CREATING OBJECTS

OBJECT TYPES

- Tables
- Views
- Functions
- User Defined Types
- Stored Procedures (to be covered in depth later)
- Many more…

TABLES

- Permanent

  CREATE TABLE TableName(Col1 INT, Col2 INT…);

- Temp table: Table destroyed when you close the session

  CREATE TABLE #TableName(Col1 INT, Col2 INT…);

  CREATE TABLE ##TableName(Col1 INT, Col2 INT…);

- Table variable: Scoping rules, table dies after code runs

  DECLARE @TableName TABLE (Col1 INT, Col2 INT…);

TABLES: AUTOPOPULATING COLUMNS

- Identity, default, computed

```
CREATE TABLE TableName(
    Col1 INT NOT NULL IDENTITY,
    Col3 DATETIME2 DEFAULT GETDATE(),
    Col3 INT DEFAULT 10,
    Col4 AS Col1 * Col3)
```

TABLES: CONSTRAINTS

- Constraints: Nullability, Primary Key, Unique, Range of values

```
CREATE TABLE TableName(
        Col1 INT NOT NULL PRIMARY KEY,
        Col2 INT UNIQUE,
        Col3 CHAR(3),
        CONSTRAINT ch_TableName_col2 CHECK(Col2 < 10000)
);
ALTER TABLE TableName ADD CONSTRAINT ch_TableName_Col3
        CHECK (Col3 IN ('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'));
```

TABLES: FOREIGN KEYS (ALSO A CONSTRAINT!)

```
CREATE TABLE NewTable (
        ColA INT
        ColB INT,
        CONSTRAINT fk_NewTable_TableName
        FOREIGN KEY (ColB) REFERENCES TableName(Col1);
```

VIEWS

- A stored query – not the data
- Can often be used in place of a table
- Makes queries easier to write when lots of joins
- Can lead to bad performance as views are overused and nested
- Don't use to insert or update data
- Don't use ORDER BY unless needing TOP
  Often used for security reasons

VIEWS: DEFINITION

```
CREATE VIEW dbo.vw_NewView AS
        SELECT Col1, Col2 AS NewColumn, ColB
        FROM NewTable AS NT
        INNER JOIN TableName AS TN ON NT.ColB = TN.Col1;
```

USER DEFINED FUNCTIONS

- Three types:
  - Scalar, returns a value
  - Inline Table-Valued (like a filtered view), returns a record set
  - Multi-statement Table-Valued, returns a record set
- Can contain logic like IF and WHILE loops
- Can often be the reason for poor performance
- Can use default values, but must use the word “default” when calling
- See demos for syntax and examples

USER DEFINED TYPES

- Use to standardize a certain data type throughout the database
- Not used frequently, but there are examples in AdventureWorks

```
CREATE TYPE [dbo].[Address] FROM [nvarchar](150) NULL;

CREATE TABLE dbo.CustomerAddress(
        AddressID INT NOT NULL IDENTITY PRIMARY KEY,
        CustomerID NOT NULL,
        AddressLine1 Address,
        AddressLine2 Address,
        City NVARCHAR(30))
```

STORED PROCEDURE

- A stored script with the ability to do just about anything
- Can take parameters.
- You can skip parameters with defaults.
- The “workhorse” of T-SQL
- Can perform work, like updates and inserts or even create a new database
- Can return rows
- Cannot be used in a query
- Results can be saved to a pre-defined table
- Call with the EXEC keyword
- Can return a value, usually indicating success or failure

STORED PROCEDURE

- Will be covered in depth in a couple of weeks

```
CREATE PROC usp_NewProc (@Param1 INT, @Param2 INT = 7,
    @Param3 VARCHAR(10) = NULL AS

    <STATEMENTS>
```

DROPPING OBJECTS

- New IF EXISTS keyword
  - DROP TABLE IF EXISTS MyTable
  - DROP FUNTION IF EXISTS dbo.udfMyFunction
- Old syntax looked at system objects

```
IF EXISTS(SELECT * FROM sys.objects WHERE Name = 'MyTable' AND TYPE = 'U')
DROP TABLE MyTable;

IF OBJECT_ID('MyTable') IS NOT NULL DROP TABLE MyTable;
```