1. SQL Query Execution Order

\* Write the sequence in which a SQL query is executed (FROM, WHERE, GROUP BY, etc.).

\*Understand how queries are processed step by step.

* **FROM** – Specifies the tables involved in the query.
* **JOIN** – Combines rows from multiple tables based on a related column.
* **WHERE** – Filters rows before any grouping is done.
* **GROUP BY** – Groups rows that have the same values in specified columns.
* **HAVING** – Filters groups based on aggregate conditions.
* **SELECT** – Specifies the columns to be returned.
* **DISTINCT** – Removes duplicate rows from the result set.
* **ORDER BY** – Sorts the result set.
* **LIMIT / OFFSET** – Restricts the number of rows returned.

2. Practice SQL Basics

\* Difference between DELETE vs. TRUNCATE commands.

|  |  |  |
| --- | --- | --- |
| Feature | Delete | Truncate |
| Scope | Removes specific rows (WHERE allowed) | Removes all rows (no WHERE allowed) |
| Speed | Slower (logs each row deletion) | |  | | --- | |  |  |  | | --- | | Faster (deallocates pages) | |
| Transaction Logging | Logs each row deletion | Minimal logging (DDL operation) |
| Rollback Support | Can be rolled back (ROLLBACK) | Cannot be rolled back (commits immediately) |
| Resets Identity Column | No | Yes |
| Type of Command | **DML** (Data Manipulation Language) | **DDL** (Data Definition Language) |

\* Create two sample tables and insert data.

1. create table ndemo(ID INT Identity(1,1),Nam VARCHAR(100),country VARCHAR(50))

insert into ndemo values('Nisha','Canada')

insert into ndemo values('Ajaey','USA')

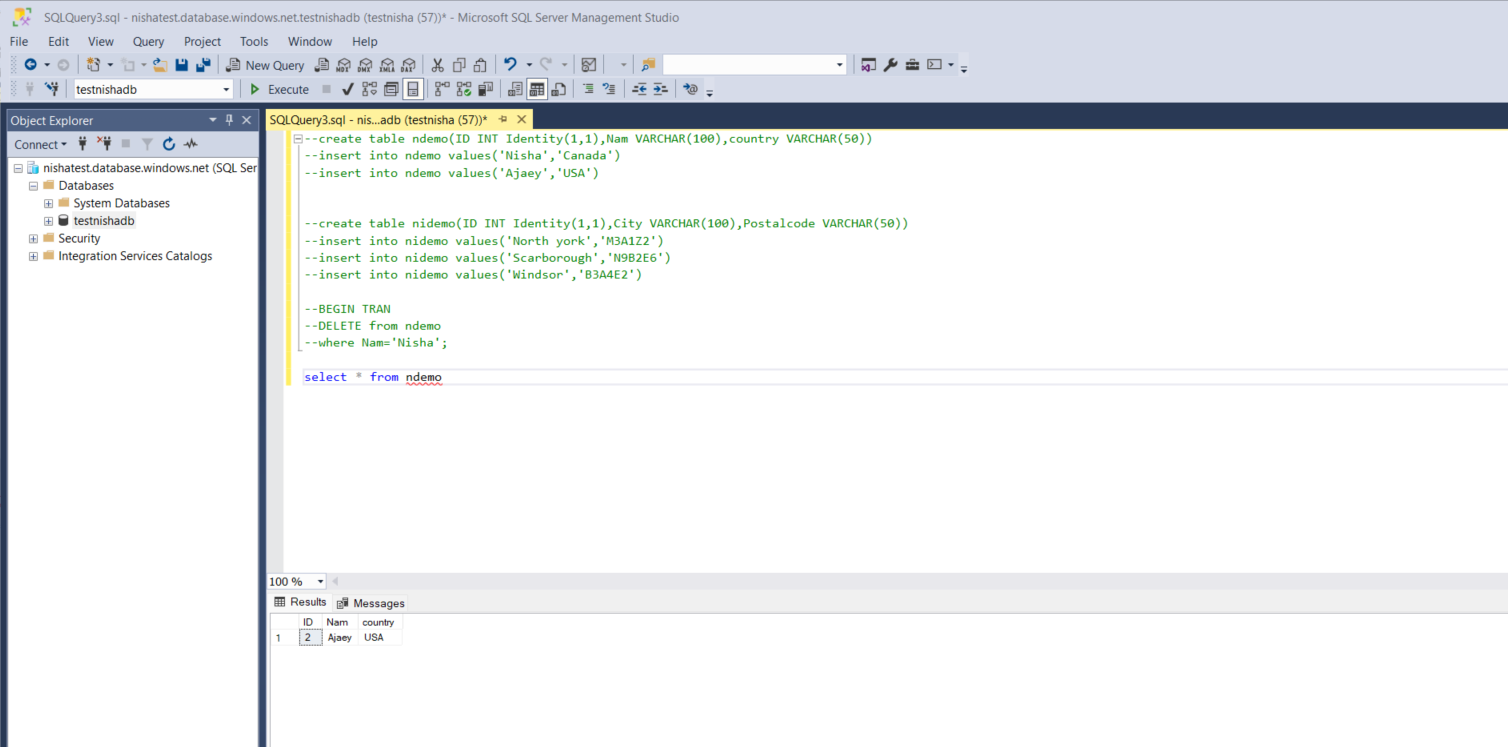
2. create table nidemo(ID INT Identity(1,1),City VARCHAR(100),Postalcode VARCHAR(50))

