before adding any indexes showed longer execution times and higher resource consumption.



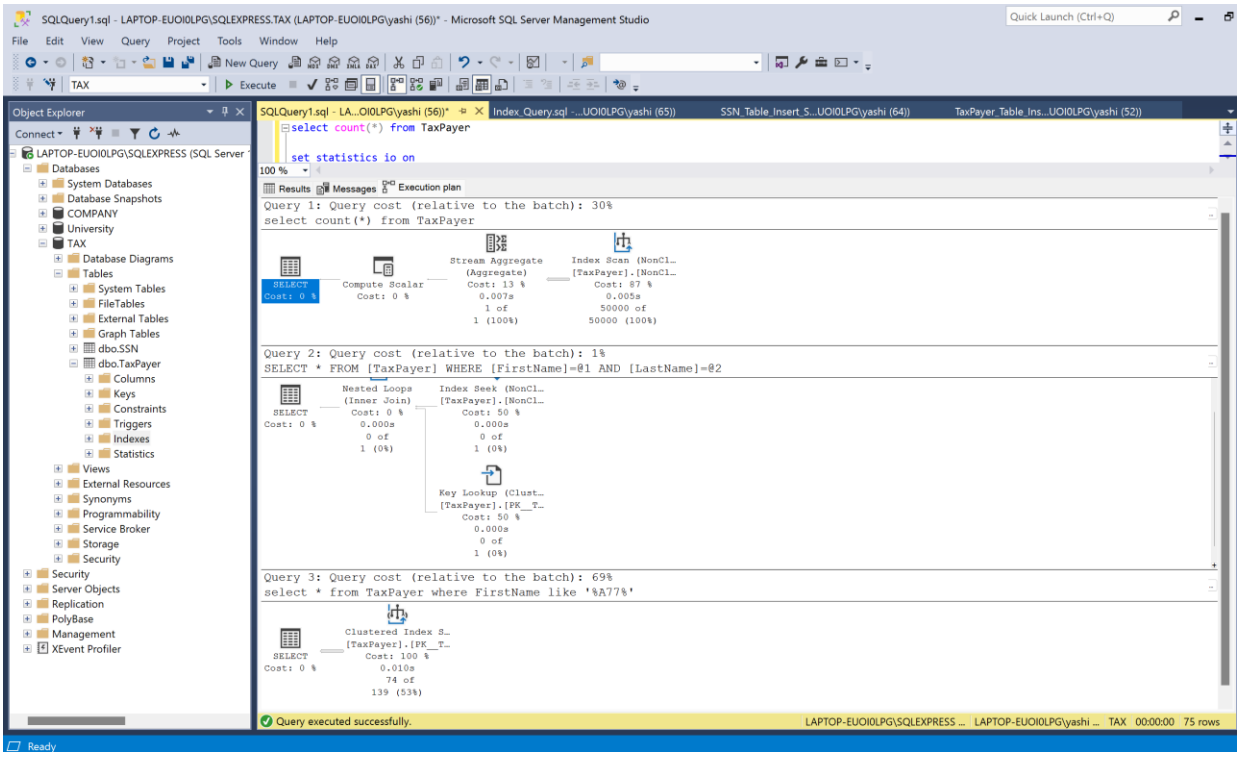

4. **Creating Non-Clustered Index:** I created a non-clustered index on the FirstName and LastName columns of the TaxPayer table by right-clicking on the 'Indexes' section, selecting 'New index', choosing 'Non-Clustered Index', and adding the specified columns.



After selecting non-clustered index, on refreshing I could see the column names in the indexes as:



