B2L1

Block 2 Lektion 1

SQL Server, SSMS and T-SQL

- MS SQL Server - instead of .NET
  - RDBMS - Relational Database Management System
  - database engine
- MS SSMS - instead of Visual Studio
  - Microsoft SQL Server Management Studio (SSMS) is an integrated environment for management of server databases
- T-SQL - instead of C#
  - An imperative Procedural Programming Language (adds functionality and types)
  - All tools and applications that communicate with a SQL Server database do so by sending T-SQL commands.

T-SQL

Transact-SQL

Purpose of Module

- Gain a basic oversight and understanding of
  - Tables
  - Schemas
  - T-SQL Syntax

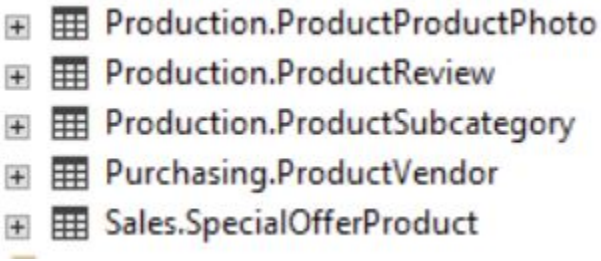
- You will learn to write queries and to reference the documentation progressively throughout the remaining lessons.

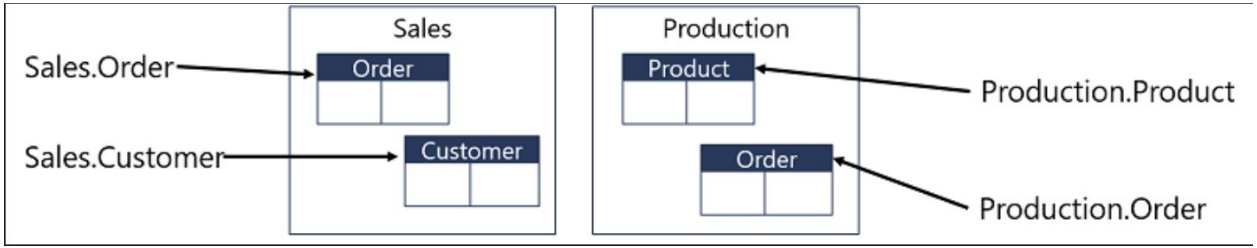
- Provide a starting point and links to additional resources

Schemas & Tables - Prefix table names with schemas

- Think of Schemas like namespaces in C#.
- Hierarchical naming system: [namespace].[table]
- Fully qualified name *example:*
  Server1.StoreDB.Sales.Order.
  [server_name].[database_name].[schema].[table_name]
- Use department or other logical unit for naming schemas

MS SSMS Object Explorer view:





Schemas & Tables

- Create database
- Connect to database
- Create schema
- Create table using schema

```sql
CREATE DATABASE MyDatabase;
GO

USE MyDatabase
GO

CREATE SCHEMA MySchema
GO

CREATE TABLE MySchema.MyTable
(
    ID int IDENTITY(1,1) PRIMARY KEY NOT NULL,
    [Name] varchar(50) NULL,
)
GO
```

SQL Statements

SQL Statements

- Data Manipulation Language (DML)
  - query and modify data. (CRUD operations)
  - *Examples:* INSERT, UPDATE, and DELETE.
- Data Definition Language (DDL)
  - life cycle management of database objects (tables, views, and procedures).
  - *Examples*: CREATE, ALTER, and DROP.
- Data Control Language (DCL)
  - Manage security permissions for users and objects
  - *Examples:* GRANT, REVOKE, DENY

Clauses (Query filter conditions)

- Every clause acts as an elimination process/filter reducing the *number of records returned* (result-set)
- Sandbox example.