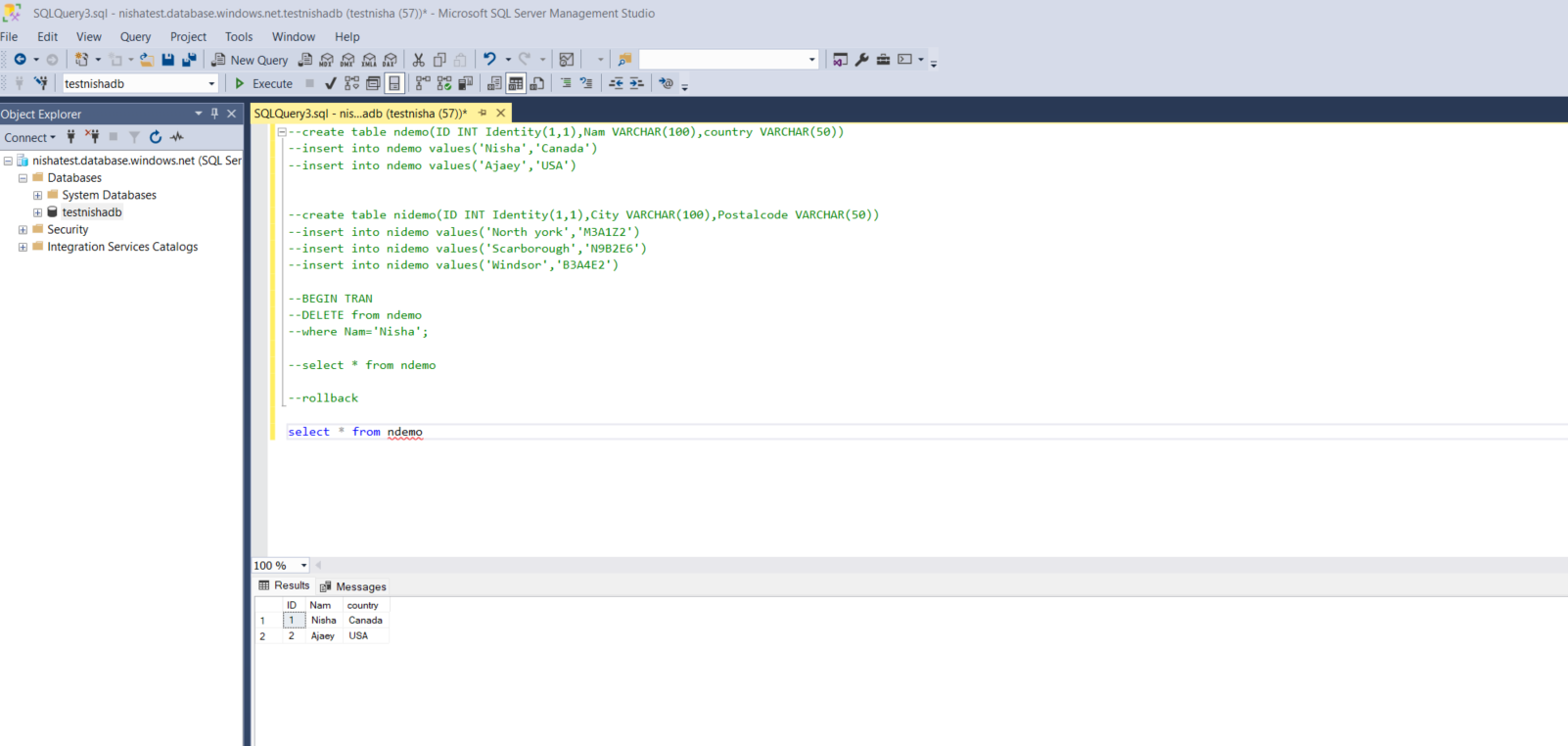
insert into nidemo values('North york','M3A1Z2')