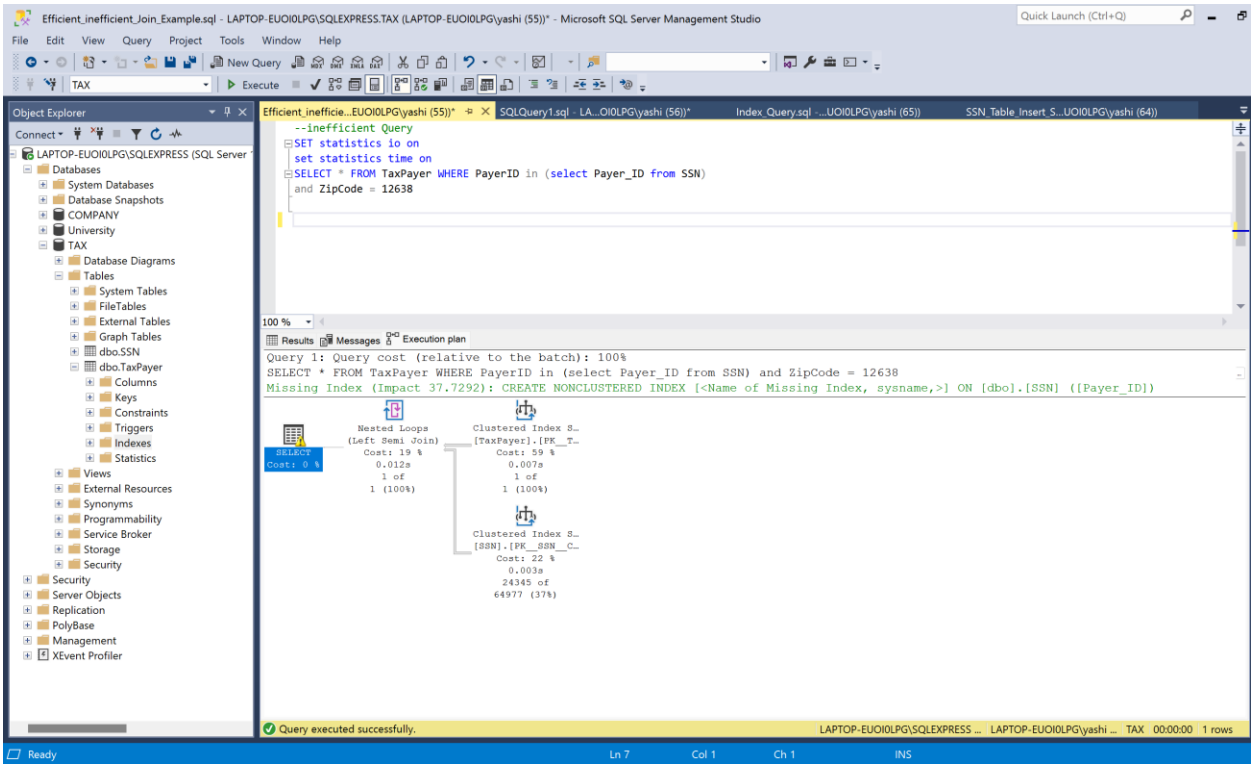

**Executed the query again with execution plan.**



This execution plan showed significant improvements in query performance, with reduced logical reads and less execution times.

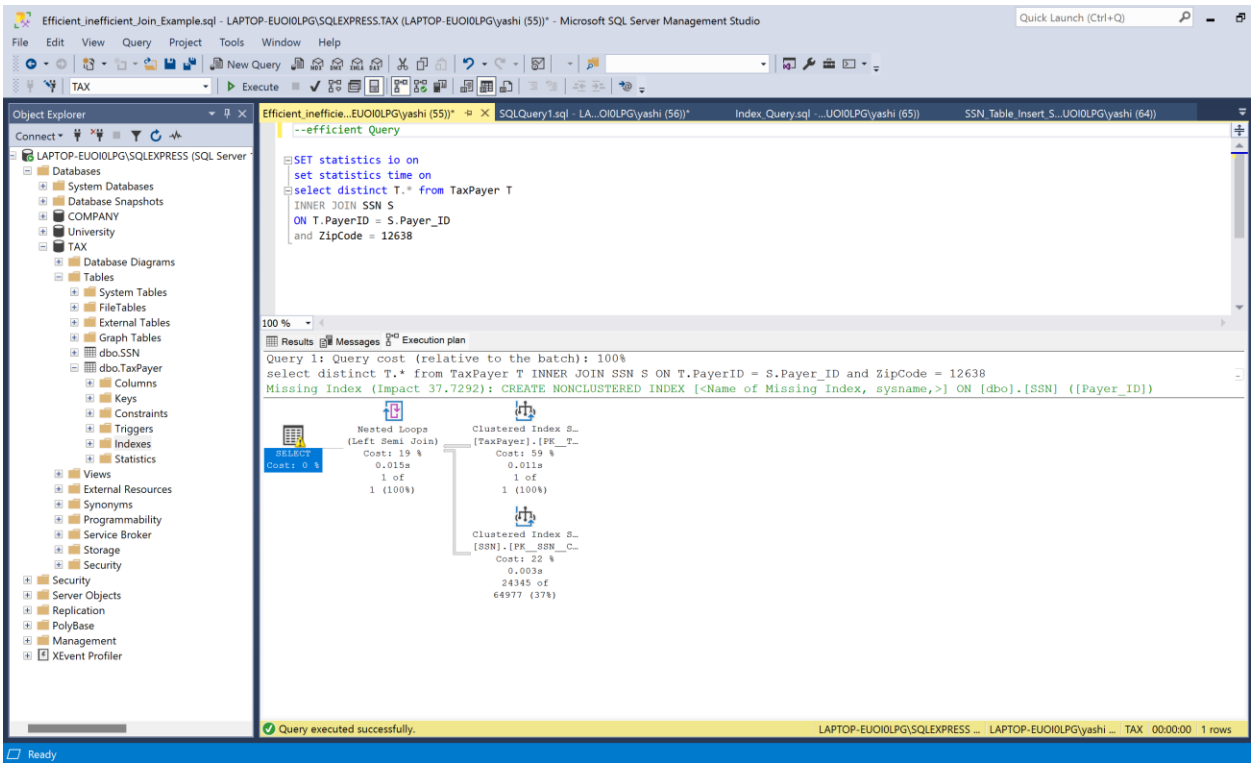
**Task 2 - Query Optimization:**

**Inefficient Query:** By executing the inefficient query from 'Efficient_inefficient_Join_Example.sql':




The inefficient queries showed higher execution times and more resource usage in all steps.

**Efficient Query:** The efficient queries showed lower execution times and more optimal resource usage.




## Conclusion:

- Understood the importance of database performance tuning and query optimization.
    - Initial queries without indexes showed high resource use and long execution times.
    - Execution plans before indexing indicated high logical reads and extended durations.
- Observed how generating indexes and optimizing queries can improve the process of data retrieval.
    - Creating a non-clustered index on the FirstName and LastName columns significantly boosted query performance.
    - Post-index execution plans showed reduced logical reads and faster execution times.
- Understood the concept of indexes, both non-clustered and clustered, in improving query performance by reducing logical reads and execution time.
- Learned to write optimized SQL queries for efficient database systems.
- The comparison between efficient and inefficient queries signifies the importance of writing optimized SQL queries for better performance. This has deepened my understanding of database management and of how strategic indexing and query writing can lead to more resource-efficient database systems.
- Learned to interpret execution plans for diagnosing and improving query performance. Recognized the value of execution plans in guiding performance tuning decisions.

## References:


1. Lab Manual – 4 uploaded on MyCourses.
2. https://youtu.be/YuRO9-rOgv4?si=Aj8euQKCy7OfmXtL