

# ***Data Loading with Incremental Processing***

## **Project Assignment - 04**

Aurabrato Ghosh

Azure Data Engineering

### **Table of Contents**

Project Overview.....	2
Source Table Creation.....	2
Watermark Table Creation .....	3
Target Tables Creation .....	4
Stored Procedure.....	4
Pipeline Design .....	6
Step-by-Step Pipeline Flow.....	7
How We Keep Track of New Data .....	8
What the Stored Procedure Does.....	9
What the Pipeline Does .....	9
Why This Is Useful.....	9
References .....	9

# Project Overview

This project demonstrates incremental data loading using SQL-based watermarking logic and dynamic pipeline execution in Azure Synapse. The goal is to avoid reprocessing already loaded data by only processing new records during each pipeline run.

## Source Table Creation

We created five source tables — two using INT columns and three using DATETIME columns for incremental tracking. Each table was populated with sample data to simulate real-world updates.

```
CREATE TABLE SalesOrders (  
    OrderID INT PRIMARY KEY,  
    CustomerName VARCHAR(100),  
    Amount DECIMAL(10, 2)  
);  
  
INSERT INTO SalesOrders (OrderID, CustomerName, Amount) VALUES  
(1, 'Alice', 250.00),  
(2, 'Bob', 180.50),  
(3, 'Charlie', 320.00);
```

### SQL Query to Create SalesOrder table and insert sample data

```
CREATE TABLE ProductInventory (  
    ProductID INT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    StockQuantity INT  
);  
  
INSERT INTO ProductInventory (ProductID, ProductName, StockQuantity) VALUES  
(101, 'Widget A', 50),  
(102, 'Widget B', 30),  
(103, 'Widget C', 20);
```

### SQL Query to Create ProductInventory table and insert sample data

```
CREATE TABLE CustomerUpdates (  
    CustomerID INT PRIMARY KEY,  
    CustomerName VARCHAR(100),  
    LastUpdate DATETIME  
);  
  
INSERT INTO CustomerUpdates (CustomerID, CustomerName, LastUpdate) VALUES  
(201, 'Alice', '2024-06-01 10:00:00'),  
(202, 'Bob', '2024-06-02 15:30:00'),  
(203, 'Charlie', '2024-06-03 09:45:00');
```

### SQL Query to Create CustomerUpdates table and insert sample data

```

CREATE TABLE EmployeeChanges (
    EmployeeID INT PRIMARY KEY,
    EmployeeName VARCHAR(100),
    ChangeTimestamp DATETIME
);

INSERT INTO EmployeeChanges (EmployeeID, EmployeeName, ChangeTimestamp) VALUES
(301, 'David', '2024-06-05 12:00:00'),
(302, 'Eva', '2024-06-06 08:20:00'),
(303, 'Frank', '2024-06-07 14:10:00');

```

SQL Query to Create EmployeeChanges table and insert sample data

```

CREATE TABLE ShipmentLogs (
    ShipmentID INT PRIMARY KEY,
    Destination VARCHAR(100),
    ShippedAt DATETIME
);

INSERT INTO ShipmentLogs (ShipmentID, Destination, ShippedAt) VALUES
(401, 'Toronto', '2024-06-08 11:00:00'),
(402, 'Montreal', '2024-06-09 13:45:00'),
(403, 'Vancouver', '2024-06-10 16:30:00');

```

SQL Query to Create ShipmentLogs table and insert sample data

## Watermark Table Creation

We created a central table called WatermarkTracking that holds the last processed value for each table.

```

CREATE TABLE WatermarkTracking (
    TableName VARCHAR(100) PRIMARY KEY,
    LastProcessedValue VARCHAR(100), -- stores INT or DATETIME as string
    DataType VARCHAR(20)            -- 'INT' or 'DATETIME'
);

-- Initial entries (set to 0 or a very early date)
INSERT INTO WatermarkTracking (TableName, LastProcessedValue, DataType) VALUES
('SalesOrders', '0', 'INT'),
('ProductInventory', '0', 'INT'),
('CustomerUpdates', '1900-01-01 00:00:00', 'DATETIME'),
('EmployeeChanges', '1900-01-01 00:00:00', 'DATETIME'),
('ShipmentLogs', '1900-01-01 00:00:00', 'DATETIME');

```

SQL Query to Create WatermarkTracking table and insert initial values

This table ensures that each execution only processes record newer than the last run.