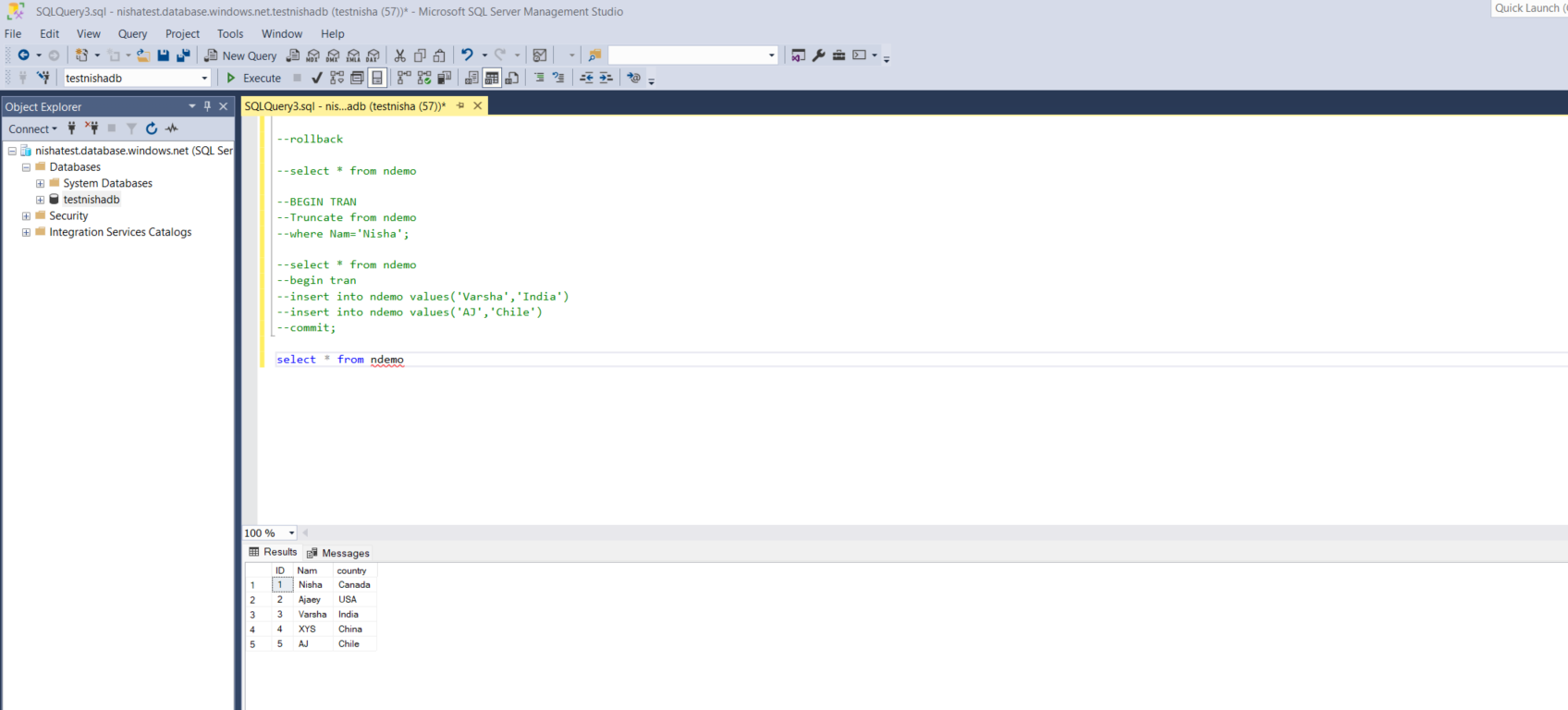
insert into nidemo values('Scarborough','N9B2E6')

insert into nidemo values('Windsor','B3A4E2')

\* Use identity columns and practice transactions (BEGIN TRANSACTION, COMMIT, ROLLBACK).



