**Introduction to T-SQL (Transact-SQL)**

T-SQL (Transact-SQL) is an extension of SQL (Structured Query Language) used in Microsoft SQL Server. It adds procedural programming capabilities like variables, loops, conditions, and error handling, making it more powerful for database operations.

T-SQL is used to:  
✅ Write complex queries for data retrieval  
✅ Perform procedural logic using loops and conditions  
✅ Create stored procedures, functions, and triggers

**Topics in T-SQL (Part 2 of SQL Tutorial)**

1️⃣ **Variables and Data Types**

* Declaring and assigning variables (DECLARE, SET, SELECT)
* Common data types (INT, VARCHAR, DATETIME, etc.)

2️⃣ **Control Flow Statements**

* IF...ELSE (Decision-making)
* WHILE (Loops)
* CASE (Conditional expressions)

3️⃣ **Stored Procedures**

* Creating and executing stored procedures (CREATE PROCEDURE)
* Passing parameters to stored procedures

4️⃣ **User-Defined Functions (UDFs)**

* Scalar functions (returning a single value)
* Table-valued functions (returning a table)

5️⃣ **Triggers**

* Automatic execution on INSERT, UPDATE, or DELETE operations
* AFTER and INSTEAD OF triggers

**1️⃣ Variables and Data Types in T-SQL**

**What is a Variable in T-SQL?**

A **variable** in T-SQL is a temporary storage location that holds a value during the execution of a query. It helps in storing data that can be used in different parts of a query.

**Declaring a Variable**

In T-SQL, you must declare a variable before using it. Use the DECLARE statement.

📌 **Syntax:**

DECLARE @variable\_name DATATYPE;

📌 **Example:**

DECLARE @StudentName VARCHAR(50);

DECLARE @TotalMarks INT;

DECLARE @DOB DATE;

**Assigning Values to Variables**

After declaring a variable, you can assign a value to it using either SET or SELECT.

📌 **Using SET:**

SET @StudentName = 'John Doe';

SET @TotalMarks = 85;

SET @DOB = '2001-06-15';

📌 **Using SELECT:**

SELECT @StudentName = 'Jane Smith';

SELECT @TotalMarks = 90;

SELECT @DOB = '2000-10-20';

**Difference:**

* SET is used to assign only **one value at a time**.
* SELECT can assign **multiple values at once** and can take values from a table.

📌 **Example using a Table Value:**

DECLARE @MaxSalary INT;

SELECT @MaxSalary = MAX(Salary) FROM Employees;

PRINT 'The highest salary is ' + CAST(@MaxSalary AS VARCHAR(10));

**Displaying Variable Values**

📌 **Using PRINT:**

PRINT 'Student Name: ' + @StudentName;

PRINT 'Total Marks: ' + CAST(@TotalMarks AS VARCHAR(10));

PRINT 'Date of Birth: ' + CAST(@DOB AS VARCHAR(20));

📌 **Using SELECT:**

SELECT @StudentName AS Student, @TotalMarks AS Marks, @DOB AS BirthDate;

**Common Data Types in T-SQL**

| **Data Type** | **Description** | **Example** |
| --- | --- | --- |
| INT | Stores whole numbers | 100, -50 |
| VARCHAR(n) | Stores text (variable length) | 'John', 'SQL123' |
| CHAR(n) | Stores text (fixed length) | 'A', 'Yes' |
| DATE | Stores date | '2025-03-19' |
| DATETIME | Stores date & time | '2025-03-19 14:30:00' |
| BIT | Boolean (0 or 1) | 1 (true), 0 (false) |
| DECIMAL(p,s) | Stores decimal numbers | 99.99, 123.45 |

**Example: Working with Variables**

DECLARE @EmpName VARCHAR(50);

DECLARE @EmpSalary DECIMAL(10,2);

DECLARE @JoiningDate DATE;

SET @EmpName = 'Alice';

SET @EmpSalary = 55000.75;

SET @JoiningDate = '2022-01-15';

PRINT 'Employee Name: ' + @EmpName;

PRINT 'Salary: ' + CAST(@EmpSalary AS VARCHAR(10));

PRINT 'Joining Date: ' + CAST(@JoiningDate AS VARCHAR(20));

✔ **Output:**

Employee Name: Alice

Salary: 55000.75

Joining Date: 2022-01-15

**Why Do We Use CAST in T-SQL?**

In T-SQL, CAST is used to **convert** one data type into another.

**Why is it needed?**

1️⃣ **PRINT only works with strings** → If we try to print a number or date directly, it will throw an error.  
2️⃣ **Different data types can't be combined** → For example, we cannot concatenate a string (VARCHAR) with an integer (INT) directly.

