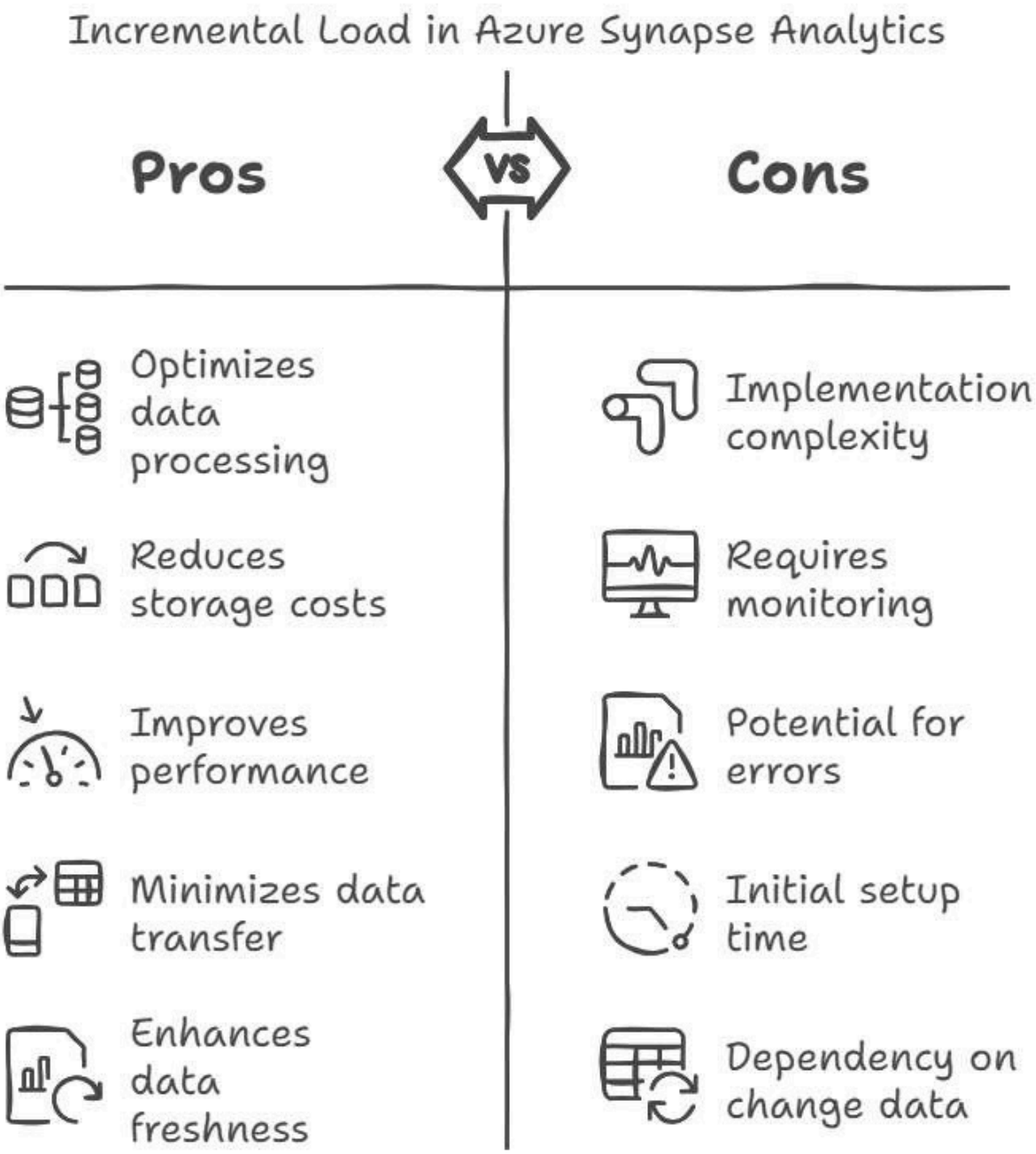
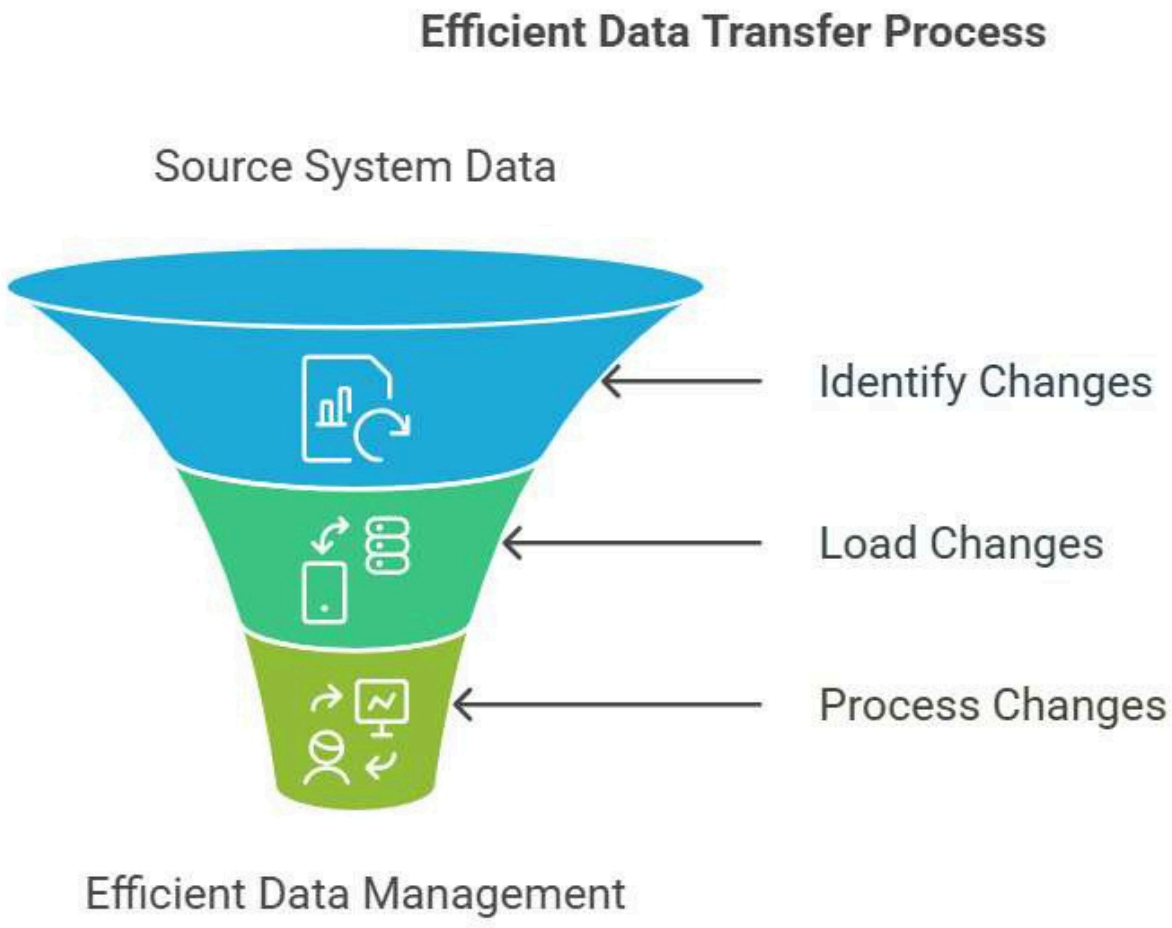
INCREMENTAL LOADING CONCEPT

Incremental load is a crucial concept in data integration and ETL processes, particularly when working with large datasets in **Azure Synapse Analytics.**



What is the Incremental Load?

Incremental load refers to the process of loading only the new or changed data from a source system into a target system, rather than reloading the entire dataset. This approach is particularly beneficial in scenarios where data volumes are large, and the cost of transferring and processing data can be significant. By focusing on just the changes, organizations can save time, reduce resource consumption, and improve overall efficiency.



Scenario

Assume you have an online retail store, and you have a lot of data being stored, which needs to be analyzed. There is an example case of understanding the changes in the data over period. Step 01 – Create 5 sample tables I have created multiple tables as follows:

Customer data table (To get the details of the customer) Create Customer id, Name , Phone number, Customer_datetime

SQL code-

```
--Customer data table (To get thedetailsofthecustomer     )
CREATE TABLE Customer (
    CustomerID INT,
    Name VARCHAR(100) NOT NULL,
    Phone VARCHAR(20),
    Customerupdateddate DATETIME       ---DeltaColumn
    );
```

Customer login table (To check the time spent online and on what products) Create Login id, Username, password and login_datetime

SQL code-

```
--Customer login table (To check the time spentonlineandonwhatproducts)
CREATE TABLE Login_id (
    LoginID INT,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Updatedlogindata DATETIME            ---DeltaColumn
    );
```

Payment table (To get the list of the transactions/payments completed) Create Transaction_id, Customer ID, Product ID and Transaction_datetime

SQL code-

```
--Payment table (To get the list of the transactions/payments completed)
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,       ---DeltaColumn
    CustomerID INT NOT NULL,
    ProductID INT NOT NULL,
    TransactionDate DATE NOT NULL,
    );
```

Inventory table (To get the list of items in the inventory) Create Product_id, Product_name, Price, Quantity.

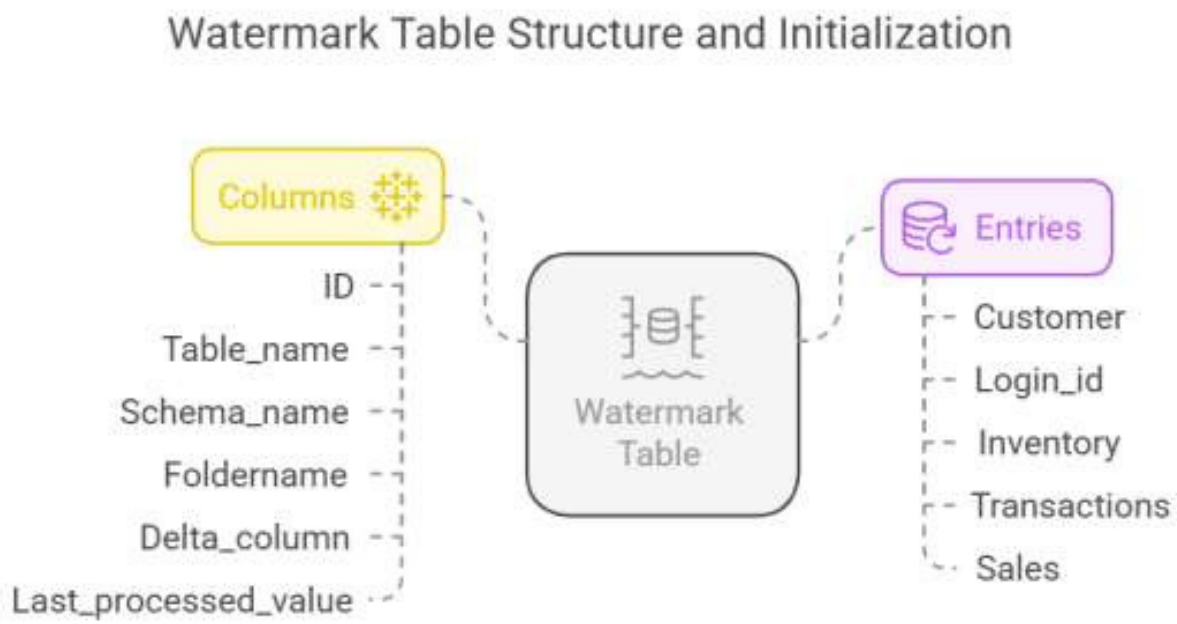
SQL code-

```
--Inventory table (To get the list of items in the inventory)
CREATE TABLE Inventory (
    ProductID INT PRIMARY KEY,            ---DeltaColumn
    ProductName VARCHAR(200) NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity >= 0),
    Price DECIMAL(10, 2) NOT NULL
    );
```

Sales table (To check the sales of the items) Create Sales ID, Product ID, Sales_log, Revenue

SQL code-

```
--Sales table (To check the salesoftheitems)
CREATE TABLE Sales (
    SalesID INT PRIMARY KEY,
    ProductID INT NOT NULL,
    Sales_log DATETIME,                  ---DeltaColumn
    Revenue DECIMAL(15, 2) NOT NULL
    );
```



Insert values/data into the tables

```
INSERT INTO Customer (CustomerID, Name, Phone, Customerupdateddate)
VALUES
(1, 'John Doe', '123-456-7890', '2023-01-01 10:00:00'),
(2, 'Jane Smith', '987-654-3210', '2023-02-01 14:30:00'),
(3, 'Bob Johnson', NULL, '2023-03-01 09:15:00');
```

```
INSERT INTO Login_id (LoginID, Username, Password, Updatedlogindata)
VALUES
(1, 'johndoe', 'password123', '2023-01-01 10:00:00'),
(2, 'janesmith', 'securepass', '2023-02-01 14:30:00'),
(3, 'bobjohnson', 'secret123', '2023-03-01 09:15:00');
```

```
INSERT INTO Inventory (ProductID, ProductName, Quantity, Price)
VALUES
(1, 'Laptop', 50, 999.99),
(2, 'Smartphone', 100, 699.99),
(3, 'Headphones', 75, 149.99);
```

```
INSERT INTO Transactions (TransactionID, CustomerID, ProductID, TransactionDate)
VALUES
(1, 1, 1, '2023-01-05'),
(2, 2, 2, '2023-02-10'),
(3, 3, 3, '2023-03-15');
```

```
INSERT INTO Sales (SalesID, ProductID, Sales_log, Revenue)

VALUES (1, 1, '2023-01-05 10:00:00', 999.99),
(2, 2, '2023-02-10 14:30:00', 699.99),
(3, 3, '2023-03-15 09:15:00', 299.98);
```

Step 02 – Create a watermark Table

This step helps to monitor the changes in the data i.e. it may be data entries, data modifications, etc.