Select Statement (DML) with multiple clauses (filter conditions)

```sql
SELECT OrderDate, COUNT(OrderID) AS Orders
FROM Sales.SalesOrder
WHERE Status = 'Shipped'
GROUP BY OrderDate
HAVING COUNT(OrderID) > 1
ORDER BY OrderDate DESC;
```

SQL Statements - Runtime evaluation

- SQL syntax is designed to read like “prose” (see query example)
- Runtime evaluation order differs from query (se runtime example)

[Evaluation of Statements with multiple clauses at runtime:](#)

FROM => Virtual table created
WHERE => filtered
GROUP BY => new virtual table based on predicate
HAVING => new virtual table based on predicate
SELECT => result-set
ORDER BY => result-set by OrderDate

NOTE!

The evaluation order is important as new virtual tables will be created based on each predicate in turn determining what data is available for the next clause.

Think Yoda :-)

```sql
--T-SQL Query:
SELECT OrderDate, COUNT(OrderID) AS Orders
FROM Sales.SalesOrder
WHERE Status = 'Shipped'
GROUP BY OrderDate
HAVING COUNT(OrderID) > 1
ORDER BY OrderDate DESC;

-- Runtime evaluation order:
FROM Sales.SalesOrder
WHERE Status = 'Shipped'
GROUP BY OrderDate
HAVING COUNT(OrderID) > 1
SELECT OrderDate, COUNT(OrderID) AS Orders
ORDER BY OrderDate DESC;
```

Query

Runtime evaluation

Literals & Identifiers in T-SQL

Syntax:

- **String literals:** ‘single quotes’ , ‘John smith’
- **numeric literals:** 1 , 1234
- **Identifiers:** ProductID (a column), Sales.SalesOrderHeader (schema.table)
- **String concatenation** (‘+’) vs **addition** (+) and **subtraction** (-) just like in C#!

```
SELECT (LastName + ', ' + FirstName) AS Name
FROM Person.Person
ORDER BY LastName ASC, FirstName ASC;
```

```
SELECT 'The order is due on ' + CONVERT(VARCHAR(12), DueDate, 101)
FROM Sales.SalesOrderHeader
WHERE SalesOrderID = 50001;
```

Syntax, formatting and naming conventions

- Capitalize T-SQL keywords,
  - e.g. SELECT, FROM, AS, etc.
  - Capitalizing keywords is a commonly used convention that makes it easier to find each clause of a complex statement
- Start a new line for each major clause of a statement.
- If the SELECT list contains more than a few columns, expressions, or aliases, consider listing each column on its own line.
- Indent lines containing subclauses or columns to make it clear which code belongs to each major clause.

A few columns:

```
SELECT (LastName + ',' + SPACE(1) + SUBSTRING(FirstName, 1, 1) + '.') AS Name, e.JobTitle
FROM Person.Person AS p
    JOIN HumanResources.Employee AS e
    ON p.BusinessEntityID = e.BusinessEntityID
WHERE e.JobTitle LIKE 'Vice%'
ORDER BY LastName ASC;
```

Column on its own line:

```
CREATE TABLE dbo.Customers
(
   CustomerId        INT    NOT NULL   PRIMARY KEY, -- primary key column
   Name      [NVARCHAR](50)  NOT NULL,
   Location  [NVARCHAR](50)  NOT NULL,
);
```