# Target Tables Creation

For each source table, a corresponding \_Target table was created to store the incrementally loaded data.

```
CREATE TABLE SalesOrders_Target (  
    OrderID INT PRIMARY KEY,  
    CustomerName VARCHAR(100),  
    Amount DECIMAL(10, 2)  
);  
  
CREATE TABLE ProductInventory_Target (  
    ProductID INT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    StockQuantity INT  
);  
  
CREATE TABLE CustomerUpdates_Target (  
    CustomerID INT PRIMARY KEY,  
    CustomerName VARCHAR(100),  
    LastUpdate DATETIME  
);  
  
CREATE TABLE EmployeeChanges_Target (  
    EmployeeID INT PRIMARY KEY,  
    EmployeeName VARCHAR(100),  
    ChangeTimestamp DATETIME  
);  
  
CREATE TABLE ShipmentLogs_Target (  
    ShipmentID INT PRIMARY KEY,  
    Destination VARCHAR(100),  
    ShippedAt DATETIME  
);
```

SQL Query to Create the Target tables for each Source table

## Stored Procedure

This stored procedure is used to load new data into target tables from the source tables. It only picks the records that were not loaded before, based on an ID or date column. It helps avoid loading the same data again and again.

It takes one input:

- @TableName: The name of the table you want to process.

## Execution Steps:

1. Check the last value that was loaded
  - It reads the WatermarkTracking table to find the last ID or timestamp it processed for that table.
2. Figure out which column to check
  - It uses OrderID for SalesOrders
  - It uses LastUpdate for CustomerUpdates
  - It uses ShippedAt for ShipmentLogs
  - It uses ChangeTimestamp for EmployeeChanges
  - It uses ProductID for ProductInventory
3. Write and run a query: It creates a SQL query that,
  - Picks only new records from the source table
  - Loads them into a matching \_Target table
  - Updates the WatermarkTracking table with the newest value
4. Run the SQL
  - The procedure uses sp\_executesql to run that SQL.

```
DROP PROCEDURE IF EXISTS usp_LoadIncrementalData;
GO

CREATE PROCEDURE usp_LoadIncrementalData
    @TableName VARCHAR(100)
AS
BEGIN
    DECLARE @LastValue VARCHAR(100)
    DECLARE @DataType VARCHAR(20)
    DECLARE @ColumnName VARCHAR(100)
    DECLARE @SQL NVARCHAR(MAX)

    SELECT
        @LastValue = LastProcessedValue,
        @DataType = DataType
    FROM WatermarkTracking
    WHERE TableName = @TableName

    IF @TableName = 'CustomerUpdates'
        SET @ColumnName = 'LastUpdate'
    ELSE IF @TableName = 'EmployeeChanges'
        SET @ColumnName = 'ChangeTimestamp'
    ELSE IF @TableName = 'ShipmentLogs'
        SET @ColumnName = 'ShippedAt'
    ELSE IF @TableName = 'SalesOrders'
        SET @ColumnName = 'OrderID'
    ELSE IF @TableName = 'ProductInventory'
        SET @ColumnName = 'ProductID'
```

```

SET @SQL = '
INSERT INTO ' + @TableName + '_Target
SELECT * FROM ' + @TableName + '
WHERE ' + @ColumnName + ' > ' +
    CASE
        WHEN @DataType = 'INT' THEN @LastValue
        ELSE '''' + @LastValue + ''''
    END + '

UPDATE WatermarkTracking
SET LastProcessedValue = (
    SELECT MAX(' + @ColumnName + ') FROM ' + @TableName + '
)
WHERE TableName = '''' + @TableName + ''''

EXEC sp_executesql @SQL
END

```

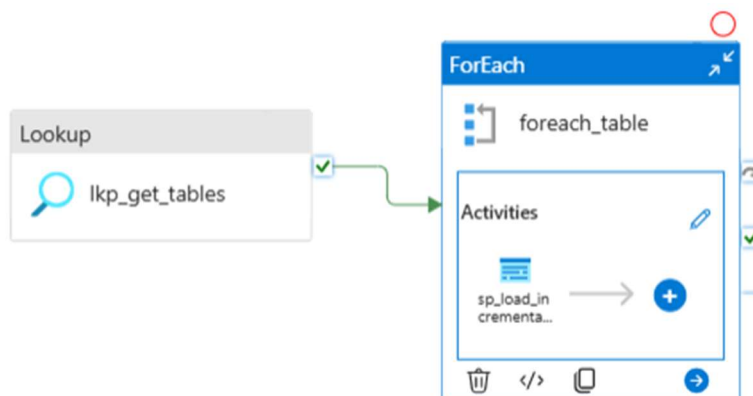
Stored Procedure to Load new data from Source to Target tables