SQL Code-

```
--Create a Watermark table
CREATE TABLE Watermark (


    ID INT PRIMARY KEY,                    --Can't accept similar id's or NULL -- Only
Unique Value
    Table_name VARCHAR(100),
       Schema_name VARCHAR(100),
       Foldername VARCHAR(50),
       Delta_column VARCHAR(100),
    Last_processed_value VARCHAR(255) NOT NULL
);
```

Inserted data into the watermark table

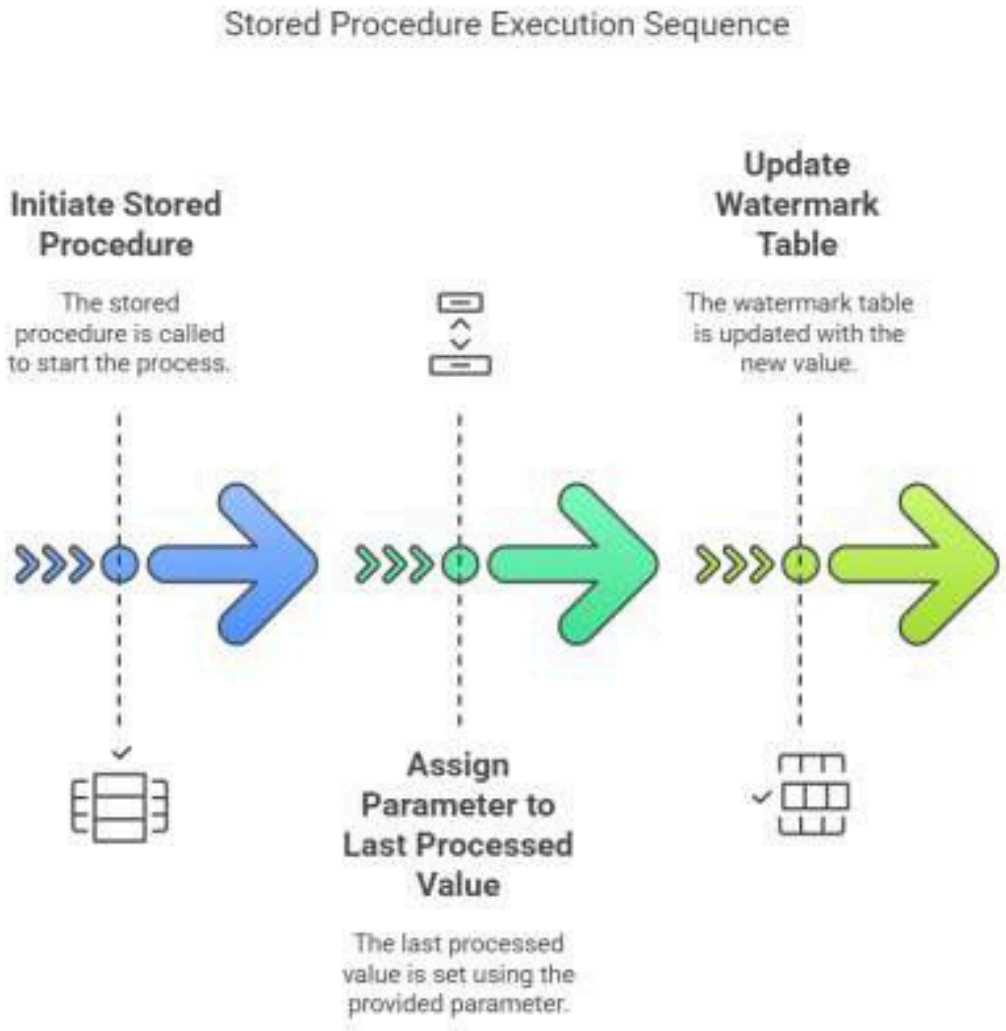
```
-- Initialize watermark entries
INSERT INTO Watermark
VALUES
(1,'Customer','dbo','RetailDB/Customer_data','Customerupdateddate','1900-01-01   00:00:00'),
(2,'Login_id','dbo','RetailDB/Login_id_data','Updatedlogindata','1900-01-01      00:00:00'),
(3,'Inventory','dbo','RetailDB/Inventory_data','ProductID','0'),
(4,'Transactions','dbo','RetailDB/Transactions_data','TransactionID','0'),
(5,'Sales','dbo','RetailDB/Sales_data','Sales_log','1900-01-01 00:00:00')
```

Now, the watermark table is ready to be used.

Step 03 – Create a stored procedure

This helps to assign a parameter to the changes in the last processed values corresponding to the table names.

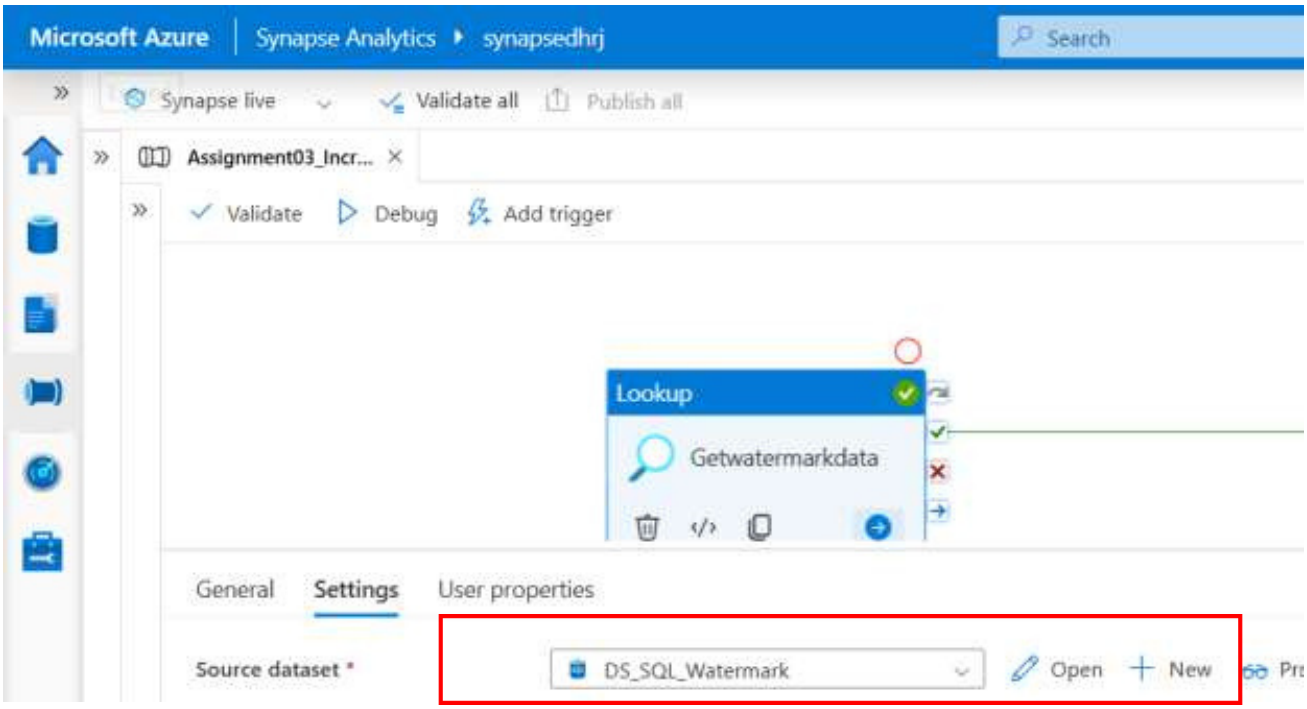
```
--CreateStored  procedure
        CREATE PROC USP_Watermark_RetailDB
               @lpv VARCHAR(100),
               @TBname VARCHAR(100)
        AS
        BEGIN
                UPDATE Watermark
                SET Last_processed_value=@lpv WHERE Table_name=@TBname
        END
```



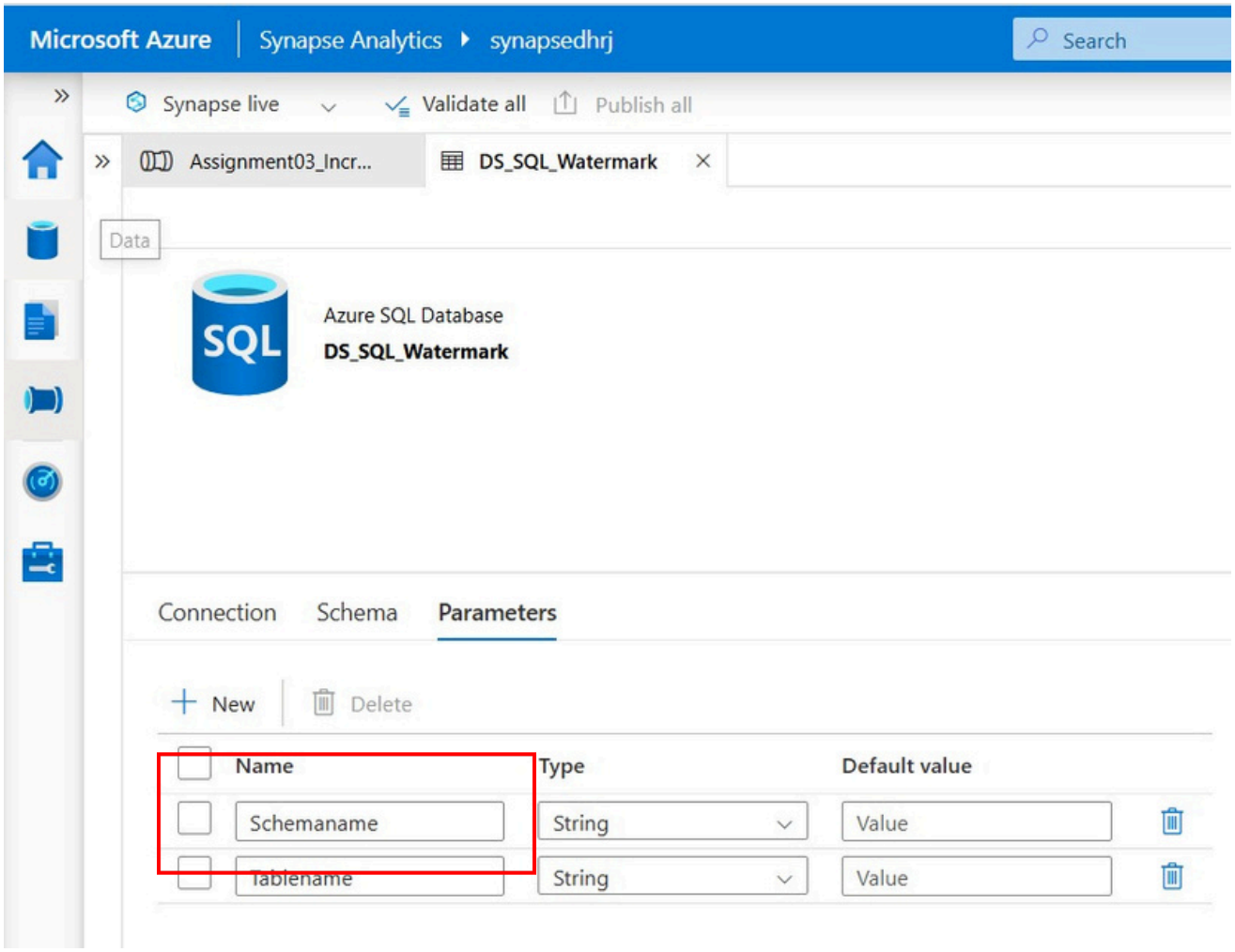
Step 04 – Creation of pipeline in Synapse

