Customer 360 Data Integration

—

Sudhanshu Kharbanda

Overview

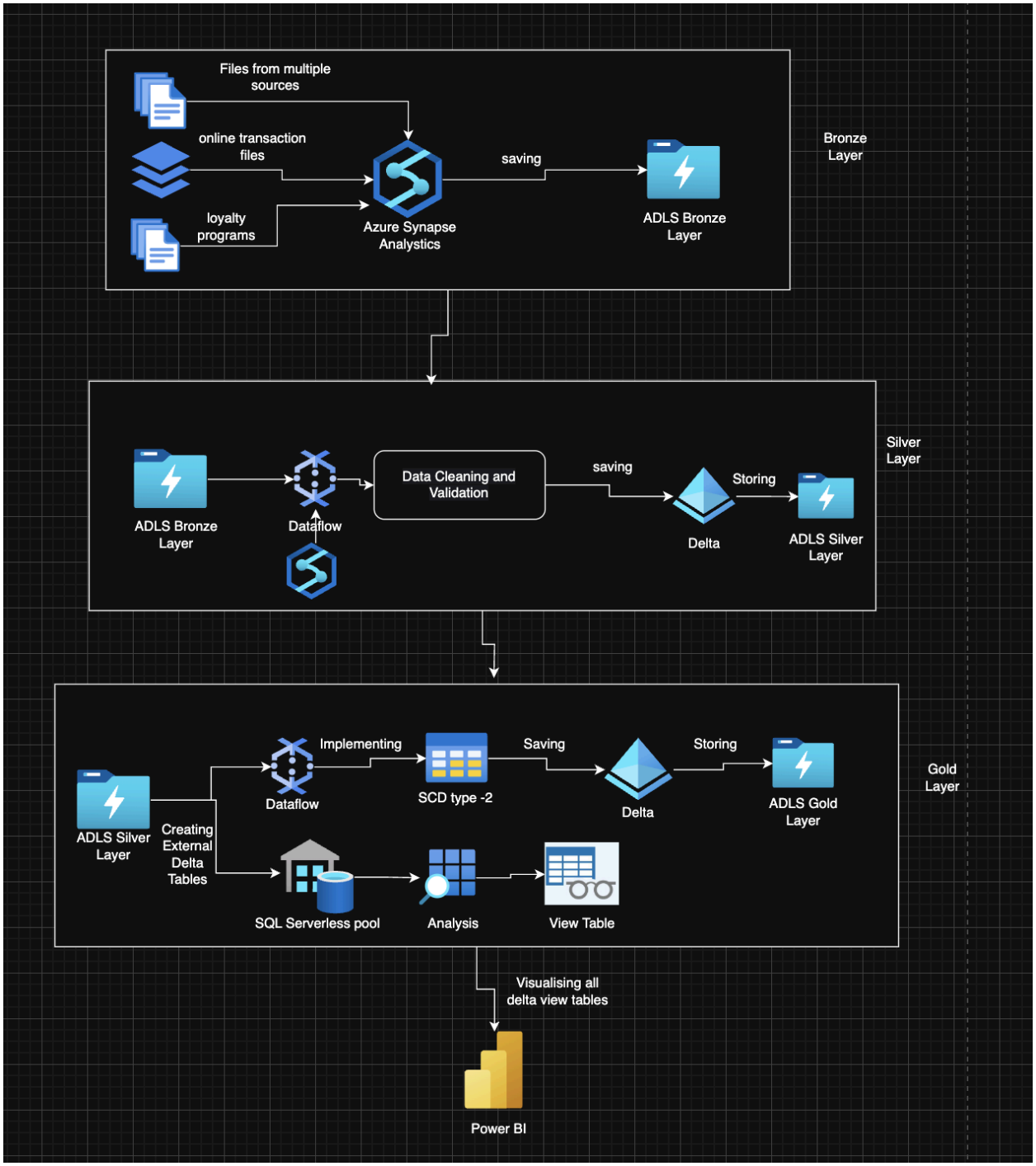
The primary objective of this project is to design and implement a **Customer 360 data pipeline** that unifies data from multiple retail channels—online transactions, in-store purchases, customer service interactions, and loyalty programs—into a centralized, analytics-ready system on Azure.

Specific goals include:

- **Integrate diverse data sources** into a unified data lake for centralized storage.
- **Implement a structured data model** with **SCD Type-2** to capture both current and historical customer information.
- **Build a scalable data pipeline** leveraging Azure Synapse Analytics, ADLS, and serverless SQL pools for ingestion, transformation, and aggregation.
- **Deliver business-ready insights** by creating analytical views such as Average Order Value (AOV), customer segmentation, and peak purchase trends.
- **Enable self-service analytics** through interactive dashboards in Power BI, providing business users with actionable insights to improve decision-making and customer engagement.

