# 1) What is the Derived Column Transformation?

Derived Column is a synchronous Data Flow transformation that lets you compute new column values or replace existing ones using the SSIS expression language. Typical uses: concatenation, conditional logic (IF/ELSE), substringing, case changes, math on numbers, defaulting NULLs, date arithmetic, and quick data cleanup.

# 2) When to Use It

* You need to add new attributes (e.g., FullName = FirstName + ' ' + LastName).
* You need to standardize values (e.g., Gender → 'M'/'F', UPPER(Company)).
* You want to replace a column in-stream without staging back to SQL first.
* Fast, in-memory transformation—no I/O cost beyond the flow itself.

# 3) Scenario for the Demo

Source: CSV with columns ID, first\_name, last\_name, gender, company\_name.  
Destination: SQL Server table. We will add four columns in-flight:  
• FullName = first\_name + ' ' + last\_name  
• Gender1 = first letter from gender using SUBSTRING (M/F)  
• Gender2 = conditional M/F using SSIS ? : expression  
• Company2 = UPPER(company\_name)

# 4) Step‑by‑Step (Visual Studio 2022 / SSIS)

1. Create an Integration Services Project and add a Data Flow Task.
2. Inside Data Flow: add Flat File Source → New connection → point to your CSV, tick Column names in the first data row, preview to confirm columns.
3. Add Derived Column and connect Flat File Source → Derived Column.
4. Open Derived Column and create these outputs (choose Add as new column for each):

```  
-- 1) FullName (handles NULLs and trims; casts to Unicode to avoid type mix)  
FullName = REPLACENULL(TRIM((DT\_WSTR,50)first\_name), "")  
 + " " + REPLACENULL(TRIM((DT\_WSTR,50)last\_name), "")  
  
-- 2) Gender1 via SUBSTRING (first character)  
Gender1 = UPPER(SUBSTRING((DT\_WSTR,10)gender, 1, 1))  
  
-- 3) Gender2 via IF/ELSE (?:) on full value  
Gender2 = UPPER((DT\_WSTR,10)gender) == "MALE" ? "M"  
 : UPPER((DT\_WSTR,10)gender) == "FEMALE" ? "F"  
 : NULL(DT\_WSTR,1)  
  
-- 4) Company2 uppercase  
Company2 = UPPER((DT\_WSTR,100)company\_name)  
```

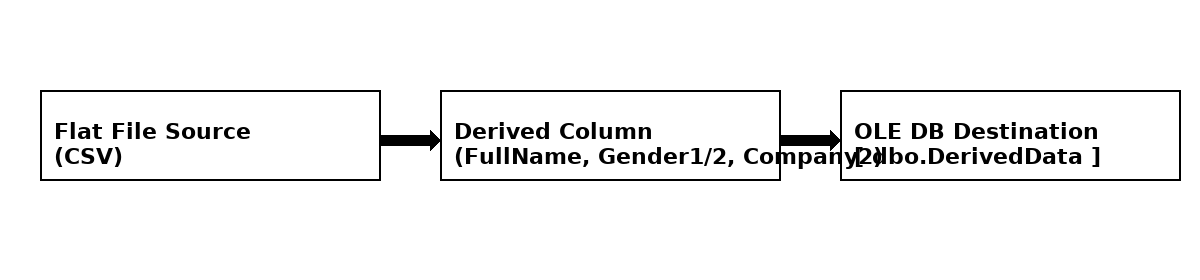
Notes:  
• Flat File text columns are usually DT\_STR (non‑Unicode). Casting to DT\_WSTR in expressions avoids mixed‑type errors when using string literals.  
• REPLACENULL(x, "") prevents concatenation from turning NULL into NULL.  
• SUBSTRING uses 1‑based start index in SSIS expressions.

1. Add OLE DB Destination → connect to SQL Server → create or choose destination table.
2. In Mappings, map existing source columns and the new derived columns (FullName, Gender1, Gender2, Company2).
3. Run the package and validate results in the destination.

# 5) Destination Table (example DDL)

```sql  
CREATE TABLE dbo.DerivedData (  
 ID INT NULL,  
 first\_name VARCHAR(50) NULL,  
 last\_name VARCHAR(50) NULL,  
 gender VARCHAR(10) NULL,  
 company\_name VARCHAR(100) NULL,  
 FullName NVARCHAR(101) NULL, -- 'First Last' (supports Unicode)  
 Gender1 NVARCHAR(1) NULL, -- 'M' or 'F' (from first letter)  
 Gender2 NVARCHAR(1) NULL, -- 'M'/'F' via conditional  
 Company2 NVARCHAR(100) NULL -- uppercased company  
);  
```  
Tip: If your destination columns are VARCHAR, you can either keep derived outputs in Unicode (NVARCHAR) and let OLE DB convert, or append a Data Conversion transform to cast DT\_WSTR → DT\_STR with CodePage 1252.

## Pipeline at a Glance



# 6) Add as New Column vs Replace Existing

* In the Derived Column editor, the drop‑down lets you Add as new column (keeps original) or Replace an existing column (saves memory and simplifies mappings).
* Replacing is great for standardizations like gender = UPPER(SUBSTRING(gender,1,1)), but keep a copy if downstream auditing needs the original value.

# 7) Common Expression Patterns (Cheat‑Sheet)

```  
-- Safe concatenation with defaults  
REPLACENULL(TRIM((DT\_WSTR,50)col1), "") + ", " + REPLACENULL(TRIM((DT\_WSTR,50)col2), "")  
  
-- Title Case (simple)  
UPPER(SUBSTRING((DT\_WSTR,100)name,1,1)) + LOWER(SUBSTRING((DT\_WSTR,100)name,2,LEN((DT\_WSTR,100)name)-1))  
  
-- Numeric: guard against divide-by-zero  
(qty == 0 ? NULL(DT\_R8) : sales / (DT\_R8)qty)  
  
-- Dates  
DATEADD("day", 7, GETDATE()) -- plus 7 days  
DATEDIFF("day", order\_date, GETDATE()) -- age in days  
  
-- Conditional buckets  
sales > 10000 ? "High" : (sales > 5000 ? "Medium" : "Low")  
```

# 8) Troubleshooting & Best Practices

* Type mismatch (DT\_STR vs DT\_WSTR) → cast inputs to a common type. For strings in expressions with literals, use DT\_WSTR.
* NULLs break concatenation → use REPLACENULL or (ISNULL(col)?"":col).
* Truncation → increase output length in destination or cast explicitly, e.g., (DT\_WSTR,120) before concatenation.
* Performance → Derived Column is very fast; keep expressions readable and avoid redundant casts.
* Pair with Data Conversion when you must output non‑Unicode DT\_STR to a VARCHAR destination.
* Keep expression snippets in annotations or a shared doc for consistent reuse.

# 9) Verify in SQL

```sql  
SELECT TOP 10 ID, first\_name, last\_name, gender, company\_name,  
 FullName, Gender1, Gender2, Company2  
FROM dbo.DerivedData;  
```