- Create a lookup activity and named it as Getwatermarkdata. Click on source dataset and create a new linked service with SQL server and name

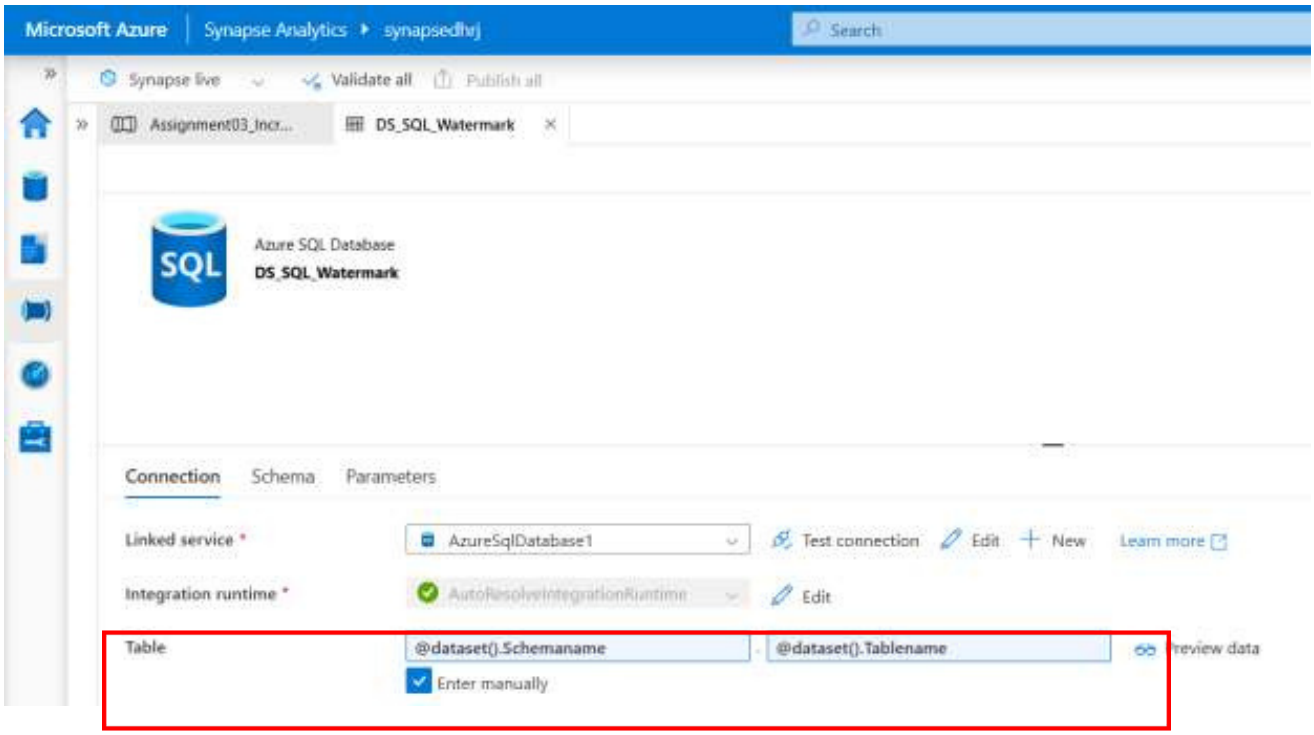
- Its as DS_SQL_Watermark.



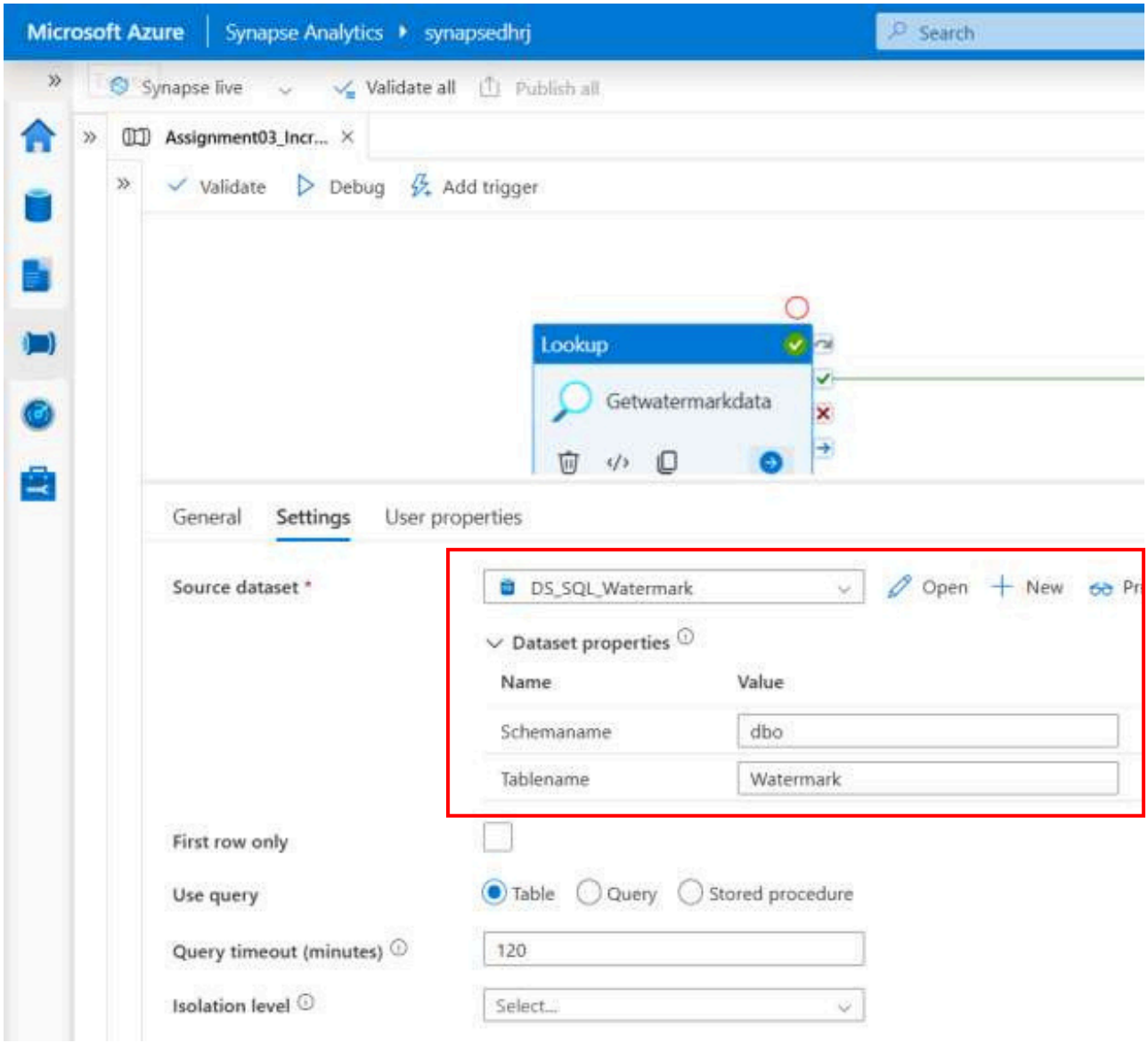
- Open the DS_SQL_Watermark dataset.
- Create 2 parameters i.e. Schemaname and Tablename for dynamically assigning the Schemaname and Tablename.



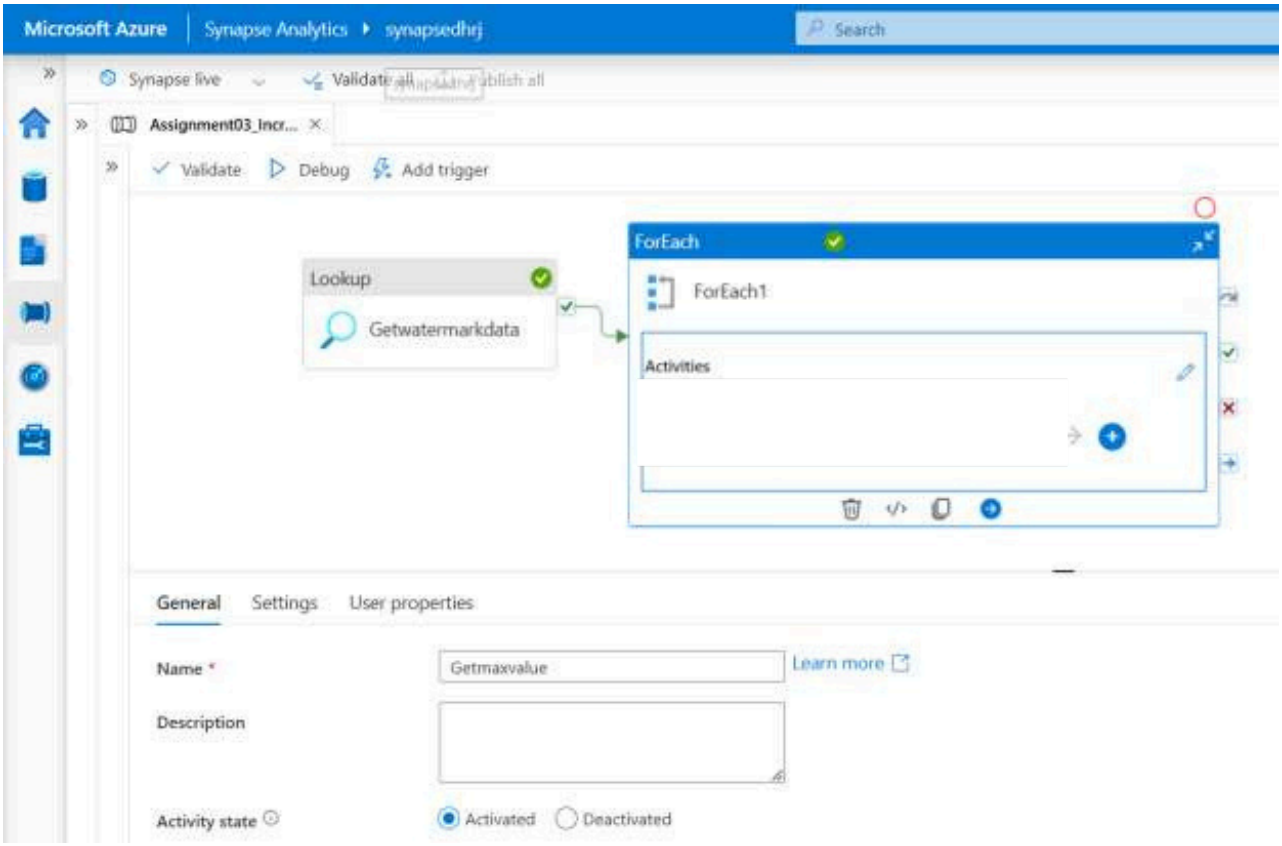
- Assigning dynamic variables to the Schema and Table name for the table



- Connect the source dataset to DS_SQL_Watermark (Watermark table created in SQL database)
- Enter the values of fields Schemaname and Tablename as dbo and Watermark respectively.