Module References

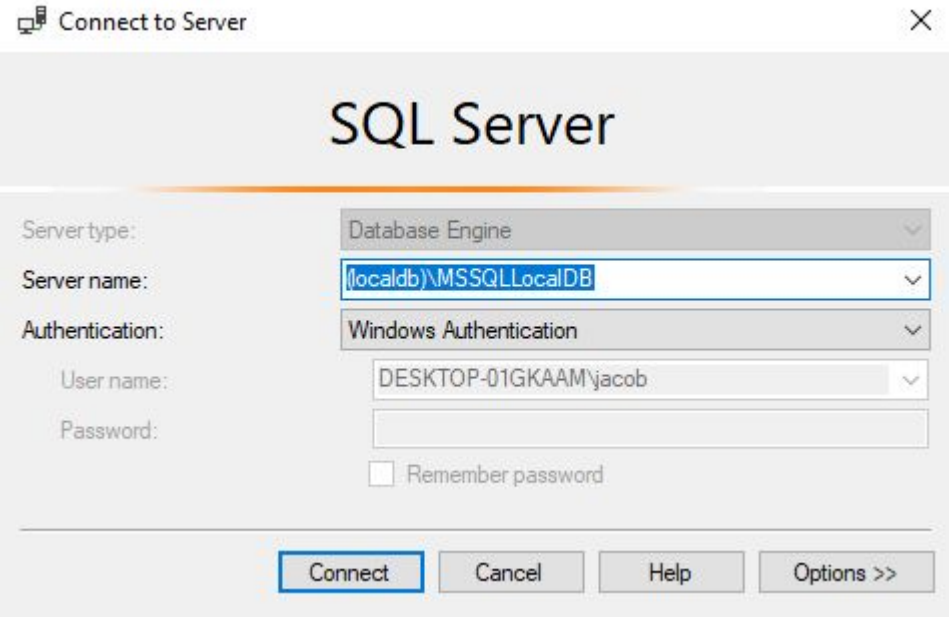
- T-SQL Tutorial: Write Transact-SQL statements - SQL Server | Microsoft Learn
- Transact-SQL Syntax Conventions (Transact-SQL) - SQL Server | Microsoft Learn
- SQL Sandbox: SQL Tutorial
- Get Started Querying with Transact-SQL - Training | Microsoft Learn