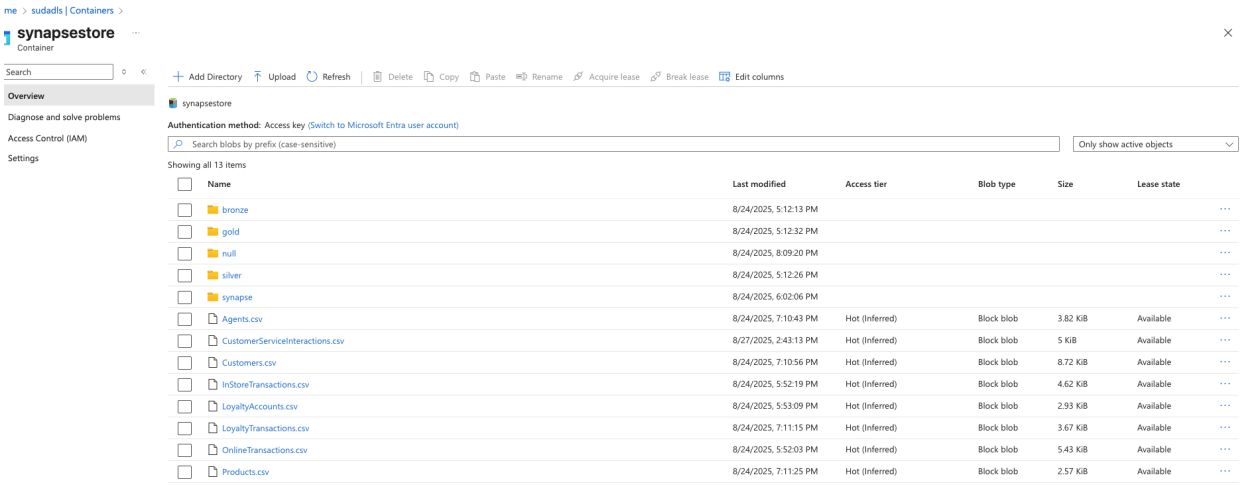
The ultimate objective is to empower the retail business with a **single, trusted view of customers** that supports personalized marketing, improved loyalty strategies, and data-driven operational efficiency.

Architecture Diagram

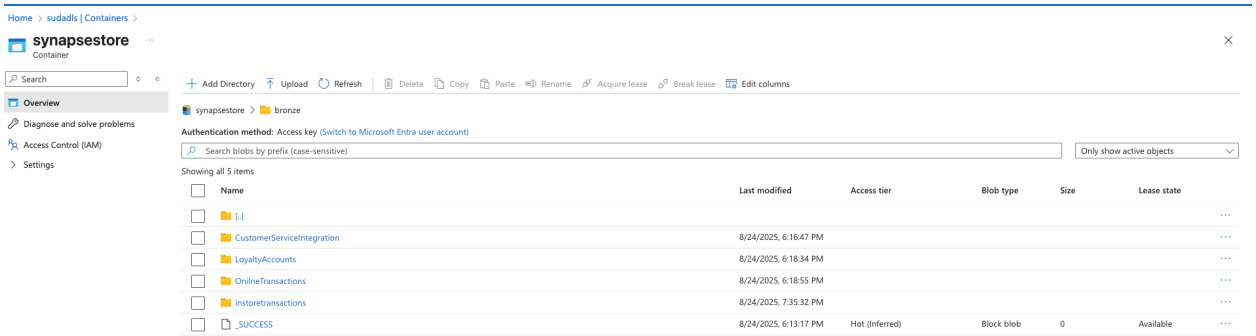


Implementation screenshots

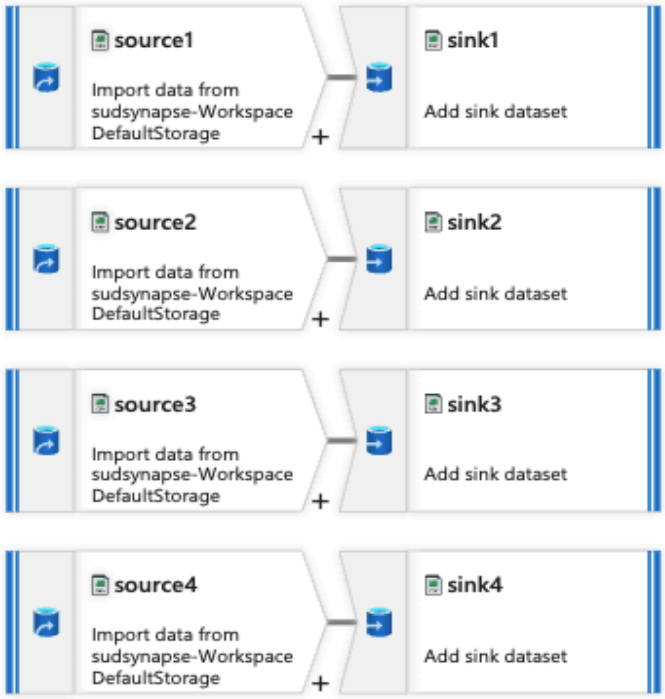
Creating Medalian directories and loading data into the ADLS Gen2.