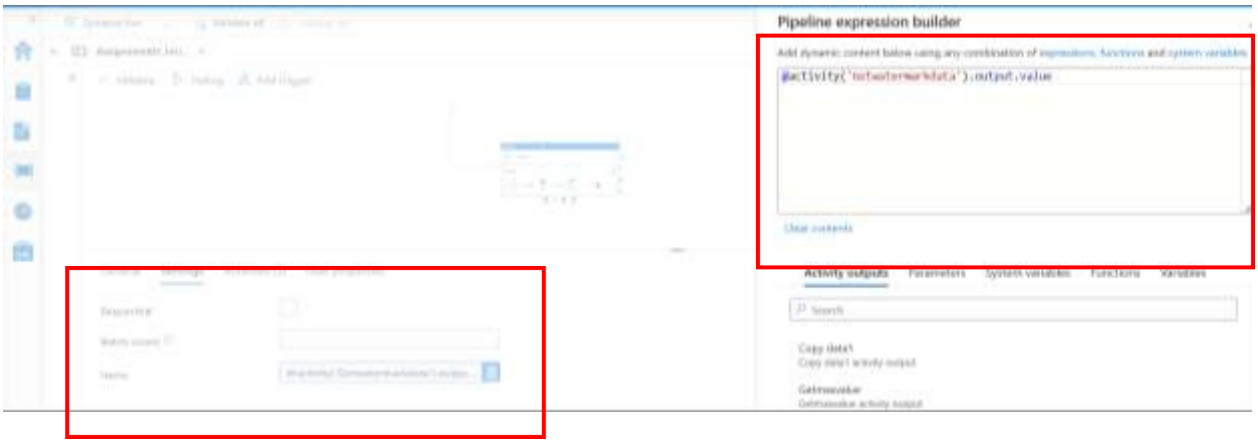


- Create a Foreach activity
- Connect the lookup activity i.e. Getwatermarkdata with the Foreach activity

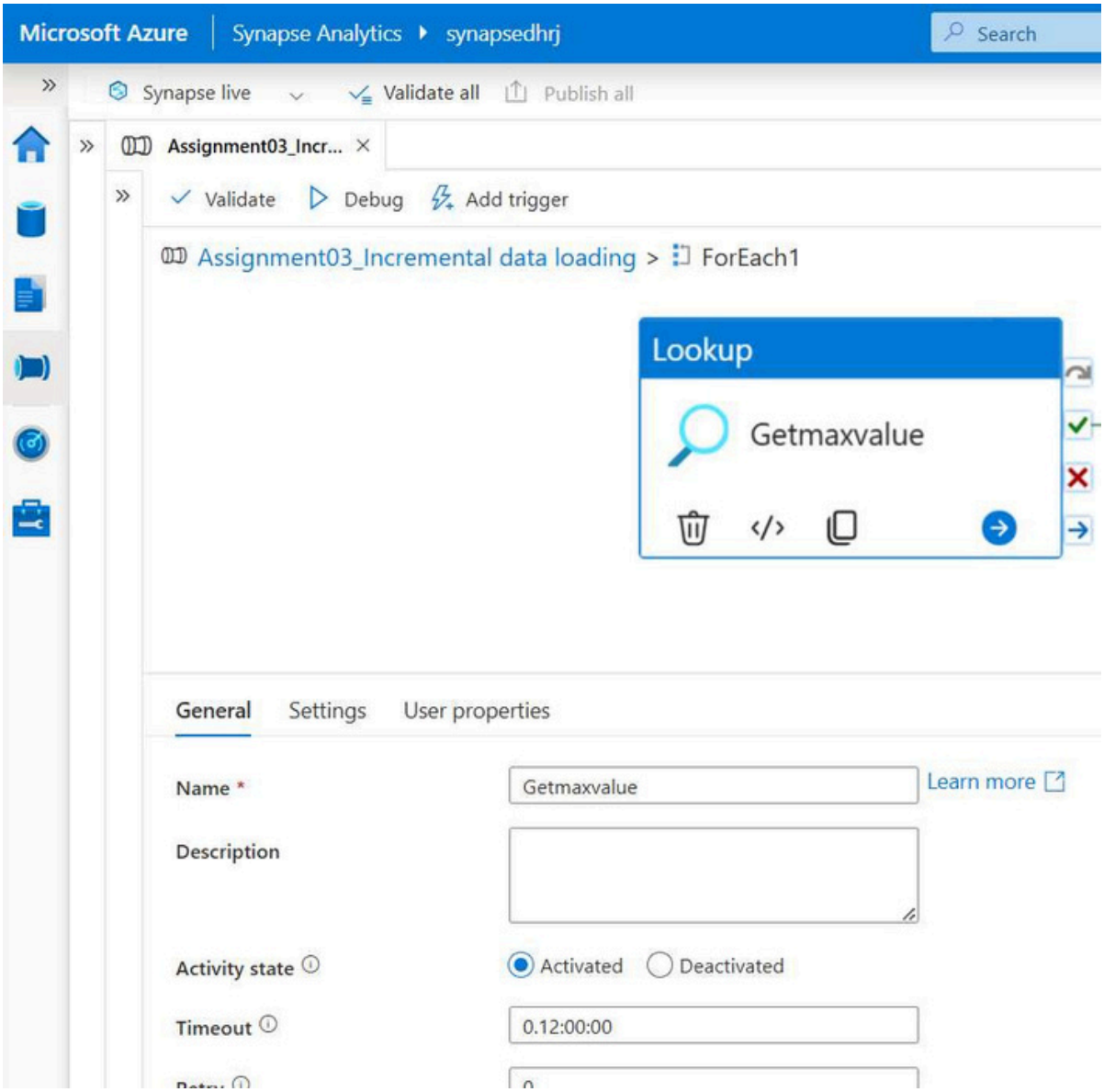


- Provide an expression in the items field in the Settings tab to connect with the lookup activity.

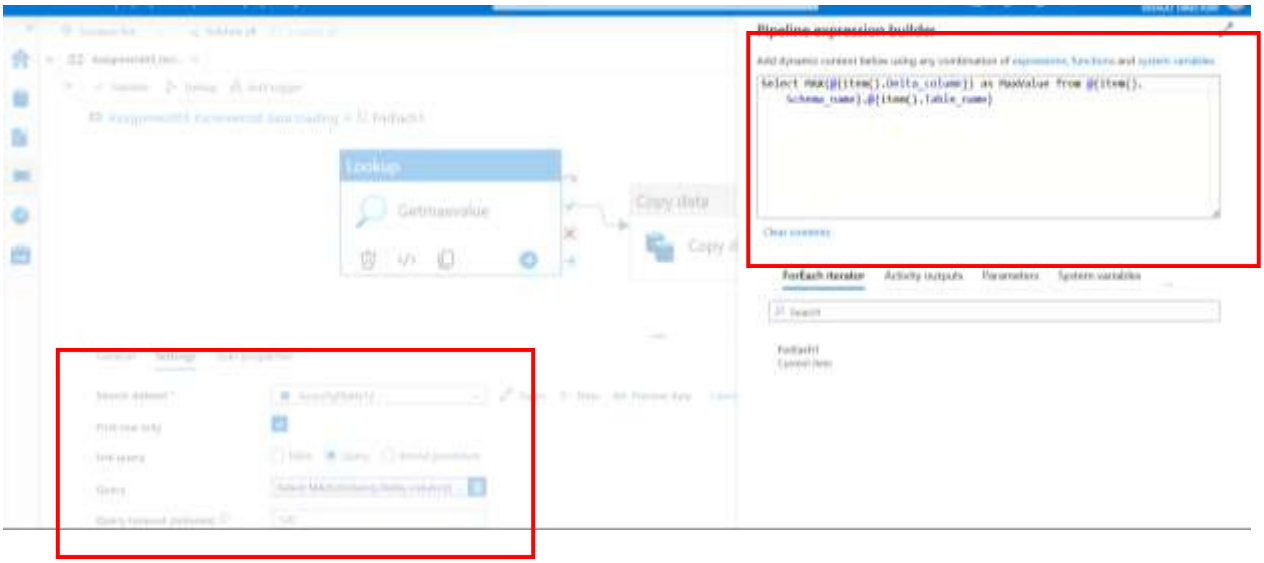
- Go to the pipeline expression builder-@activity(‘Lookup tablename’).output.value i.e. in current scenario @activity('Getwatermarkdata').output.value



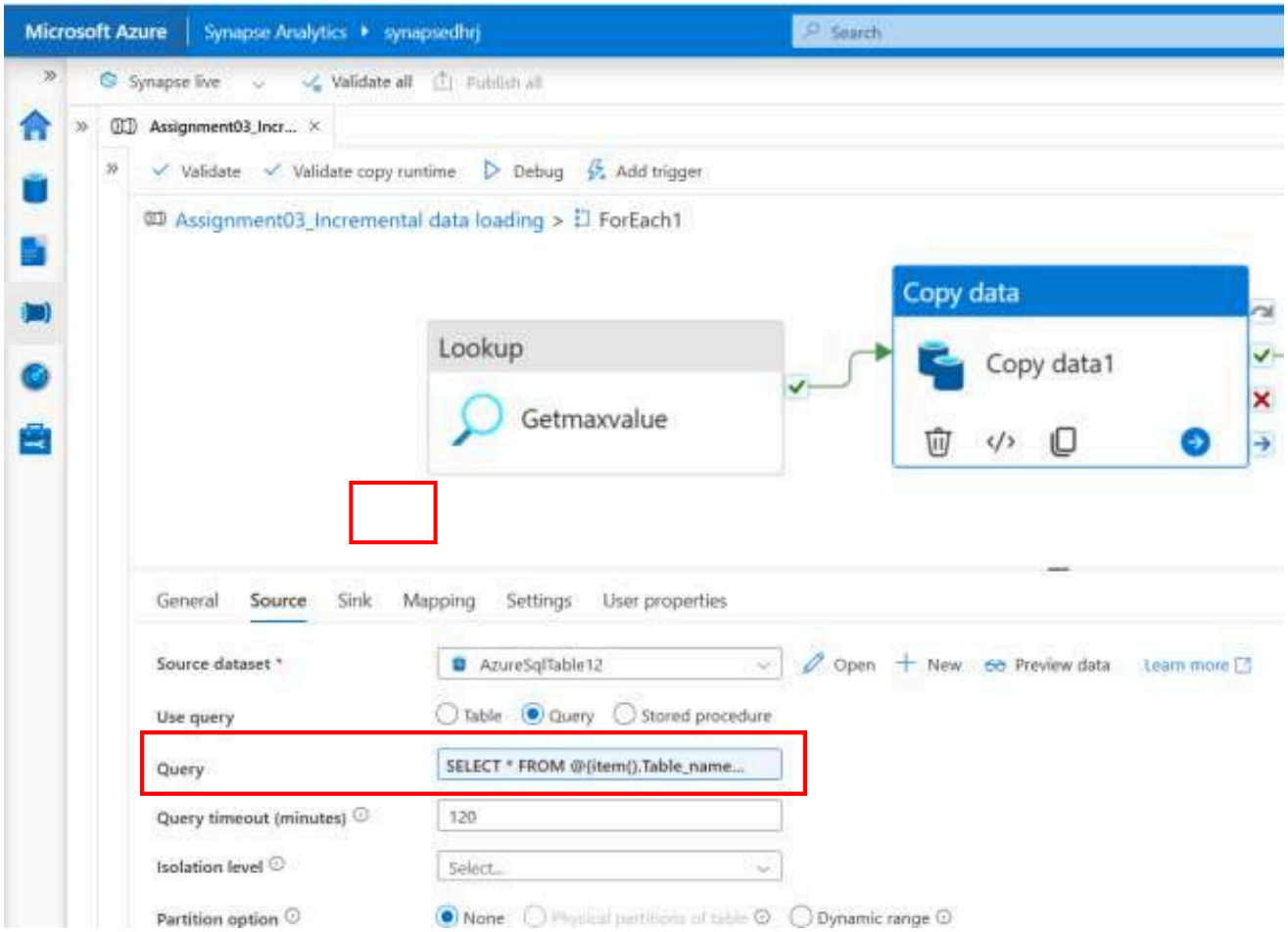
- Created a Lookup activity inside the Foreach activity for deriving the maximum value from the watermark table.
- Name it as lookup activity as Getmaxvalue



