9. WITH Statement

The WITH keyword is used to make definitions within the query, but not stored in the database schema.

Syntax

```
WITH temporary_name AS (Subquery)

SELECT Column1, Column2, ...
FROM temporary_name
WHERE Condition1
```

Table: Appointments

	Id	Date	DoctorId	PatientId
▶	1	2020-06-12	1	2
	2	2020-06-13	3	2
	3	2020-06-14	3	1
	4	2020-06-13	1	4
	5	2020-06-13	3	6
	6	2020-06-14	2	3
	7	2020-06-15	2	2
	8	2020-06-15	2	2

Table: Doctors

	Id	Name
▶	1	Williams
	2	Smith
	3	Clark
	4	Johnson

Table: Patients

	Id	Name
▶	1	Robert
	2	James
	3	David
	4	Michael
	5	Oliver
	6	Mary

Data Script:

```
CREATE TABLE Appointments(
    Id int NOT NULL,
    Date date NOT NULL,
    DoctorId int NOT NULL,
    PatientId int NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Doctors(
    Id int NOT NULL,
    Name varchar(50) NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Patients(
    Id int NOT NULL,
    Name varchar(50) NOT NULL,
    PRIMARY KEY (Id)
);

INSERT INTO Appointments(Id, Date, DoctorId, PatientId)
VALUES (1, '06/12/2020', 1, 2), (2, '06/13/2020', 3, 2),(3, '06/14/2020', 3, 1),
(4, '06/13/2020', 1, 4), (5, '06/13/2020', 3, 6), (6, '06/14/2020', 2, 3);

INSERT INTO Doctors(Id, Name)
VALUES (1, 'Williams'), (2, 'Smith'), (3, 'Clark'), (4, 'Johnson');

INSERT INTO Patients(Id, Name)
VALUES (1, 'Robert'), (2, 'James'), (3, 'David'), (4, 'Michael'), (5, 'Oliver'),
(6, 'Mary');
```

Script 1:

```
WITH A AS
(
	SELECT Appointments.Id, Appointments.Date, Patients.Name AS PatientName
	, DoctorId
	FROM Appointments
	INNER JOIN Patients ON Appointments.PatientId = Patients.Id
	INNER JOIN Doctors ON Appointments.DoctorId = Doctors.Id
)
SELECT * FROM A WHERE Date = '06/13/2020'
```

Script 2:

```
WITH A([Patient_Id], [Appointment Date], [Patient Name], [Doctor Id])  AS
(
	SELECT Appointments.Id, Appointments.Date, Patients.Name AS PatientName,
DoctorId
	FROM Appointments
	INNER JOIN Patients ON Appointments.PatientId = Patients.Id
	INNER JOIN Doctors ON Appointments.DoctorId = Doctors.Id
)

SELECT * FROM A WHERE [Appointment Date] = '06/13/2020'
```

Example:

1. Filter data returned from a subquery.
2. Insert into a table from a CSV file.

Script 1:

```
WITH employee AS (SELECT * FROM Employees)
SELECT * FROM employee WHERE ID < 5
```

Script 2:

```
BULK INSERT Employees
FROM 'E:EmployeeList.csv'
WITH ( FORMAT='CSV');
```

Script 3:

```
WITH N AS (SELECT 3 AS Number UNION ALL SELECT 1 AS Number UNION ALL SELECT 4 AS
Number UNION ALL SELECT 1 AS Number)
SELECT * FROM N ORDER BY Number
```

Run this script and check the result.

SQL Script:

```
WITH NewTable1 AS (SELECT 3 AS Number UNION ALL SELECT 1 AS Number UNION ALL SELECT
4 AS Number)

, NewTable2 AS (SELECT Number * 10 AS Number FROM NewTable1)

SELECT * FROM NewTable2
```

SQL Script:

```
WITH NewTable1 AS (SELECT 3 AS Number UNION ALL SELECT 1 AS Number UNION ALL SELECT
4 AS Number)

, NewTable2 AS (SELECT Number * 10 AS Number FROM NewTable1)

, NewTable3 AS (SELECT Number + 1000 AS Number FROM NewTable2)

, NewTable4 AS (SELECT Number / 2 AS Number FROM NewTable3)

SELECT * FROM NewTable4 ORDER BY Number
```

Import data from CSV Source

```
BULK INSERT Articles
FROM 'E:Articles.CSV'
WITH ( FORMAT='CSV');
```

Import data from TEXT Source

```
BULK INSERT Articles
FROM 'E:Articles.TXT'
WITH ( FORMAT='CSV');
```

Example 1 - Declared table having one column

Table1 is the name of the declared table.
Column1_Lat is the column name.

Script 1:

```
DECLARE @MinLatArea1 FLOAT = 0.01251;
DECLARE @MaxLatArea1 FLOAT = 0.05251;
```

```
WITH Table1(Column1_Lat) AS
(
	SELECT ROUND(@MinLatArea1, 3) UNION ALL SELECT ROUND(@MaxLatArea1, 3)
)

SELECT * FROM Table1
```

Output 1:

Column1_Lat
0.013
0.053

Script 2:

```
DECLARE @MinLatArea1 FLOAT = 0.01251;
DECLARE @MaxLatArea1 FLOAT = 0.05251;

WITH SEC(Number) AS
(
	SELECT ROUND(@MinLatArea1, 3) UNION ALL SELECT ROUND(@MaxLatArea1, 3)
)

SELECT * FROM SEC
```

Output 2:

Number
0.013
0.053

Example 2 - Declared table having two columns

Table1 is the name of the declared table.
Column1_Lat and Column2_Lng are columns.

Script:

```
DECLARE @MinLatArea1 FLOAT = 0.01251;
DECLARE @MaxLatArea1 FLOAT = 0.05251;
```

```
DECLARE @MinLngArea1 FLOAT = 0.04151;
DECLARE @MaxLngArea1 FLOAT = 0.08151;


WITH Table1(Column1_Lat, Column2_Lng) AS
(
	SELECT ROUND(@MinLatArea1, 3), ROUND(@MinLngArea1, 3)

	UNION ALL

	SELECT ROUND(@MaxLatArea1, 3), ROUND(@MaxLngArea1, 3)
)

SELECT * FROM Table1
```

Output:

Column1_Lat	Column2_Lng
0.013	.052
0.042	.082

COALESCE

The COALESCE will return the first not nullable value in a list.

Example:
Selecting the first not nullable value in a list.

Script:

```
SELECT COALESCE( NULL, 2, 5, 10, NULL, 4) AS [Value];
SELECT COALESCE( 2, 5, 10, NULL, 4) AS [Value];
SELECT COALESCE(  NULL,  NULL, 2, 5, 10, NULL, 4) AS [Value];
```