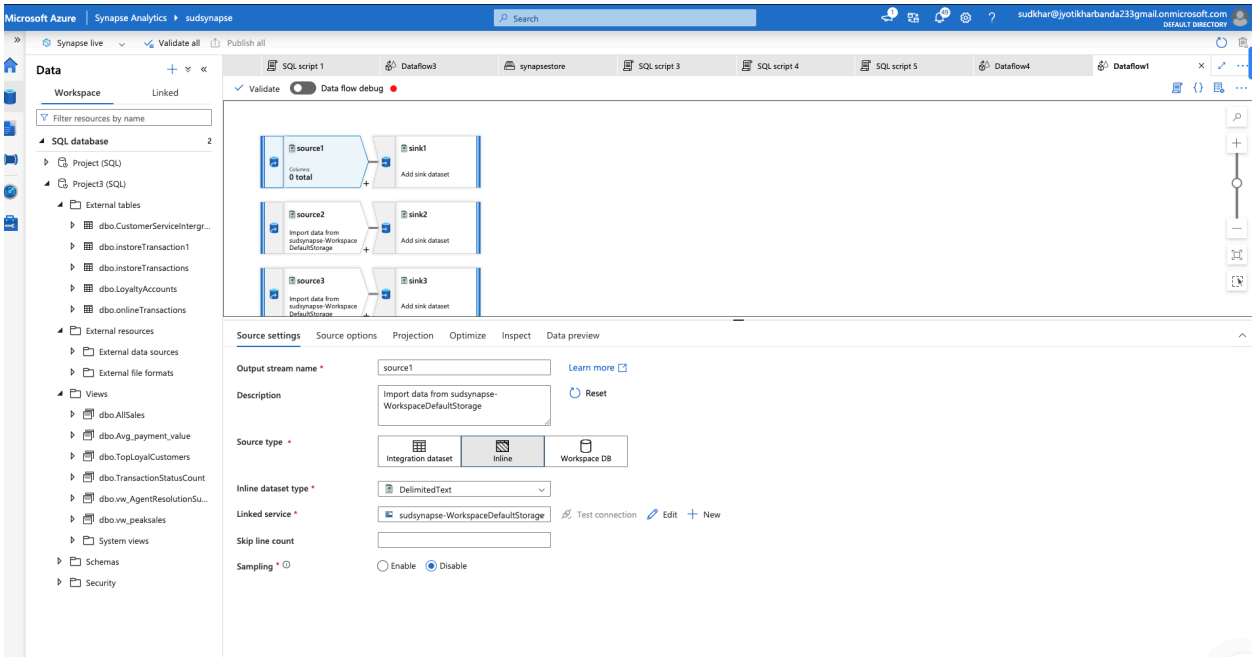
Creating folder for csv files inside the bronze layer.



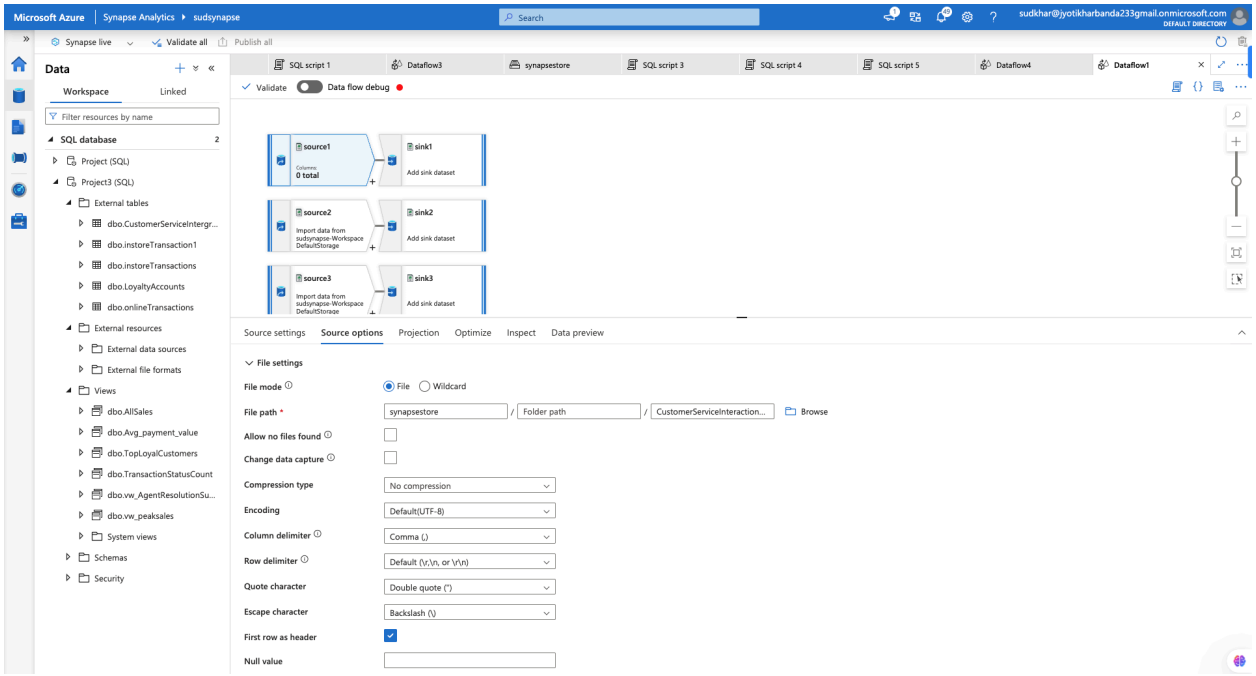
Here is the Dataflow that we will be creating for saving the file in bronze layer from adls gen2.