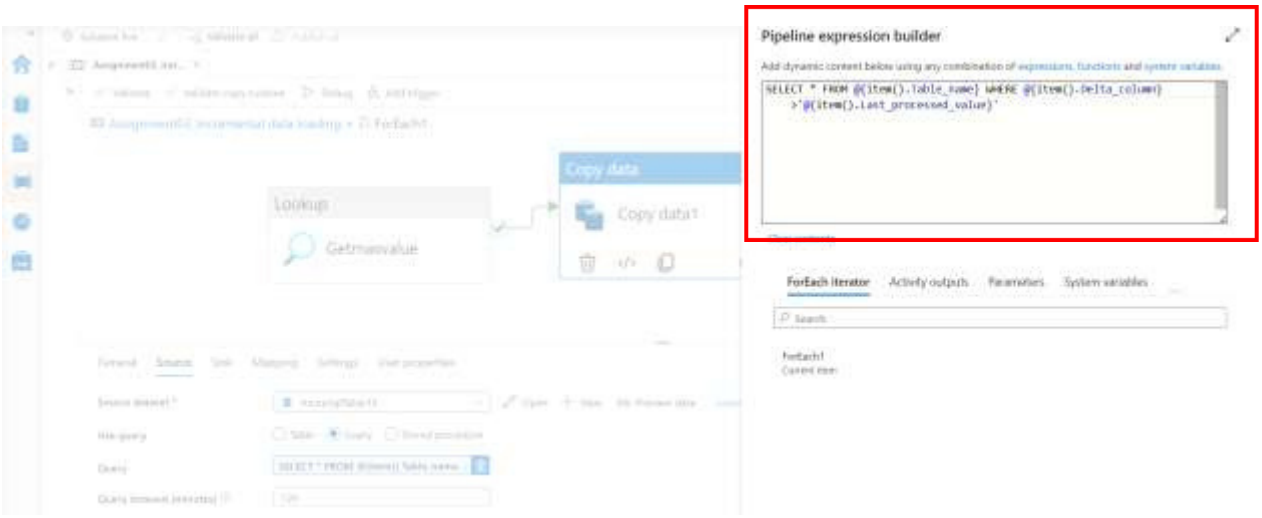
- Providing a query dynamically to get the maximum value from the watermark table General expression (For getting the maximum value from the watermark table) SELECT MAX('Columnname') as MaxValue FROM dbo.Watermark
- Dynamic expression (Converting dynamically the above expression) SECLECT MAX(@(item().Delta_Column)) as MaxValue FROM @(item().Schema_name).@(item().Table_name)



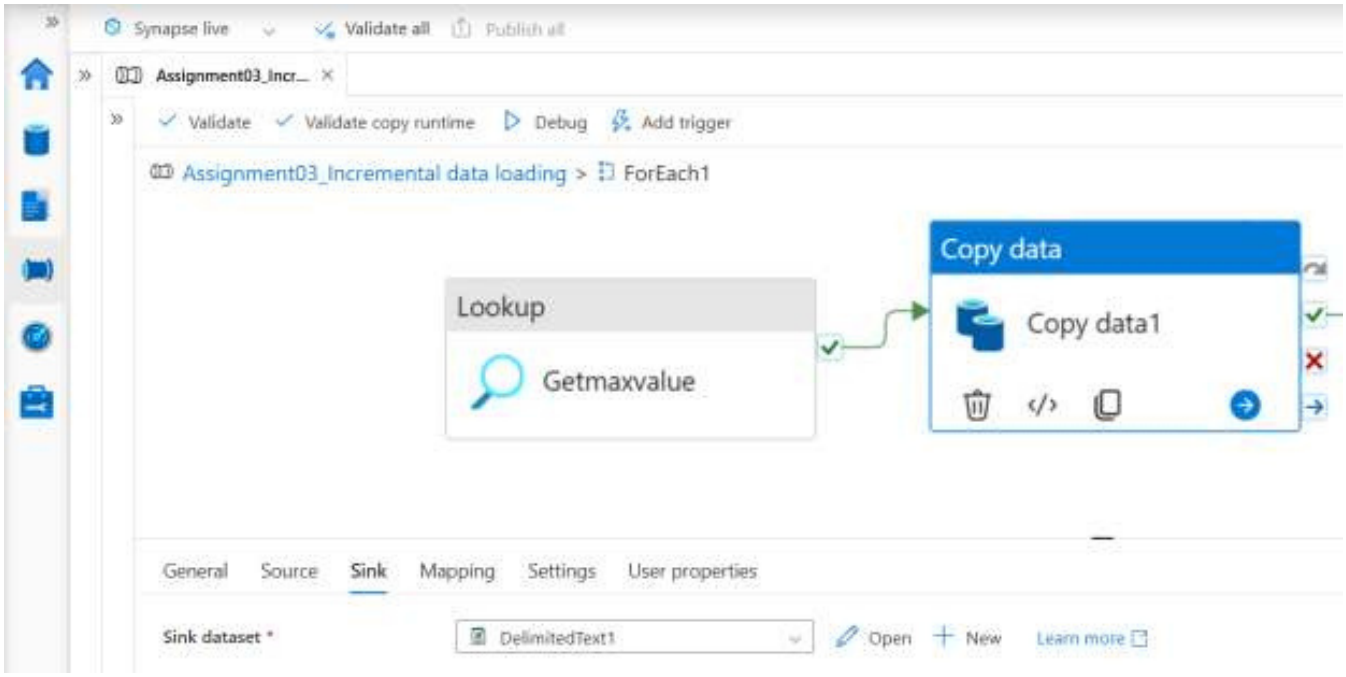
- Adding a Copy data activity
- Connect with the lookup activity i.e. Getmaxvalue activity.
- Select Source and write a dynamic query to copy only the modified values from the watermark table



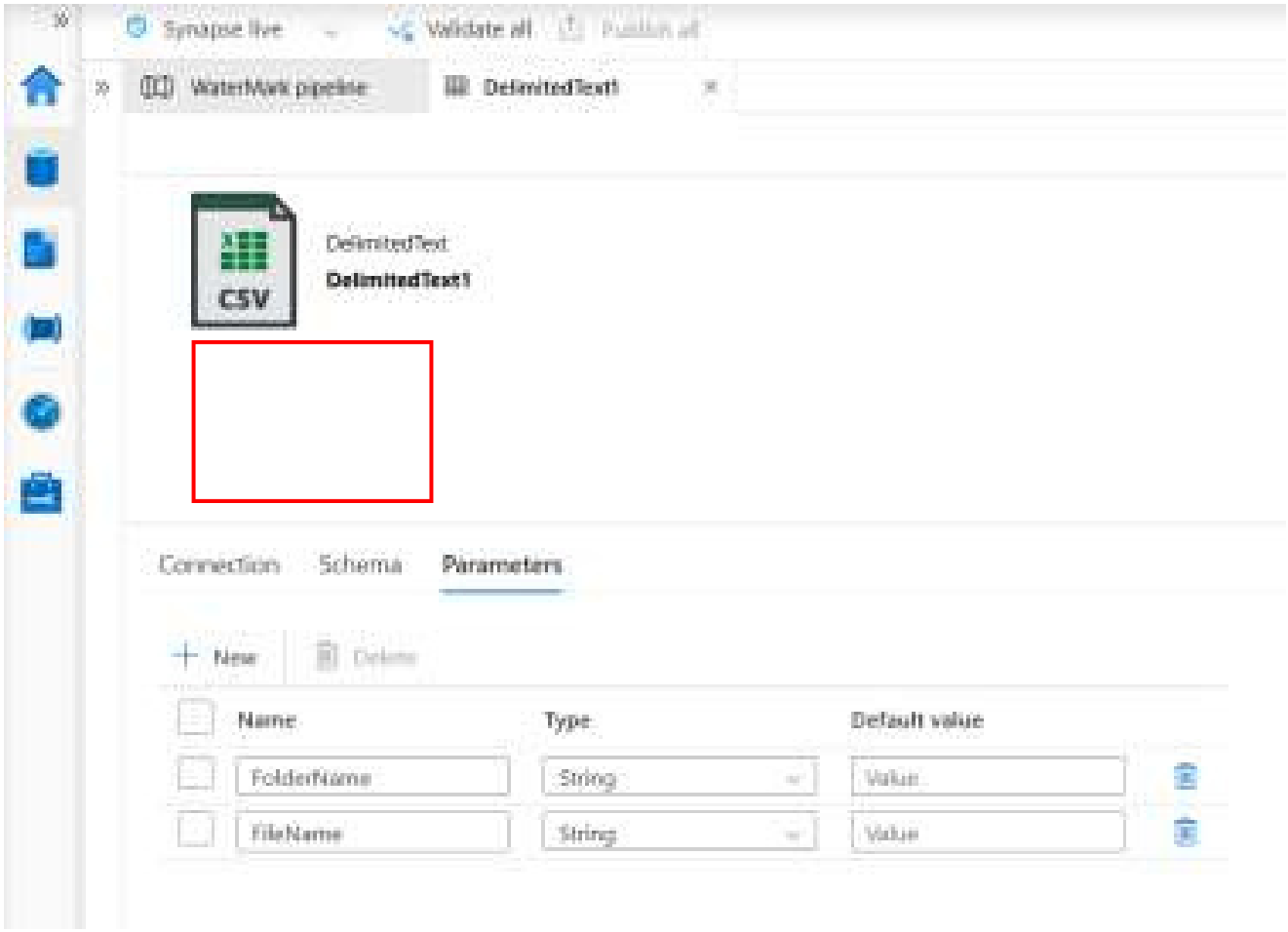
- General expression (To check for modified values or changes in the data entries) SELECT * FROM TABLE WHERE Delta_Column > ‘Last processed value column’
- Dynamic expression (Converting dynamically the above expression) SELECT * FROM @(item().Table_name) WHERE @(item().Delta_column)>’@(item().Last_processed_value)



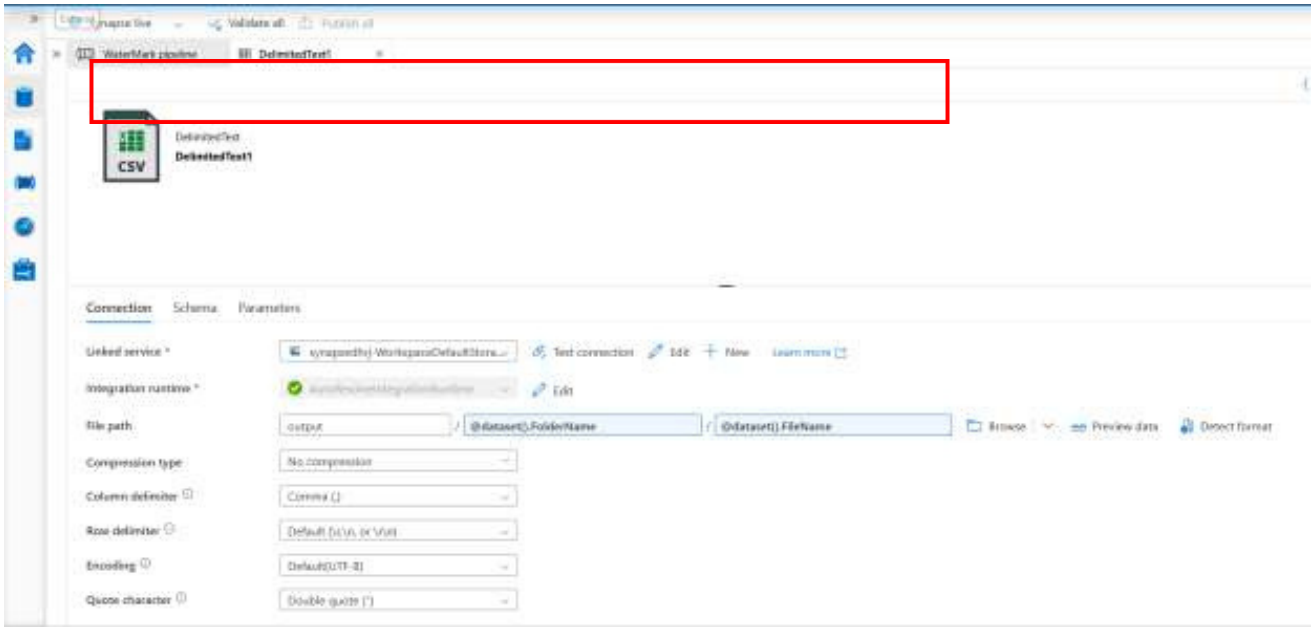
- Select Sink and create a dataset name DelimitedText1 to store the output into a Azurelakegen2 or Blob storage .