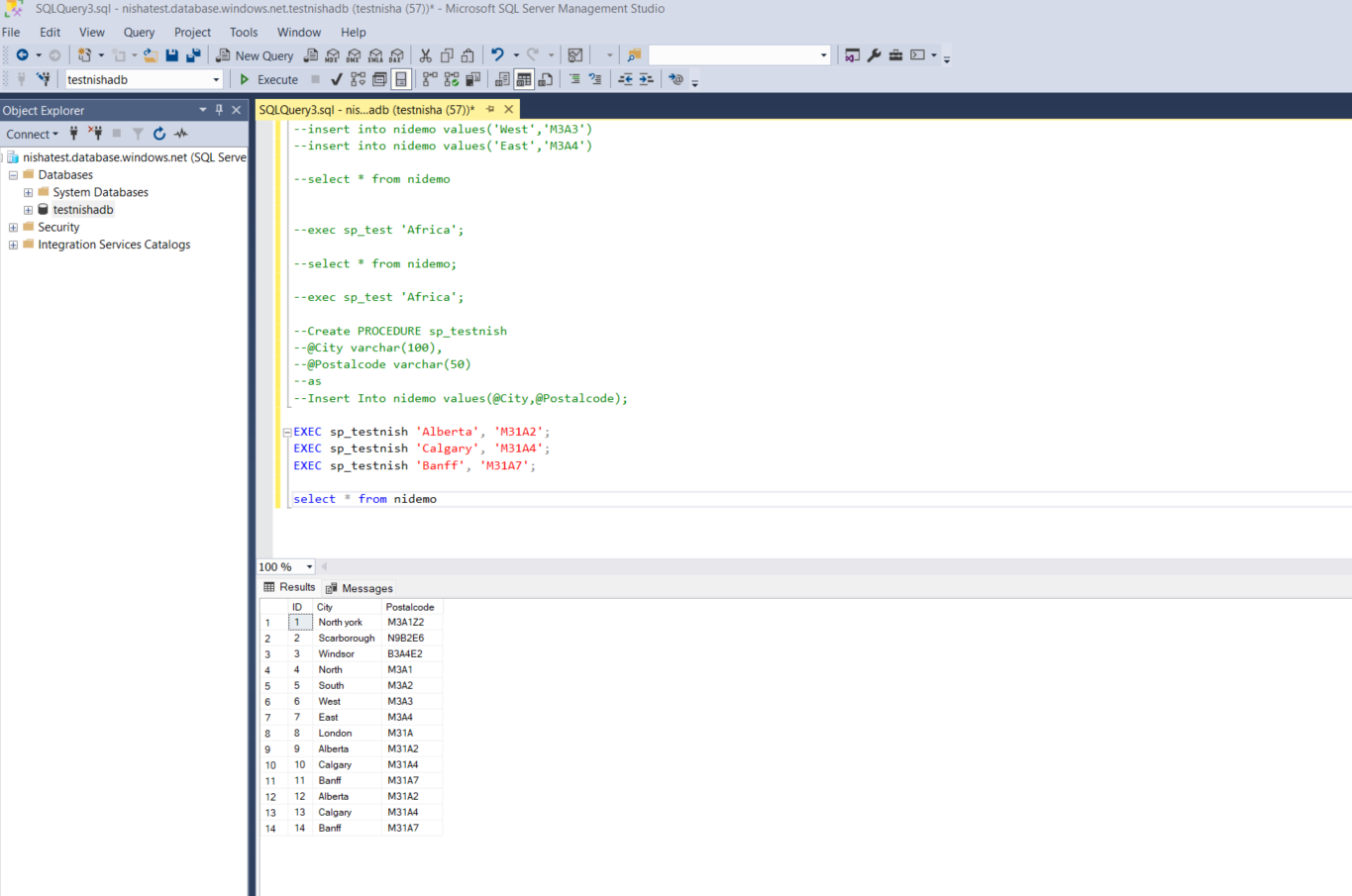




3. Stored Procedures

\* Create a stored procedure and execute it with parameters.

\* Write a real-world use case for stored procedures.