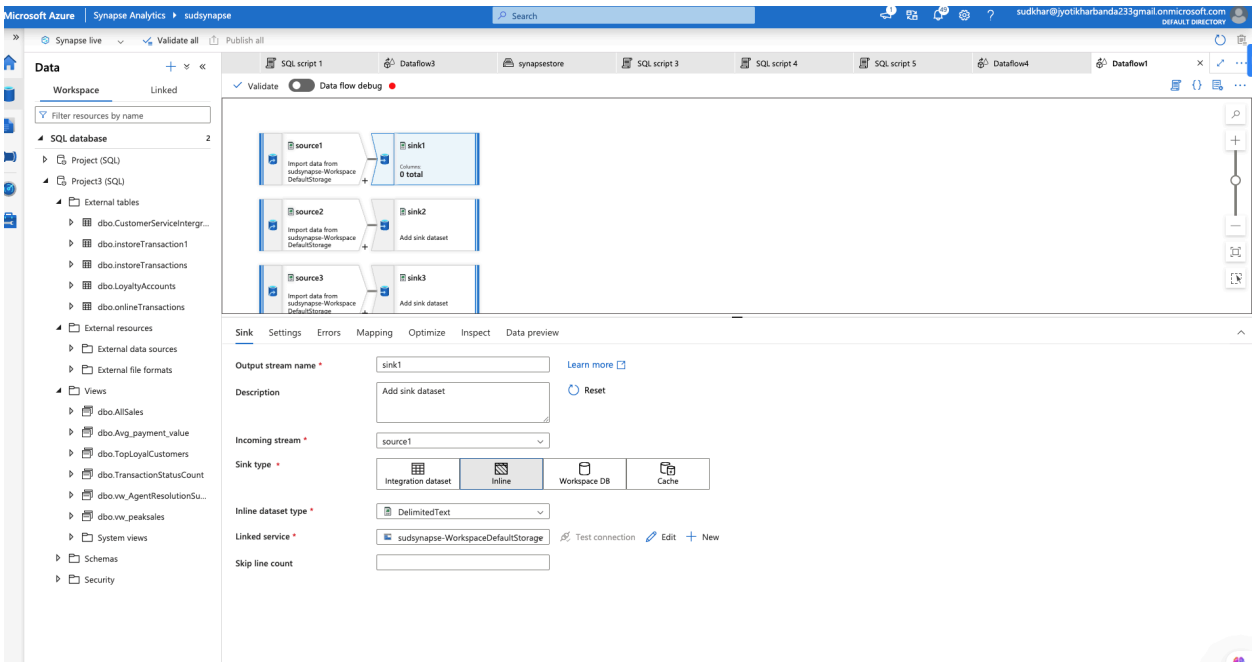
Go to source selecting the input data type.