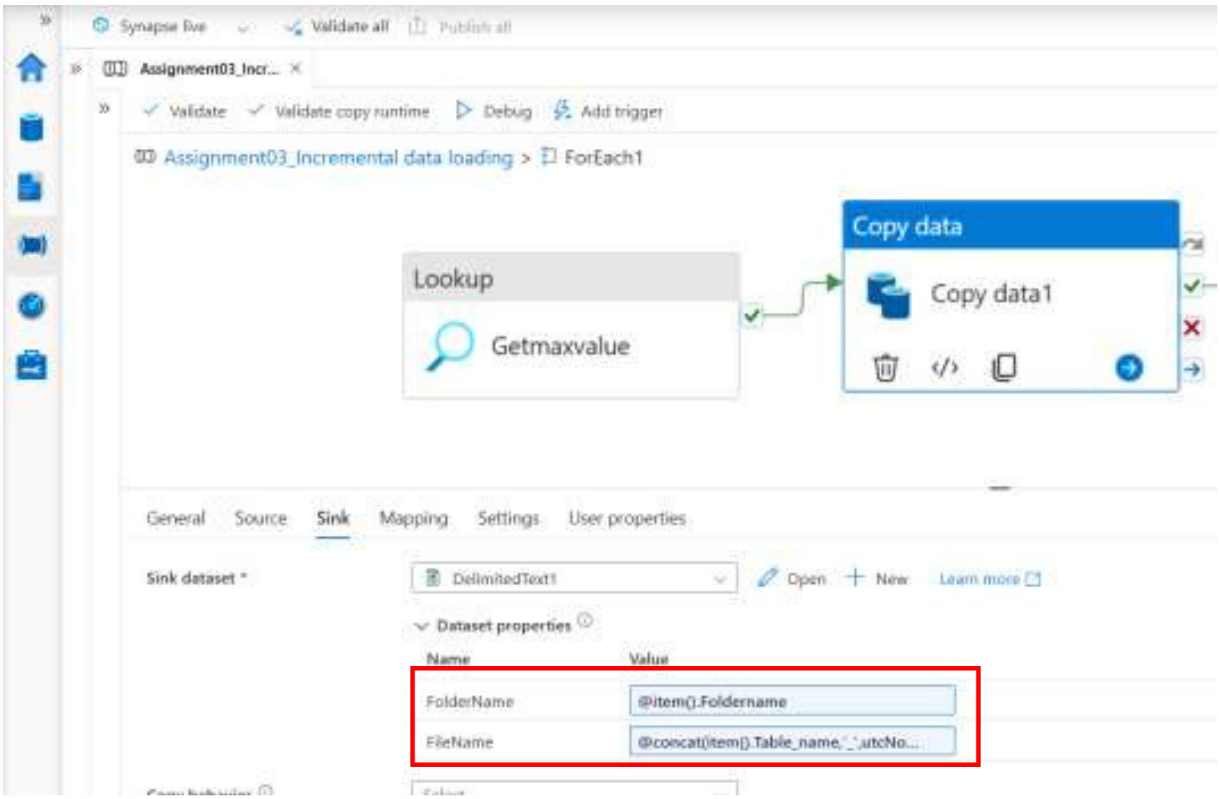
- Create 2 parameters named FolderName and FileName to store the folder name and file names respectively



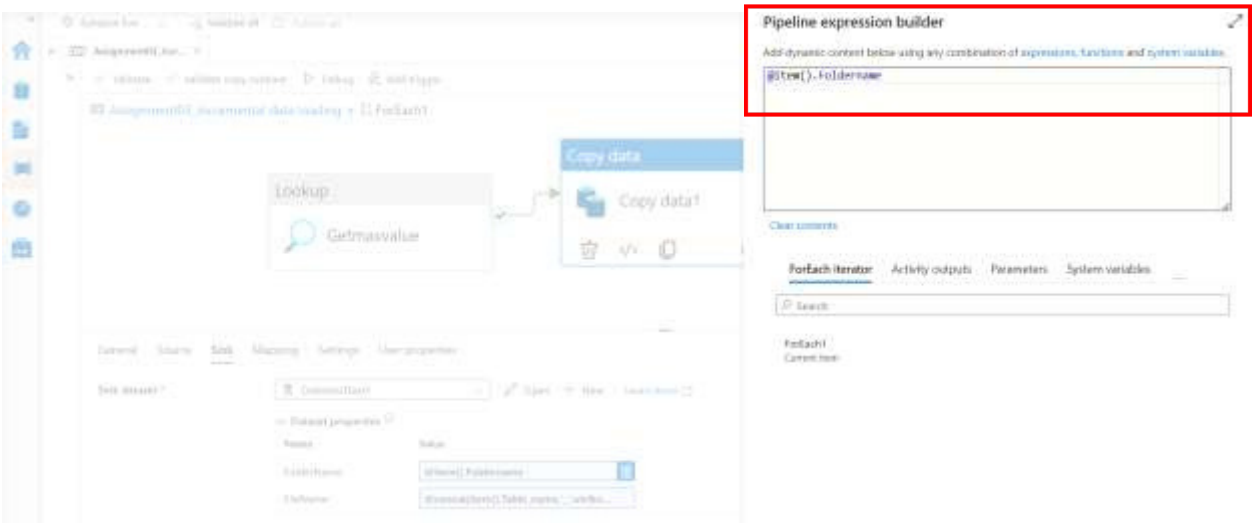
- Assign the parameters dynamically to the folder name and file name in the File path.



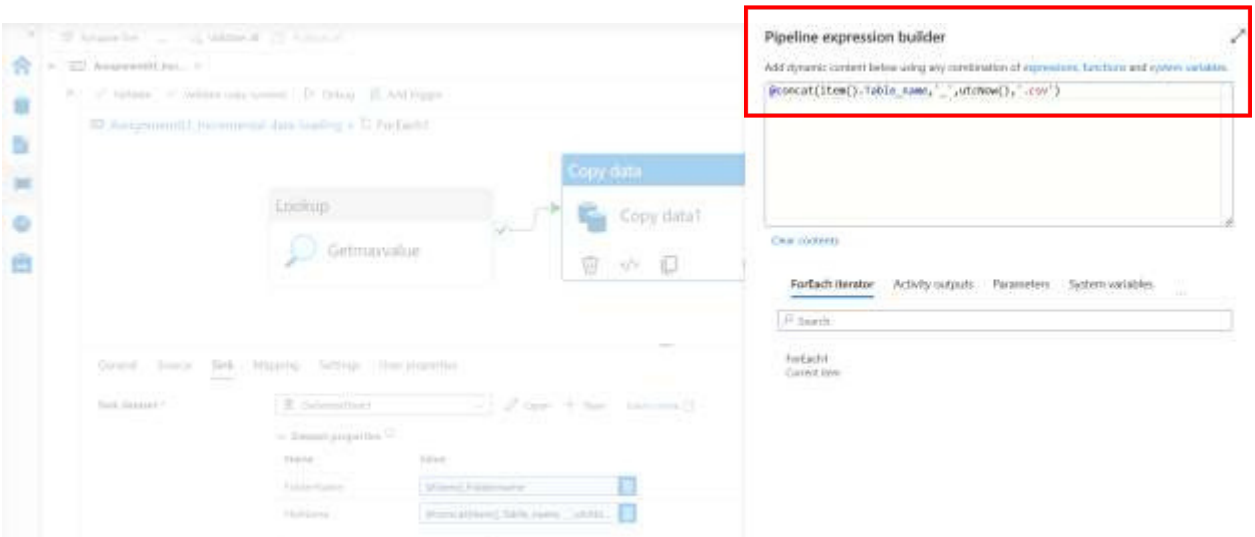
- Provide expressions as dynamic queries in the Foldername field and Filename fields.



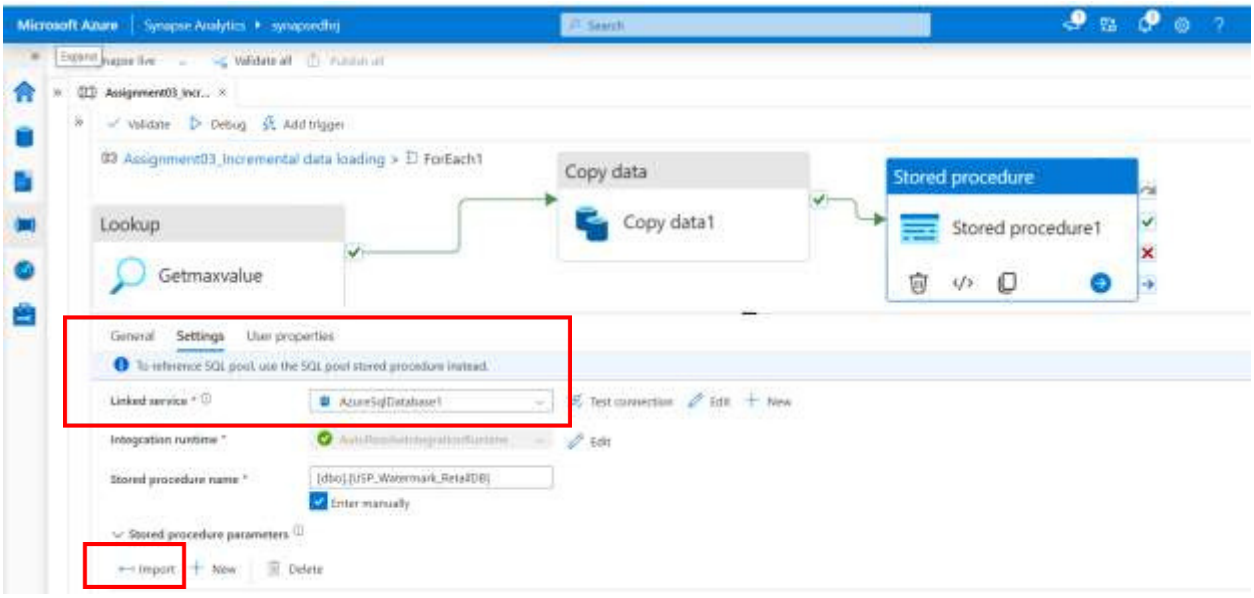
- Expression for the Foldername field would be @(item().Foldername)
- This expression will output the values with corresponding foldernames.



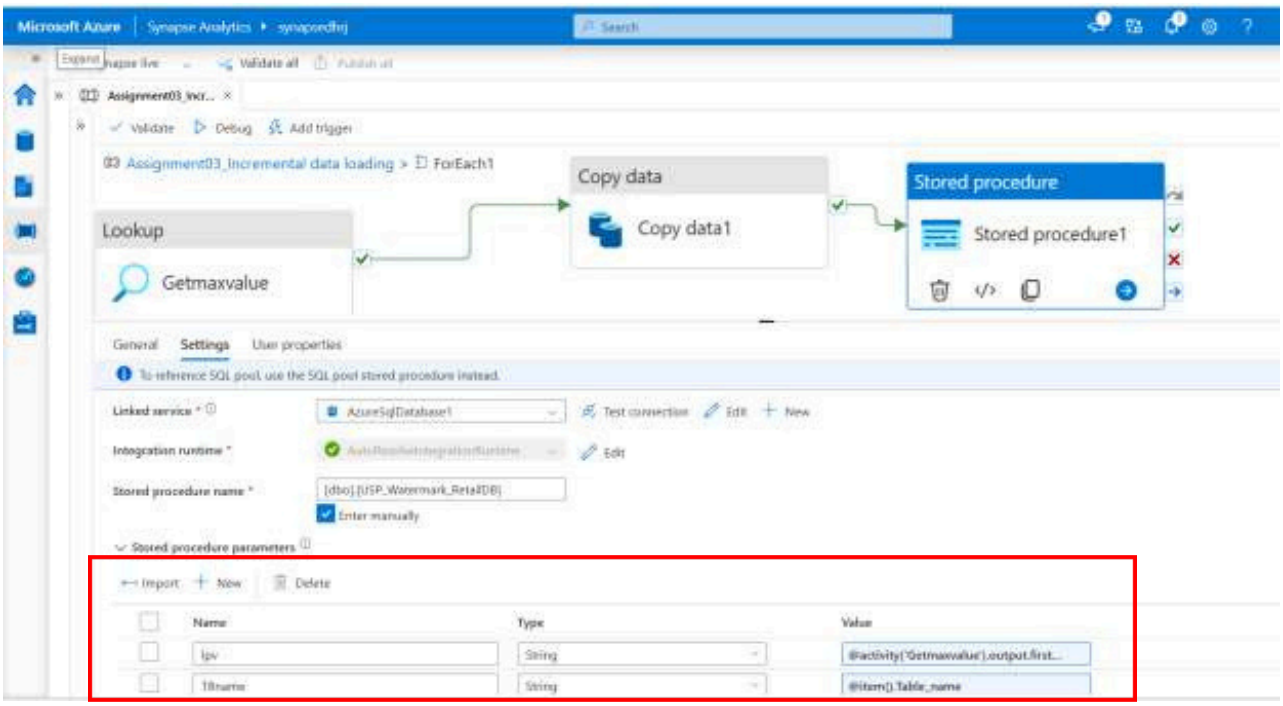
- Expression for the Foldername field would be @(item().Foldername) and @concat(item().Tablename,'_',utcNow(),'.csv')
- This expression will output the values as a .csv file format with current timestamp.