**Example Without CAST (Error)**

DECLARE @TotalMarks INT = 85;

PRINT 'Total Marks: ' + @TotalMarks;

🚨 **Error:**

Operand data type int is invalid for + operator.

**Example With CAST (Correct)**

DECLARE @TotalMarks INT = 85;

PRINT 'Total Marks: ' + CAST(@TotalMarks AS VARCHAR(10));

✔ **Output:**

Total Marks: 85

**Other Ways to Convert Data Types**

1️⃣ **Using CONVERT()**

PRINT 'Marks: ' + CONVERT(VARCHAR(10), @TotalMarks);

2️⃣ **Using FORMAT() (For Dates)**

DECLARE @DOB DATE = '2001-06-15';

PRINT 'Date of Birth: ' + FORMAT(@DOB, 'yyyy-MM-dd');

✔ **Output:**

Date of Birth: 2001-06-15

**✅ Exercise: Working with Variables & CAST**

Write a T-SQL script to:

1. Declare variables for a student's **Name**, **Age**, and **Exam Date**.
2. Assign values to these variables.
3. Print them using CAST.

**2️⃣ Control Flow Statements in T-SQL**

**What are Control Flow Statements?**

Control flow statements allow us to control the execution of SQL statements based on conditions, loops, or decision-making logic.

📌 **T-SQL Control Flow Statements Include:**

1. IF...ELSE → Decision-making
2. CASE → Conditional expressions inside SELECT
3. WHILE → Loop execution

**1️⃣ IF...ELSE (Decision Making)**

IF...ELSE is used to execute SQL statements based on a condition.

📌 **Syntax:**

IF condition

BEGIN

-- SQL statements if condition is TRUE

END

ELSE

BEGIN

-- SQL statements if condition is FALSE

END

📌 **Example: Checking Student Result**

DECLARE @Marks INT = 85;

IF @Marks >= 50

PRINT 'Pass';

ELSE

PRINT 'Fail';

✔ **Output:**

Pass

**2️⃣ CASE (Conditional Expressions)**

CASE is used inside SELECT to return different values based on conditions.

📌 **Syntax:**

SELECT

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

ELSE default\_result

END AS ColumnName

FROM TableName;

📌 **Example: Grading System**

DECLARE @Marks INT = 85;

SELECT

CASE

WHEN @Marks >= 90 THEN 'A+'

WHEN @Marks >= 80 THEN 'A'

WHEN @Marks >= 70 THEN 'B'

WHEN @Marks >= 60 THEN 'C'

ELSE 'Fail'

END AS Grade;

✔ **Output:**

Grade

-----

A

📌 **Example with a Table (Employee Bonus Calculation)**

SELECT EmployeeName, Salary,

CASE

WHEN Salary > 50000 THEN 'High Bonus'

WHEN Salary BETWEEN 30000 AND 50000 THEN 'Medium Bonus'

ELSE 'Low Bonus'

END AS BonusCategory

FROM Employees;

**3️⃣ WHILE (Looping)**

The WHILE loop is used to execute SQL statements **repeatedly** until a condition becomes FALSE.

📌 **Syntax:**

WHILE condition

BEGIN

-- SQL statements

END

📌 **Example: Printing Numbers from 1 to 5**

DECLARE @Counter INT = 1;

WHILE @Counter <= 5

BEGIN

PRINT 'Number: ' + CAST(@Counter AS VARCHAR(10));

SET @Counter = @Counter + 1;

END

✔ **Output:**

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

📌 **Example: Inserting Multiple Records Using WHILE**

DECLARE @i INT = 1;

WHILE @i <= 5

BEGIN

INSERT INTO Students (StudentName, Marks)

VALUES ('Student ' + CAST(@i AS VARCHAR(10)), 50 + @i);

SET @i = @i + 1;

END

**⏳ Combining IF, CASE, and WHILE**

📌 **Example: Using IF and WHILE Together**

DECLARE @Num INT = 1;

WHILE @Num <= 5

BEGIN

IF @Num % 2 = 0

PRINT CAST(@Num AS VARCHAR(10)) + ' is Even';

ELSE

PRINT CAST(@Num AS VARCHAR(10)) + ' is Odd';

SET @Num = @Num + 1;

END

✔ **Output:**

1 is Odd

2 is Even

3 is Odd

4 is Even

5 is Odd

**✅ Exercise: Control Flow in T-SQL**

Try writing a script that:

1. **Takes a student’s marks** as a variable.
2. **Uses IF...ELSE** to determine if the student passed or failed.
3. **Uses CASE** to assign grades (A, B, C, Fail).
4. **Uses WHILE** to print numbers from 10 to 15.