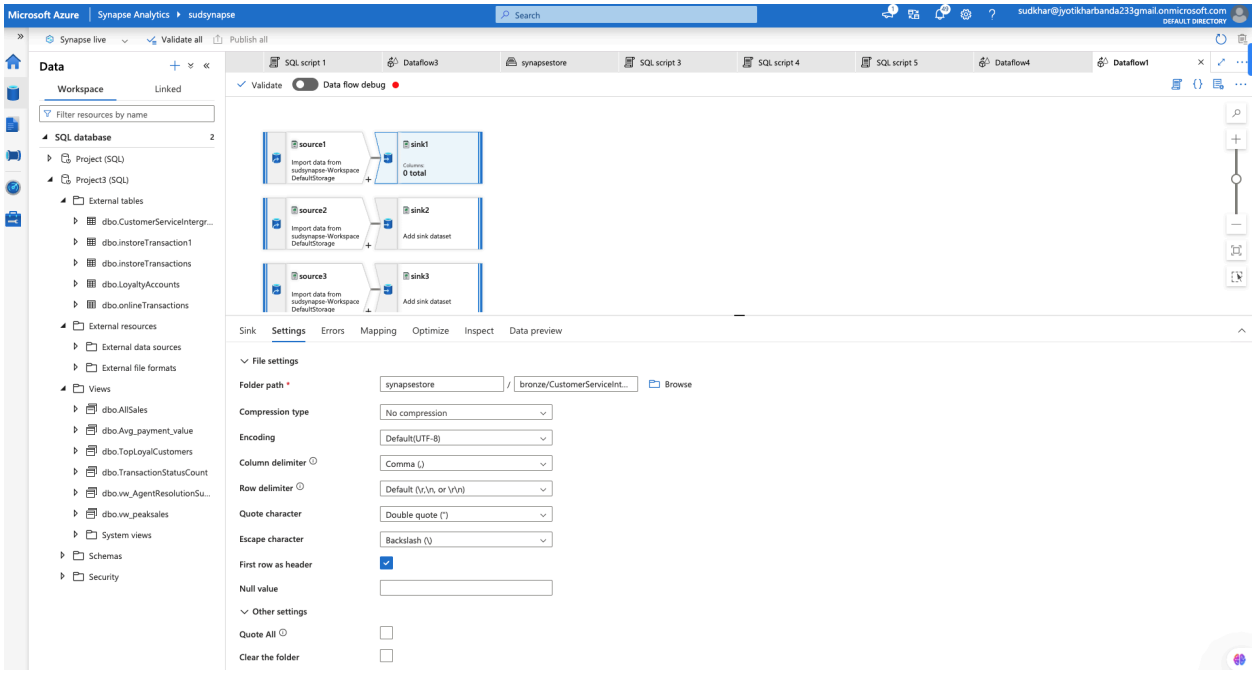
Provide the file path for the source



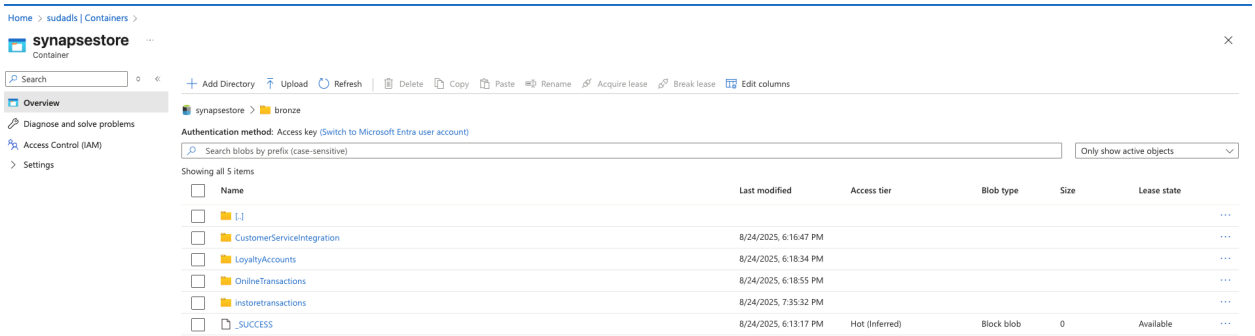
Select the datatype in which data has been stored.



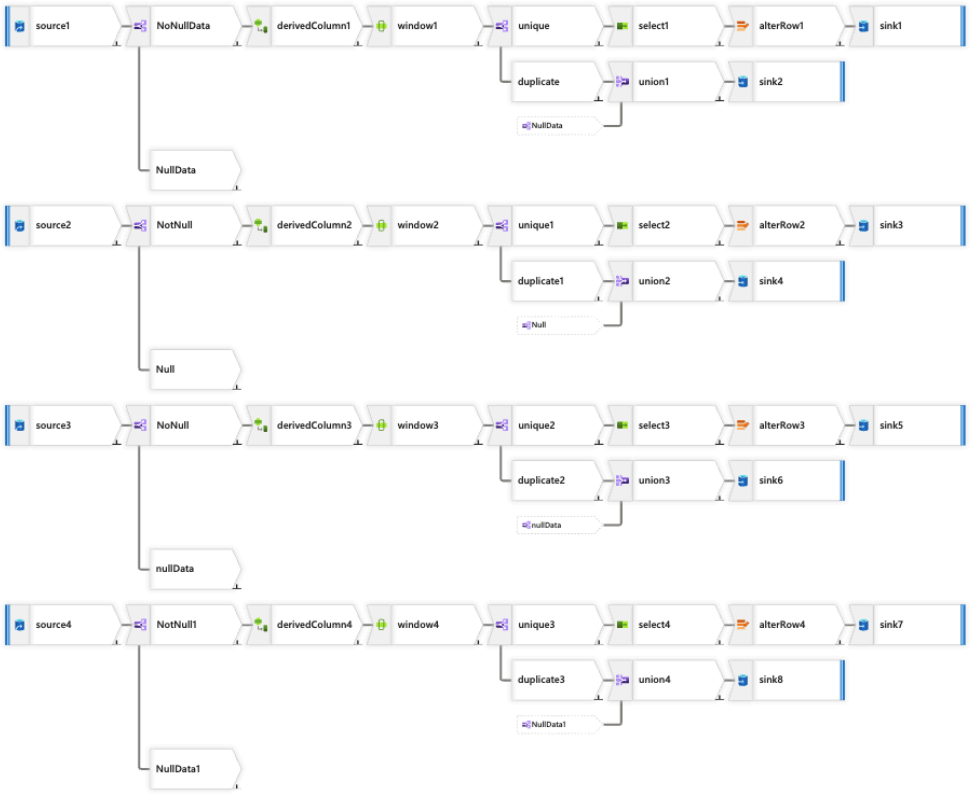
Provide the file path where data has to stored in the bronze layer.