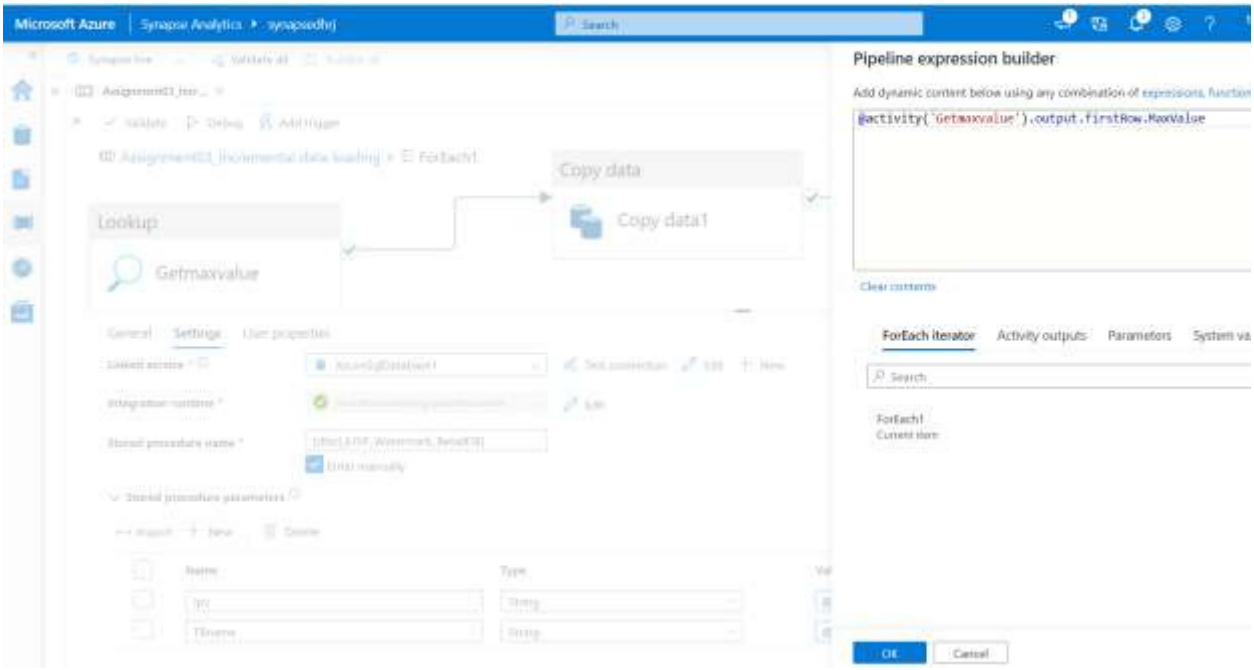
- Next Step - Adding a Stored procedure activity.
- Linking the source to the SQL database.



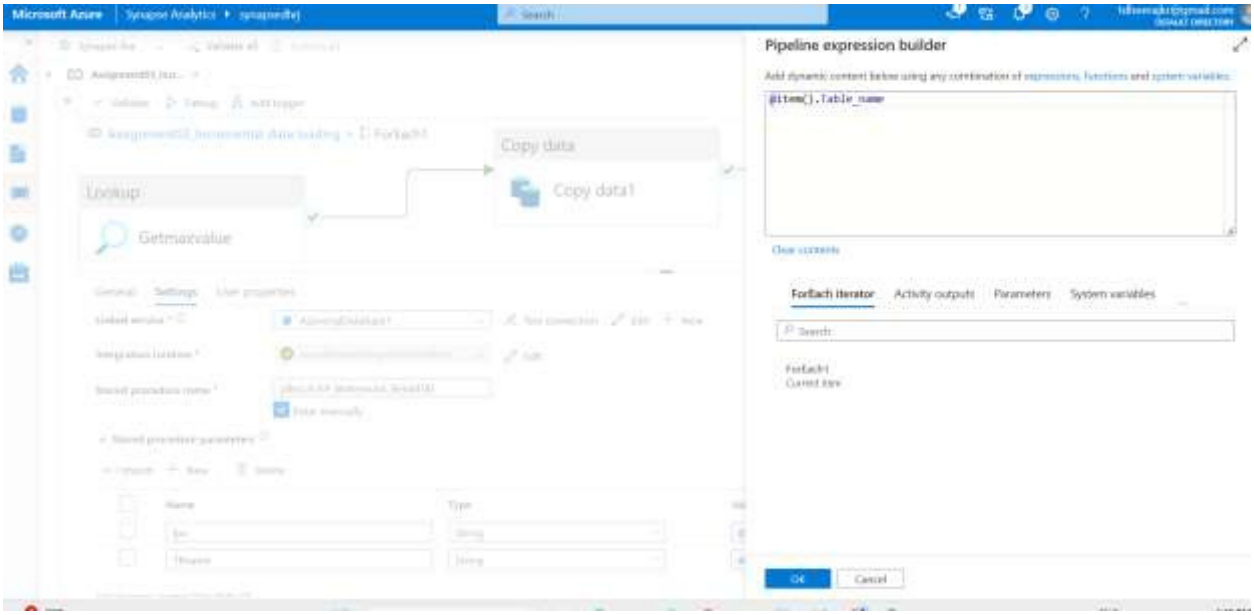
- Next click on Import parameters to import all the stored procedures created .



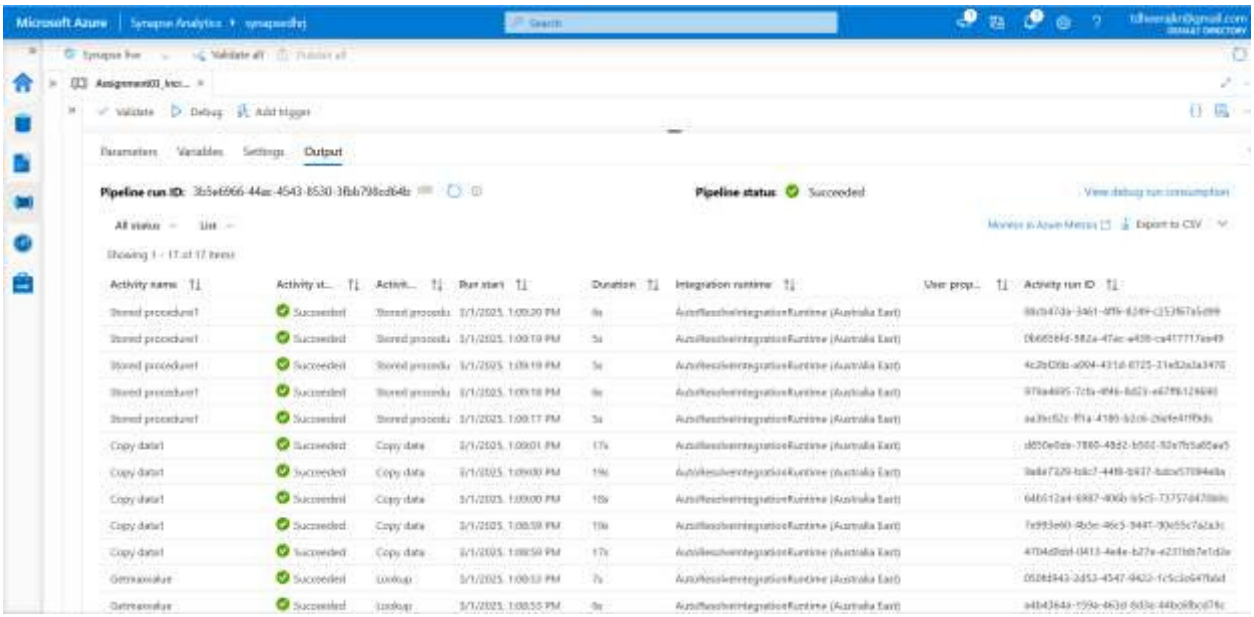
- Providing the dynamic expression for the lpv @activity('Getmaxvalue').output.firstRow.Maxvalue
- This will only input the modified values from the dataset.



- Providing the dynamic expression for the TBname .@item().Table_name.
- This will only input the modified values from the respective Table names.

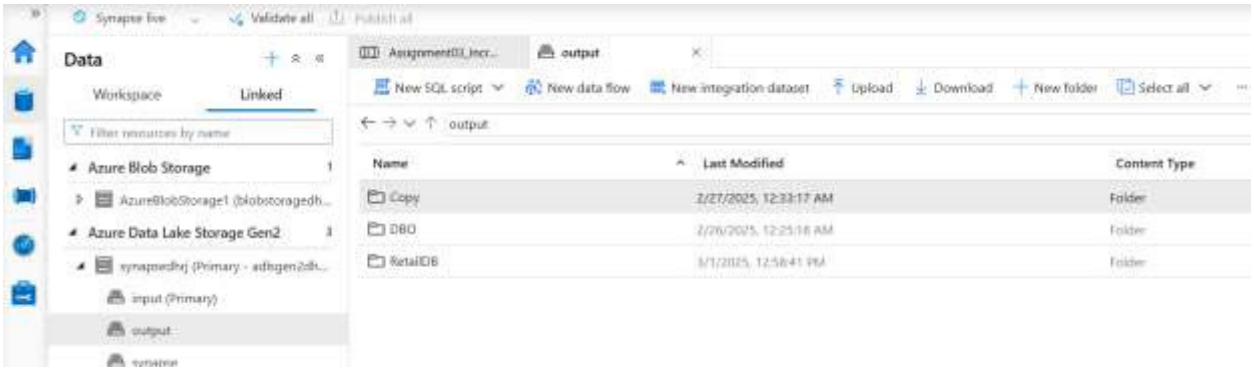


- Publishing and checking the pipeline if its working

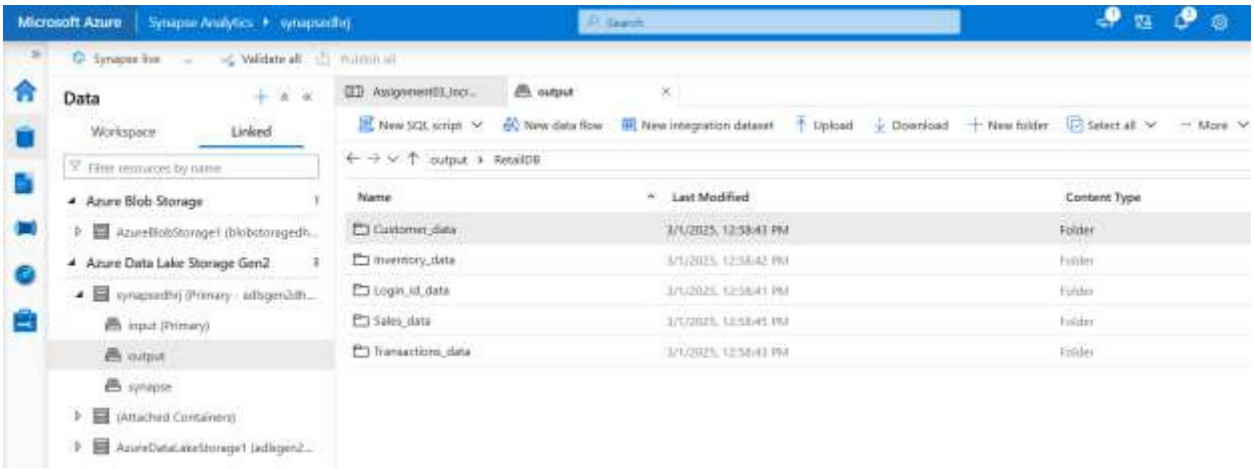


Checking the output folders...