MS SSMS

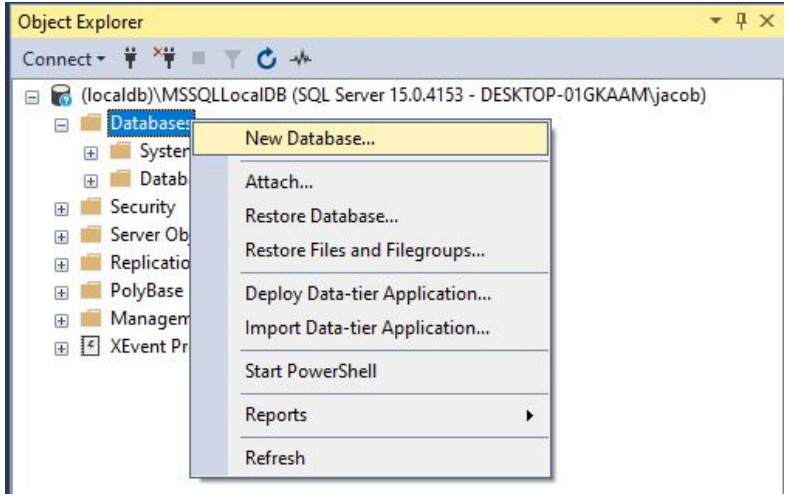
Sequel Server Management Studio

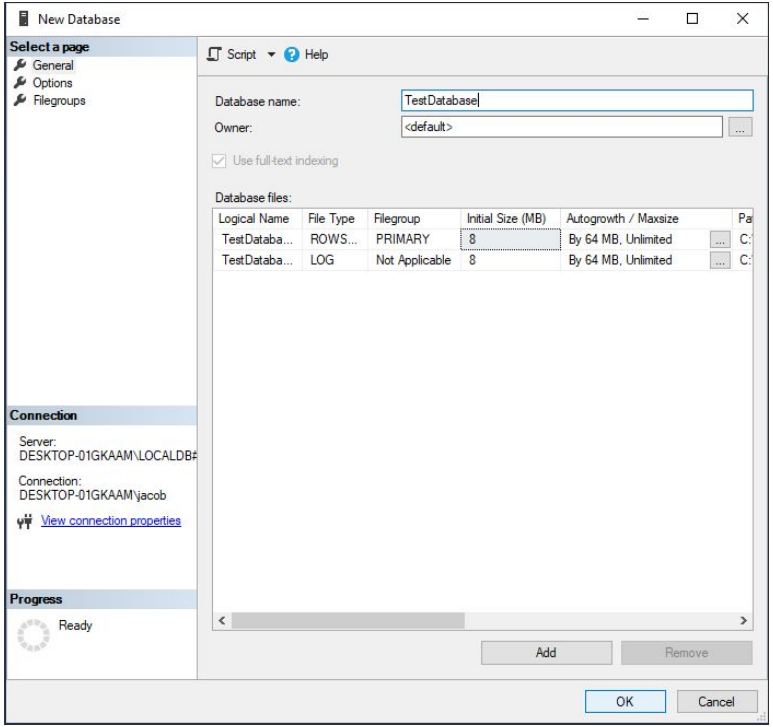
SSMS: Connect to SQL Server

- **Server name**: (localdb)\MSSQLLocalDB
  - Lightweight version useful for development since SQL Server instances are designed to utilise all available resources
  - Alternative server instance: “localhost” OR “.”
- **Authentication**: Windows Authentication
- Connect!