Here is the location where data has been stored.

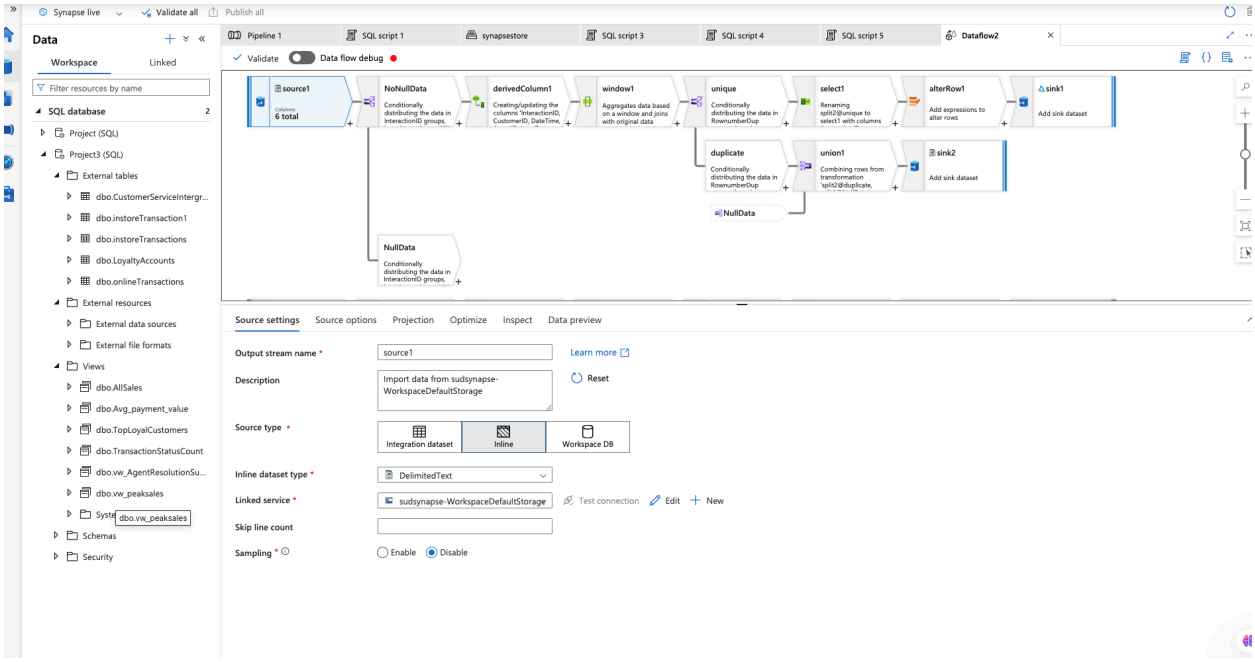


Data Flow for Data cleaning

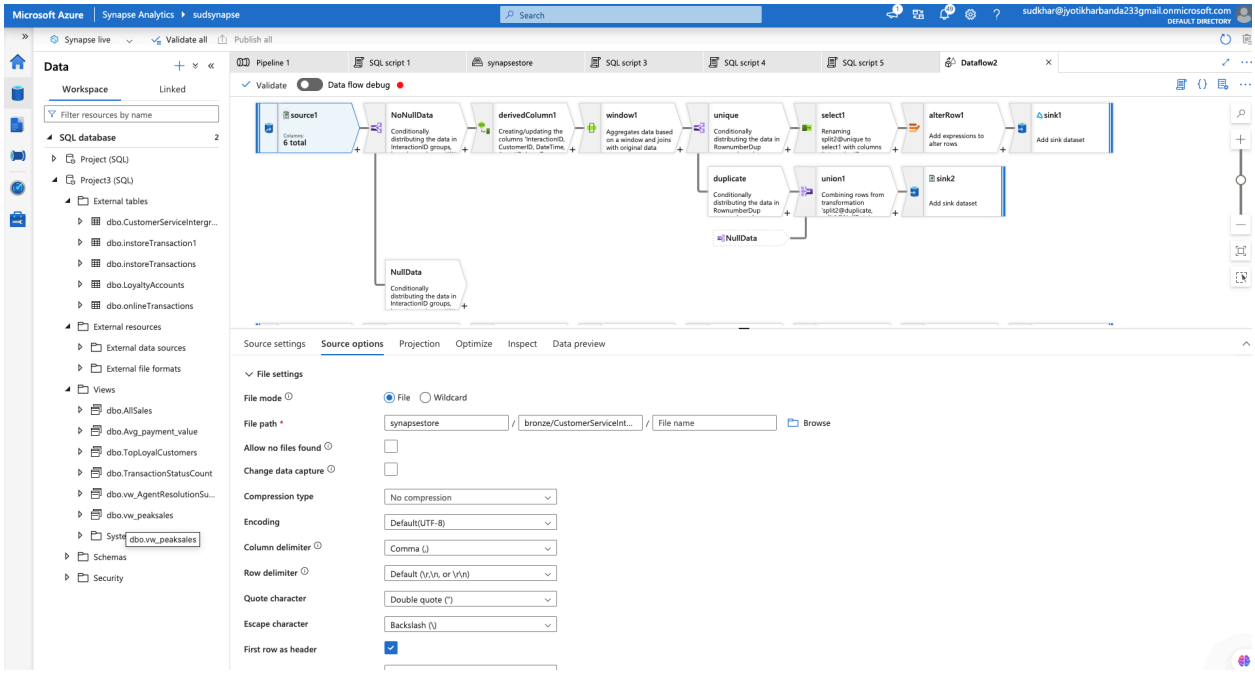