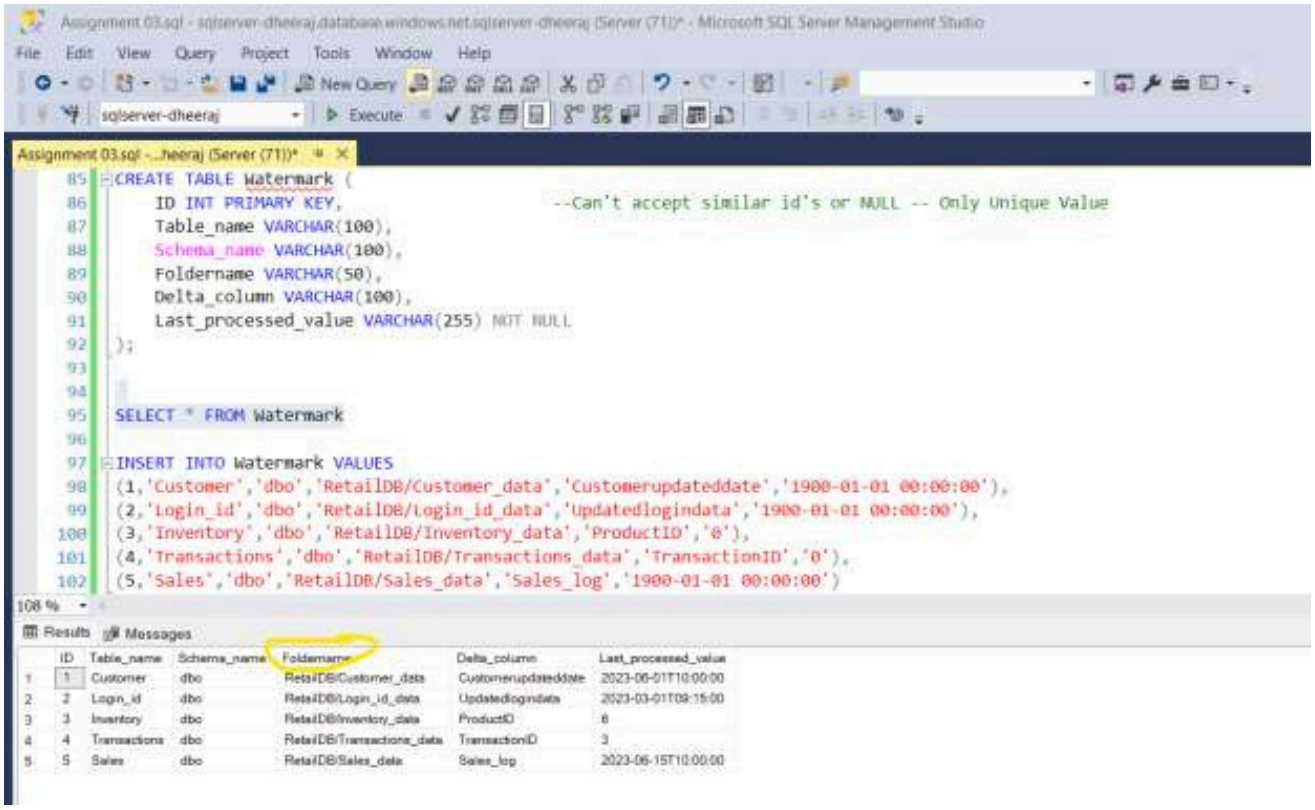
- RetailDB folder is created



- As mentioned in the Watermark table all the corresponding folders have been created i.e. Customer_data folder, Inventory_data folder, Login_id_data folder, Sales_data folder and Transactions_data folder.



- Cross checking the folder names in the SSMS.
- Checking in SSMS whether the name are matching with the Foldername



- Checking for the Incremental load.
- Adding more values into Customer, Inventory and Sales tables and publish in Synapse to check whether the incremental load is working or not ?

Assignment 03.sql -...heeraj (Server (71))*

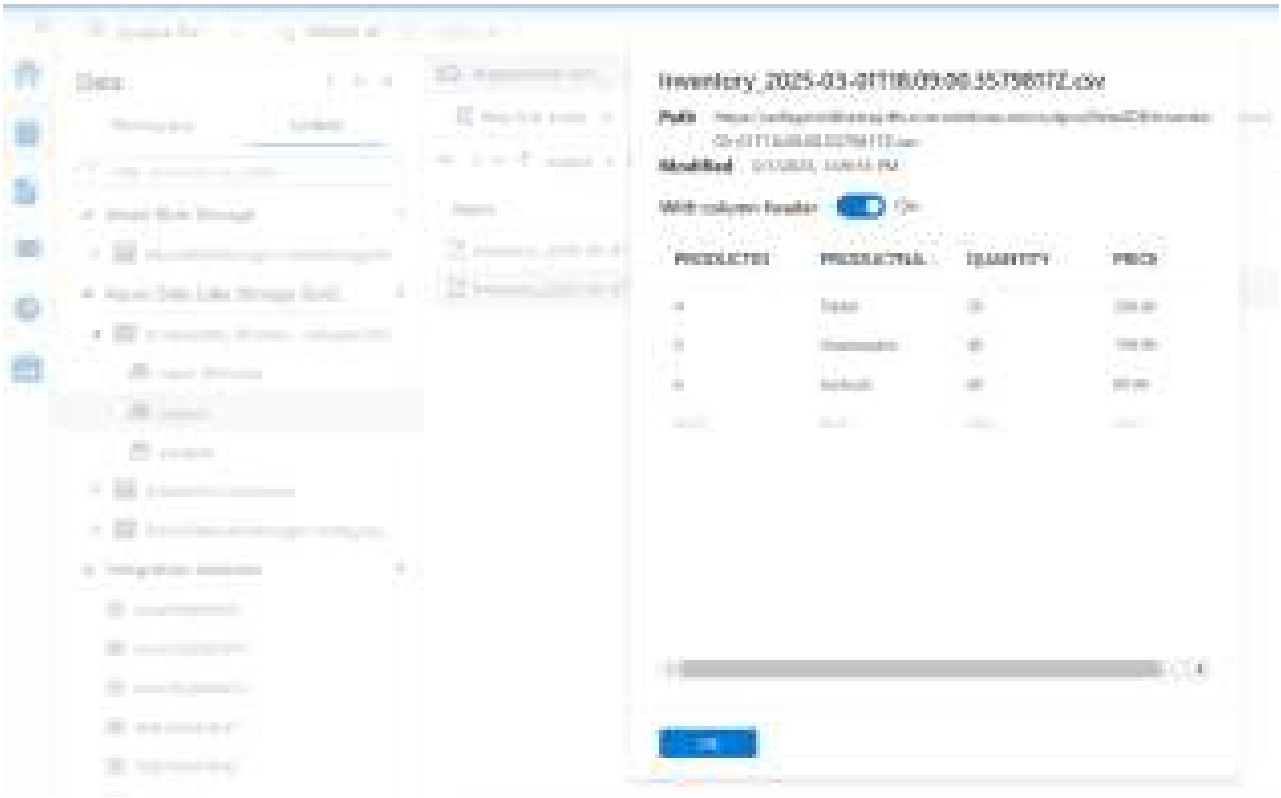
```
124  END
125
126
127  INSERT INTO Customer (CustomerID, Name, Phone, Customerupdateddate)
128  VALUES
129  (4, 'Alice Brown', '555-123-4567', '2023-04-01 12:00:00'),
130  (5, 'Charlie Davis', '555-789-0123', '2023-05-01 11:00:00'),
131  (6, 'Emily Wilson', NULL, '2023-06-01 10:00:00');
132
133  INSERT INTO Inventory (ProductID, ProductName, Quantity, Price)
134  VALUES
135  (4, 'Tablet', 30, 299.99),
136  (5, 'Smartwatch', 40, 199.99),
137  (6, 'Earbuds', 60, 89.99);
138
139
140  INSERT INTO Sales (SalesID, ProductID, Sales_log, Revenue)
141  VALUES
142  (4, 4, '2023-04-05 12:00:00', 299.99),
143  (5, 5, '2023-05-10 11:00:00', 199.99),
144  (6, 6, '2023-06-15 10:00:00', 89.99);
145
```

108 %

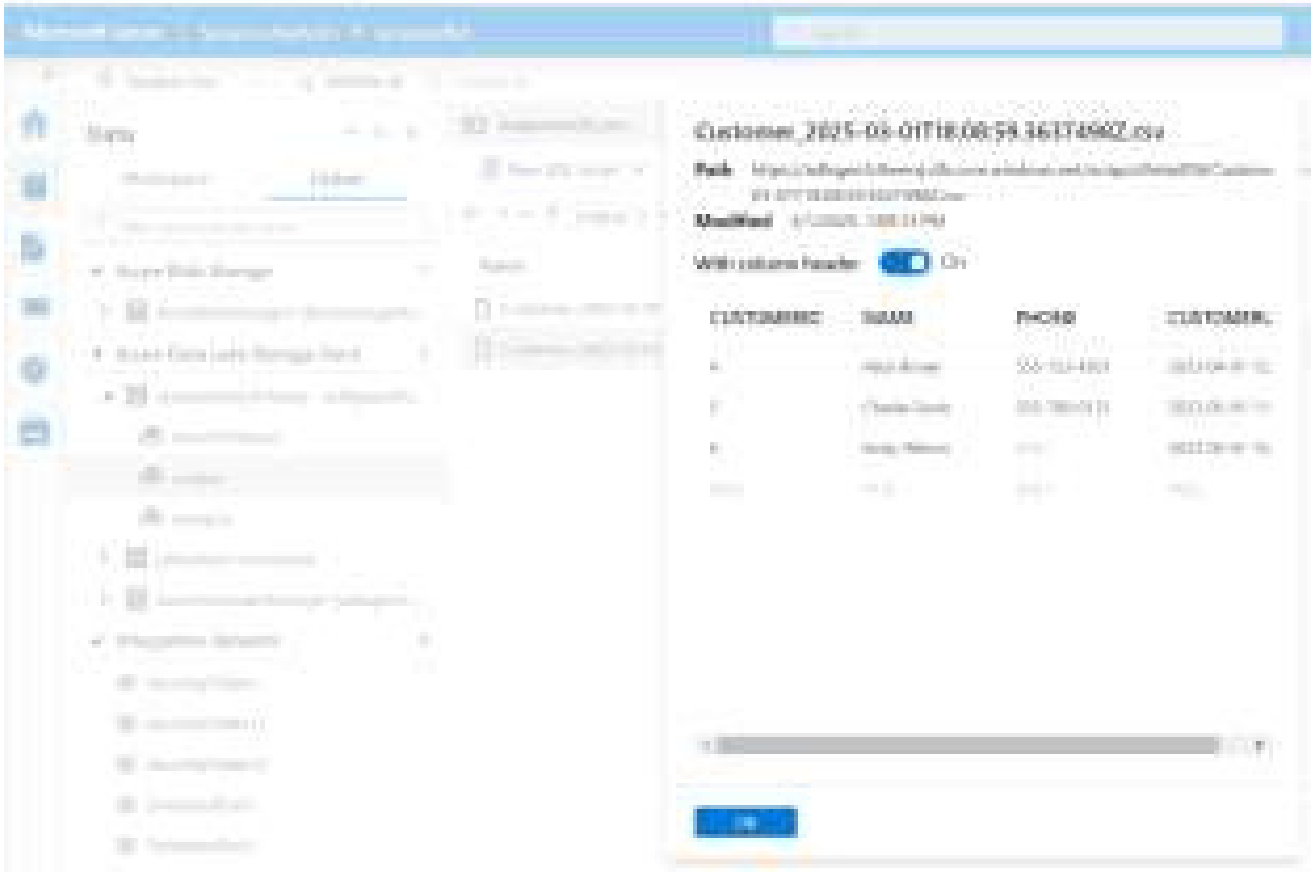
Results | Messages

	ID	Table_name	Schema_name	Foldername	Delta_column	Last_processed_value
1	1	Customer	dbo	RetailDB/Customer_data	Customerupdateddate	2023-06-01T10:00:00
2	2	Login_id	dbo	RetailDB/Login_id_data	Updatedlogindata	2023-03-01T09:15:00
3	3	Inventory	dbo	RetailDB/Inventory_data	ProductID	6
4	4	Transactions	dbo	RetailDB/Transactions_data	TransactionID	3
5	5	Sales	dbo	RetailDB/Sales_data	Sales_log	2023-06-15T10:00:00

- Verifying the output values in the corresponding file locations.
- **Inventory table**



- **Customer table**



- **Sales table**