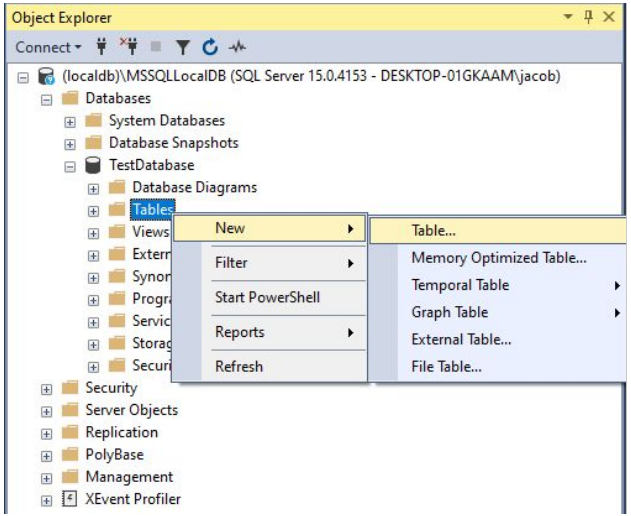


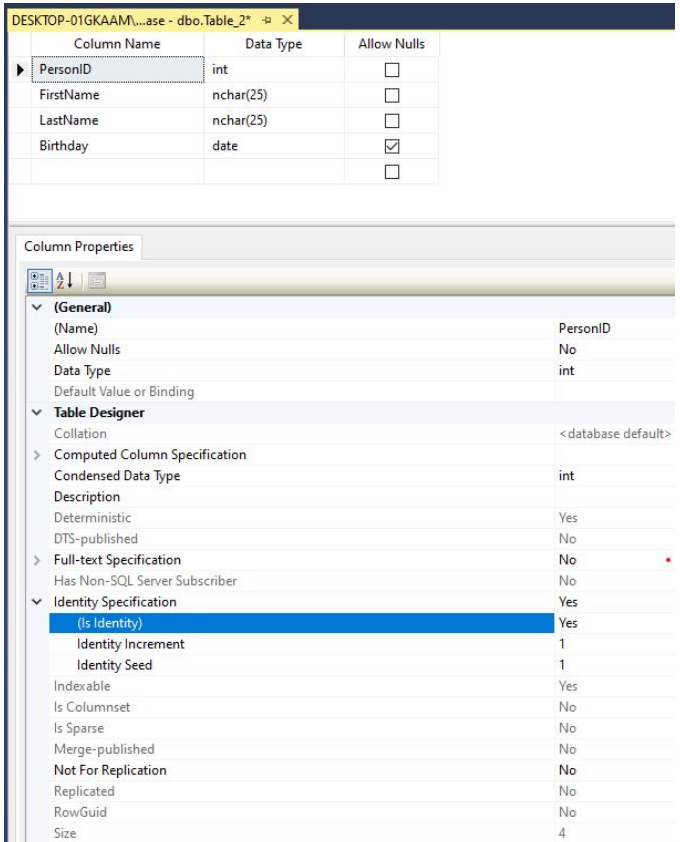
SSMS: Create Database

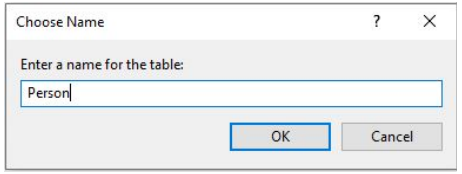




Create Table

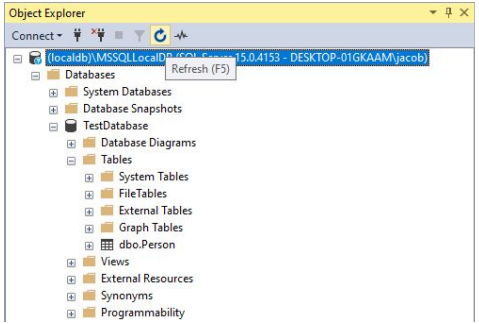






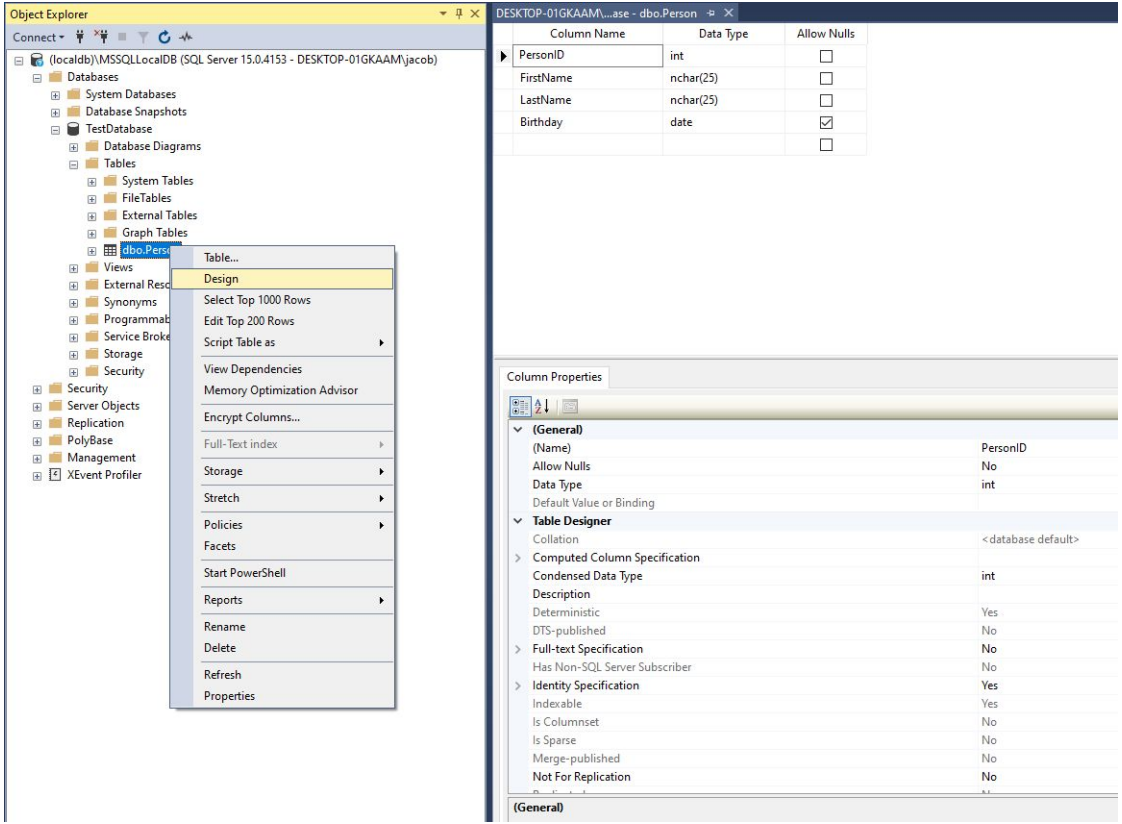
Once you finish designing your table hit "Save" or "ctrl + s" and give it a name.