These are 4 dataflows where 4 csv files has been loaded and cleaned into silver layer in delta table format.

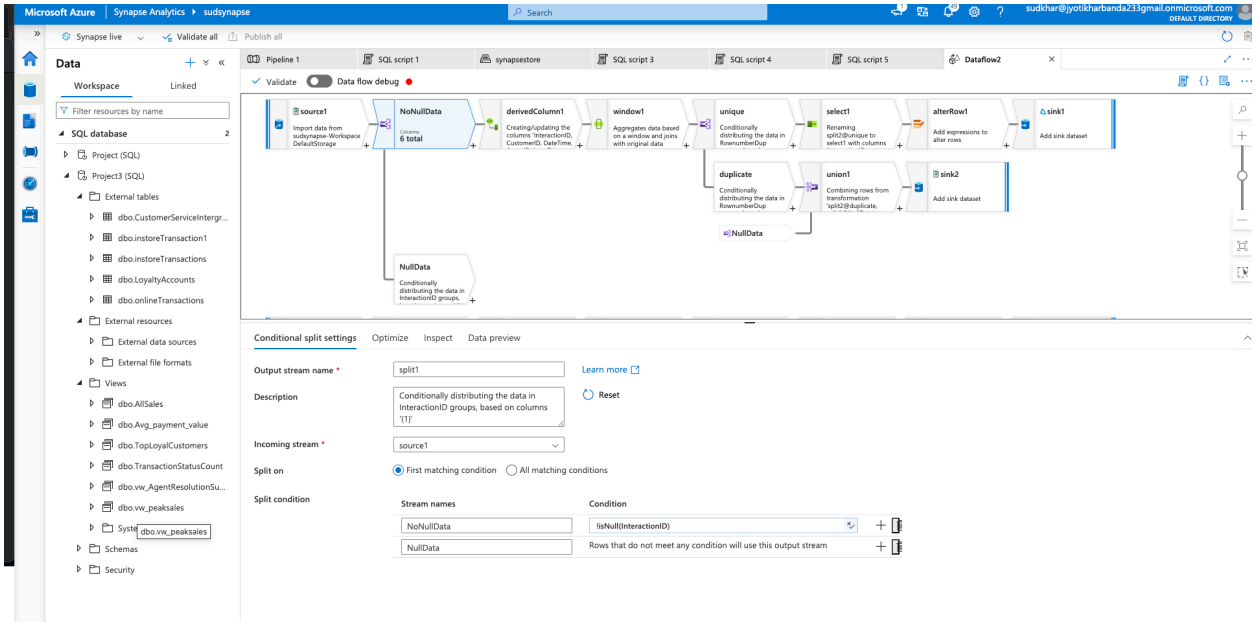
Source activity selecting the dataset type.