## Pipeline Design

The pipeline automatically loads new data from each table, one by one. It runs a stored procedure that checks the last value loaded before (from the WatermarkTracking table) and copies only the new rows into the matching \_Target table.

### Activities Used in the Pipeline

- Lookup: Reads the list of all source tables from WatermarkTracking table
- For Each: Loops through each table one at a time
- Stored Procedure: Runs the stored procedure to load new records



Pipeline Structure

## Step-by-Step Pipeline Flow

### ❖ Lookup Activity



- Name: lkp\_get\_tables
- SQL Query: SELECT TableName FROM WatermarkTracking;
- Purpose: It fetches the names of all tables that should be processed.

### ❖ For Each Activity

- Name: foreach\_table
- Items: @activity('lkp\_get\_tables').output.value
- Setting: Sequential = true
- Purpose: Loops through each table returned from the Lookup.

### ❖ Stored Procedure Activity (inside For Each)

- Name: sp\_load\_incremental\_data
- Linked Service: Your Azure SQL DB
- Procedure Name: [dbo].[usp\_LoadIncrementalData]
- Parameter Mapping:
  - Parameter: @TableName
  - Value: @item().TableName
- Purpose: Calls the stored procedure and passes the current table name.

General	Settings	User properties
Source dataset *	<div>AzureSqlTable7</div>	 Open  New
First row only	<input type="checkbox"/>	
Use query	<input type="radio"/> Table <input checked="" type="radio"/> Query <input type="radio"/> Stored procedure	
Query	<div>SELECT TableName FROM Watermark...</div>	
Query timeout (minutes) ⓘ	<div>120</div>	
Isolation level ⓘ	<div>Select...</div>	
Partition option ⓘ	<input checked="" type="radio"/> None <input type="radio"/> Physical partitions of table ⓘ <input type="radio"/> Dynamic range ⓘ	

Lookup activity settings

General
Settings
Activities (1)
User properties

Sequential
☒

Items

## For Each activity settings

General
Settings
User properties

*ⓘ* To reference SQL pool, use the SQL pool stored procedure instead.

Linked service \* ⓘ

Integration runtime \*
☒ AutoResolveIntegrationRuntime

Stored procedure name \*

☐ Enter manually

Stored procedure parameters ⓘ

<input type="checkbox"/>	Name	Type *	Value
<input type="checkbox"/>	TableName	String	<input type="text" value="@item().TableName"/>

## Stored Procedure settings

## How We Keep Track of New Data

We use a table called WatermarkTracking to remember the last record we loaded from each table. It stores:

- The table name
- The last ID or date we loaded
- The column we are using to check for new data (like OrderID or LastUpdate)
- This way, we only load data that is newer than the last time



## What the Stored Procedure Does

The stored procedure:

- Look at the last value from the watermark table.
- Picks only rows that have a higher ID or newer date.
- Copies those rows into the \_Target table.
- Updates the watermark table to save the new last value.

So, each time it runs, it only adds new data.

## What the Pipeline Does

The Synapse pipeline automates the whole process:

- It gets the list of tables to process.
- It goes through each table one by one.
- For each table, it calls the stored procedure.

This means we don't need to write separate logic for every table — the pipeline handles all of them using a loop.

## Why This Is Useful

- It saves time by skipping already loaded data.
- It is easy to manage because everything is automated.
- We can add more tables later by just updating the watermark table.
- It is clean and simple, so it's easy to understand and update in the future.

## References

- Microsoft Learn Documentation – Azure Synapse Pipelines  
<https://learn.microsoft.com/en-us/azure/synapse-analytics/data-integration/concepts-pipelines-activities>
- Microsoft Learn – Incremental Data Loading Patterns  
<https://learn.microsoft.com/en-us/azure/data-factory/tutorial-incremental-copy-overview>
- Personal notes and practice environment using Azure Synapse and SQL Server (2025)