Refresh Object Explorer and you should see your new table "dbo.Person" under "Tables"



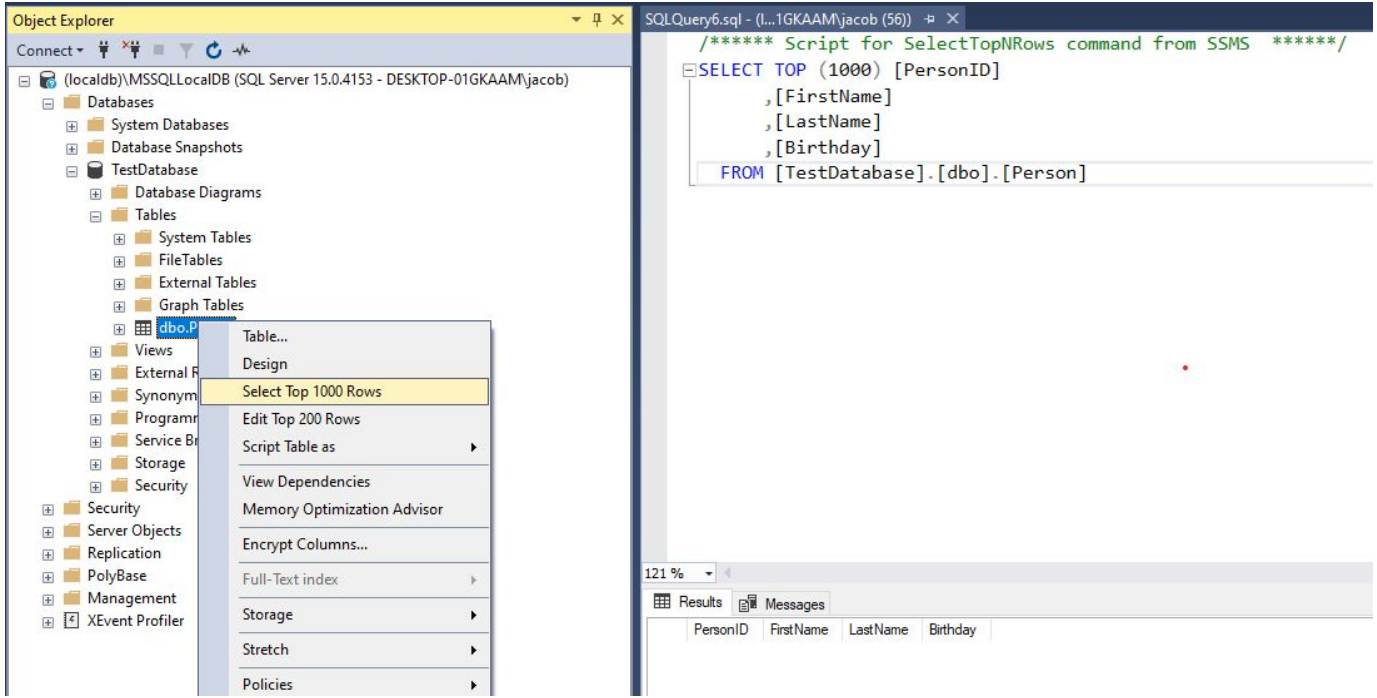
SSMS: Design table

- Add columns
- Modify Data type
- Add Constraints and properties



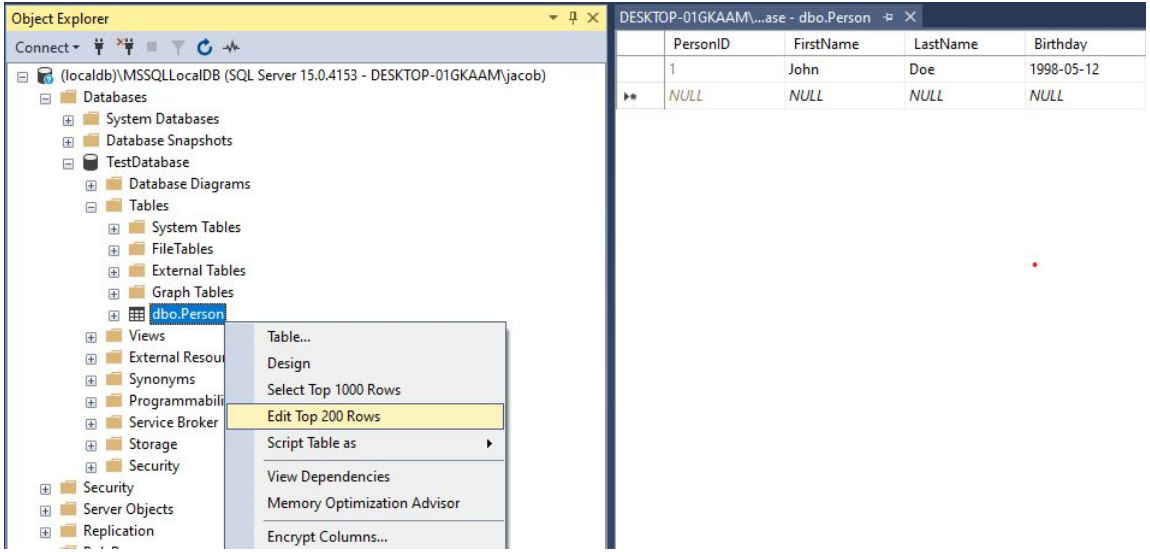
SSMS: Select Top 1000 Rows

- Generates script
- Result-set of 1000 records