4. Data Types and Character Handling

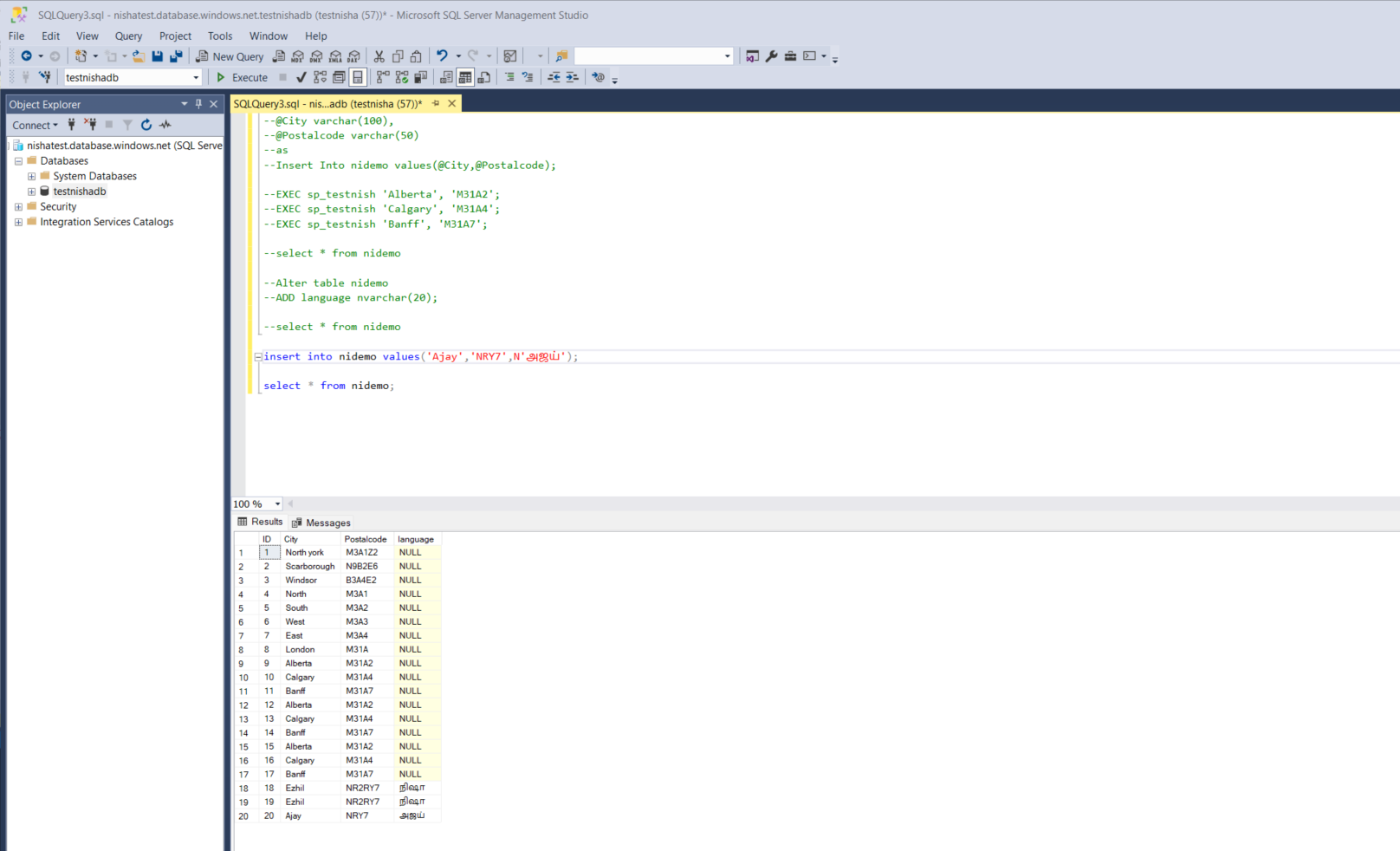
\* Study various SQL data types (integer, float, date, time).

|  |  |
| --- | --- |
| CATEGORY | COMMON DATA TYPES |
| Numeric | INT, BIGINT, DECIMAL, FLOAT |
| Date & time | DATE, TIME, DATETIME, TIMESTAMP |
| String | CHAR, VARCHAR, TEXT |
| Boolean | BIT (0 or 1) |
| Others | BLOB, JSON, UUID |

\* Understand the difference between nvarchar vs. varchar with examples.

|  |  |  |
| --- | --- | --- |
| Features | Varchar | Nvarchar |
| Storage | 1 byte per character (non-Unicode) | 2 bytes per character (Unicode) |
| Supports Unicode | No (Only supports single-byte character sets) | Yes (Supports multilingual & special characters) |
| When to Use | When storing plain English text | When storing multilingual data (e.g., Chinese, Arabic, emojis) |
| Syntax | |  | | --- | |  |  |  | | --- | | VARCHAR(n) | | NVARCHAR(n) |

\* Insert multilingual data using nvarchar.



5. Query Optimization

\* Analyze how queries execute and optimize them for efficiency.

\* Review execution plans and indexing strategies.

Optimizing SQL queries involves understanding the **execution plan**, **indexing strategies**, and applying best practices such as:

* Using proper **indexes** for JOIN and WHERE clauses.
* Minimizing the use of **subqueries**, **DISTINCT**, and unnecessary **functions** on indexed columns.
* Analyzing the execution plan to spot **bottlenecks** and improving them by reducing full table scans, switching joins, or modifying indexes.

6. Hands-on Practice

\* Apply all discussed SQL concepts in practical exercises.