# WHILE loops

INTERMEDIATE SQL SERVER

**Ginger Grant**
Instructor

# Using variables in T-SQL

- Variables are needed to set values `DECLARE @variablename data_type`
  - Must start with the character @

# Variable data types in T-SQL

- `VARCHAR(n)` : variable length text field
- `INT` : integer values from -2,147,483,647 to +2,147,483,647
- `DECIMAL(p ,s)` or `NUMERIC(p ,s)` :
  - `p` : total number of decimal digits that will be stored, both to the left and to the right of the decimal point
  - `s` : number of decimal digits that will be stored to the right of the decimal point

# Declaring variables in T-SQL

```
-- Declare Snack as a VARCHAR with length 10
DECLARE @Snack VARCHAR(10)
```

# Assigning values to variables

```
-- Declare the variable
DECLARE @Snack VARCHAR(10)
-- Use SET a value to the variable
SET @Snack = 'Cookies'
-- Show the value
SELECT @Snack
```

```
+--------------------+
|(No column name)    |
+--------------------+
|Cookies             |
+--------------------+
```

```
-- Declare the variable
DECLARE @Snack VARCHAR(10)
-- Use SELECT assign a value
SELECT @Snack = 'Candy'
-- Show the value
SELECT @Snack
```

```
+--------------------+
|(No column name)    |
+--------------------+
|Candy               |
+--------------------+
```

# WHILE loops

- WHILE evaluates a true or false condition
- After the WHILE, there should be a line with the keyword BEGIN
- Next include code to run until the condition in the WHILE loop is true
- After the code add the keyword END
- BREAK will cause an exit out of the loop
- CONTINUE will cause the loop to continue

# WHILE loop in T-SQL (I)

```
-- Declare ctr as an integer
DECLARE @ctr INT
-- Assign 1 to ctr
SET @ctr = 1
-- Specify the condition of the WHILE loop
WHILE @ctr < 10
    -- Begin the code to execute inside WHILE loop
    BEGIN
       -- Keep incrementing the value of @ctr
       SET @ctr = @ctr + 1
       -- End WHILE loop
    END
-- View the value after the loop
SELECT @ctr
```

```
+--------------------+
|(No column name)    |
+--------------------+
|10                  |
+--------------------+
```

# WHILE loop in T-SQL (II)

```
-- Declare ctr as an integer
DECLARE @ctr INT
-- Assign 1 to ctr
SET @ctr = 1
-- Specify the condition of the WHILE loop
WHILE @ctr < 10
    -- Begin the code to execute inside WHILE loop
    BEGIN
       -- Keep incrementing the value of @ctr
       SET @ctr = @ctr + 1
```

```
       -- Check if ctr is equal to 4
       IF @ctr = 4
           -- When ctr is equal to 4, the loop will break
           BREAK
       -- End WHILE loop
    END
```

# Let's practice!

INTERMEDIATE SQL SERVER

# Derived tables

INTERMEDIATE SQL SERVER

**Ginger Grant**
Instructor

# What are Derived tables?

- Query which is treated like a temporary table
- Always contained within the main query
- They are specified in the `FROM` clause
- Can contain intermediate calculations to be used the main query or different joins than in the main query

# Derived tables in T-SQL

```
SELECT a.* FROM Kidney a
-- This derived table computes the Average age joined to the actual table
JOIN (SELECT AVG(Age) AS AverageAge
      FROM Kidney) b
ON a.Age = b.AverageAge
```

# Let's practice!

INTERMEDIATE SQL SERVER

# Common Table Expressions

INTERMEDIATE SQL SERVER

**Ginger Grant**
Instructor

# CTE syntax

```
-- CTE definitions start with the keyword WITH
-- Followed by the CTE names and the columns it contains
WITH CTEName (Col1, Col2)
AS
-- Define the CTE query
(
-- The two columns from the definition above
    SELECT Col1, Col2
    FROM TableName
)
```

# CTEs in T-SQL

```
-- Create a CTE to get the Maximum BloodPressure by Age
WITH BloodPressureAge(Age, MaxBloodPressure)
AS
(SELECT Age, MAX(BloodPressure) AS MaxBloodPressure
 FROM Kidney
 GROUP BY Age)

-- Create a query to use the CTE as a table
SELECT a.Age, MIN(a.BloodPressure), b.MaxBloodPressure
FROM Kidney a
-- Join the CTE with the table
JOIN BloodpressureAge b
     ON a.Age = b.Age
GROUP BY a.Age, b.MaxBloodPressure
```

# Let's practice!

INTERMEDIATE SQL SERVER