- No records currently in empty table



SSMS: Edit Top 200 Rows

- Displays top 200 rows/records in table
- Edit values in designer
- Hit enter to commit value



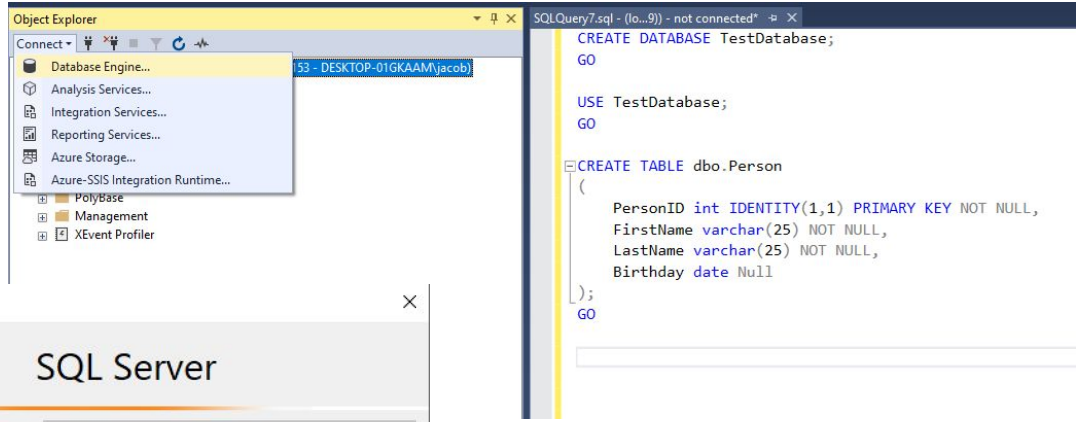
Code Along

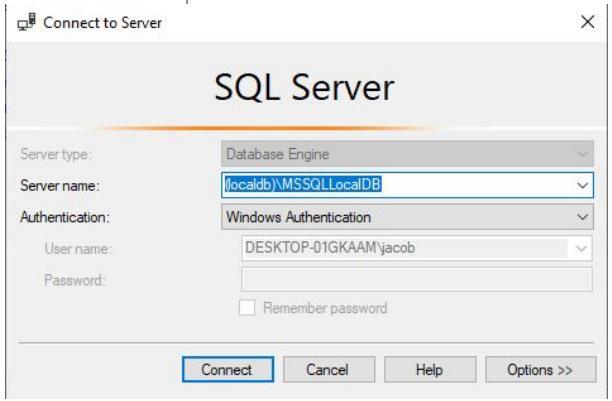
Let’s repeat these steps together…

… then we will try to accomplish the same thing using T-SQL!

SSMS: Create DB and Table using SQL Query

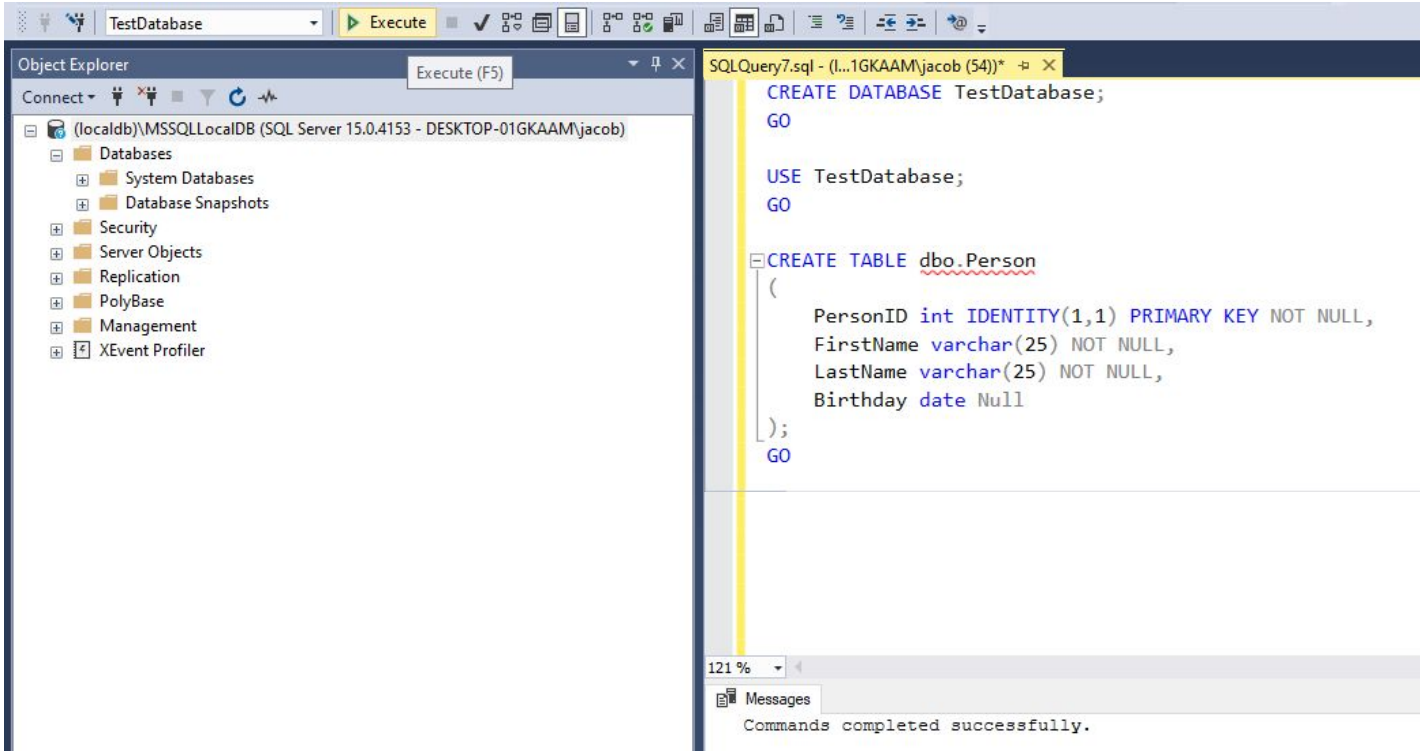
You can accomplish all the steps taken using the designer tools using a simple query





SSMS: New Query (ctrl + n)

- Connect to Database
- Create New query
- Separate statement/batch using ; and GO
- F5 to run
- Select/highlight part of your query to execute only that statement



Further Reading

- Quickstart: Connect and query a SQL Server instance using SQL Server Management Studio (SSMS)
- Tips and tricks for using SQL Server Management Studio (SSMS)