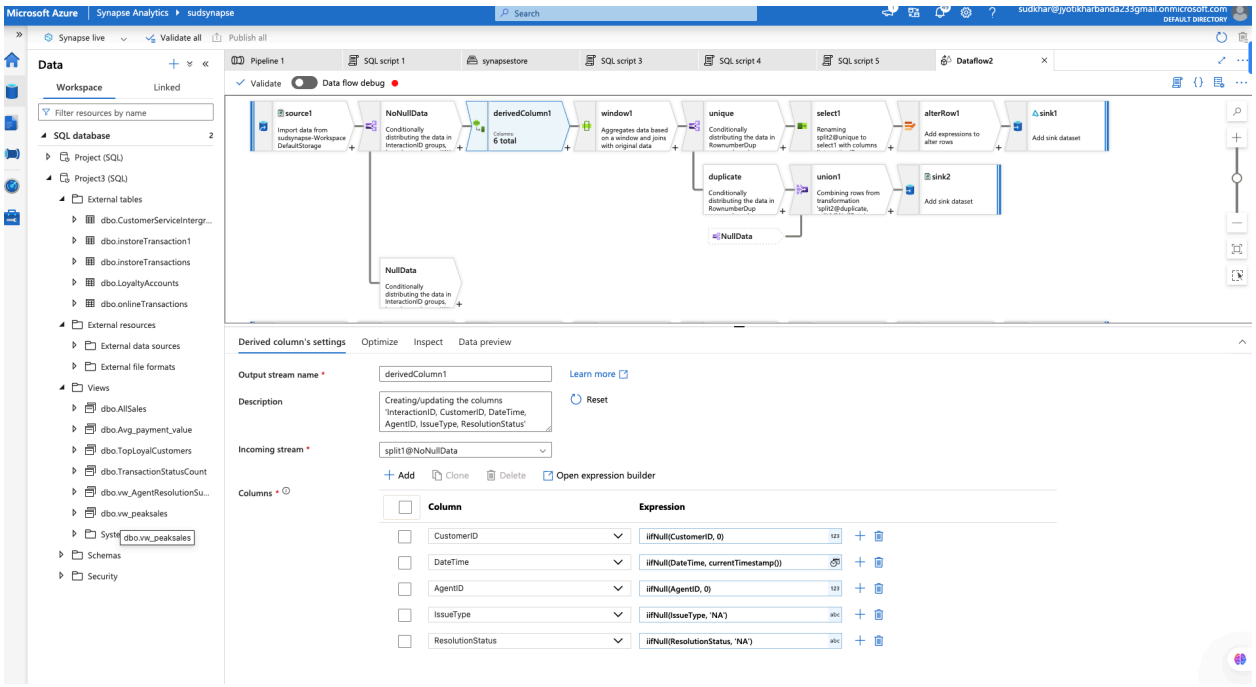
Providing the path from where data has been loaded.



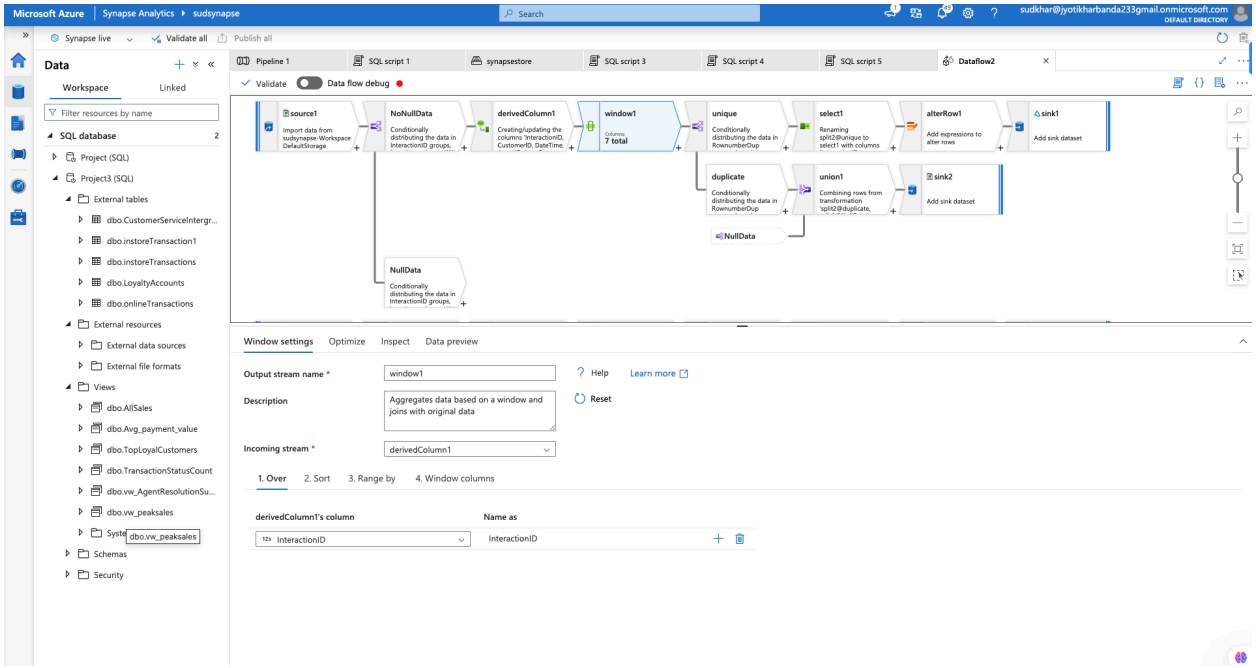
Adding conditional split for removing all the rows with null primary column ie. Interaction ID for this file.



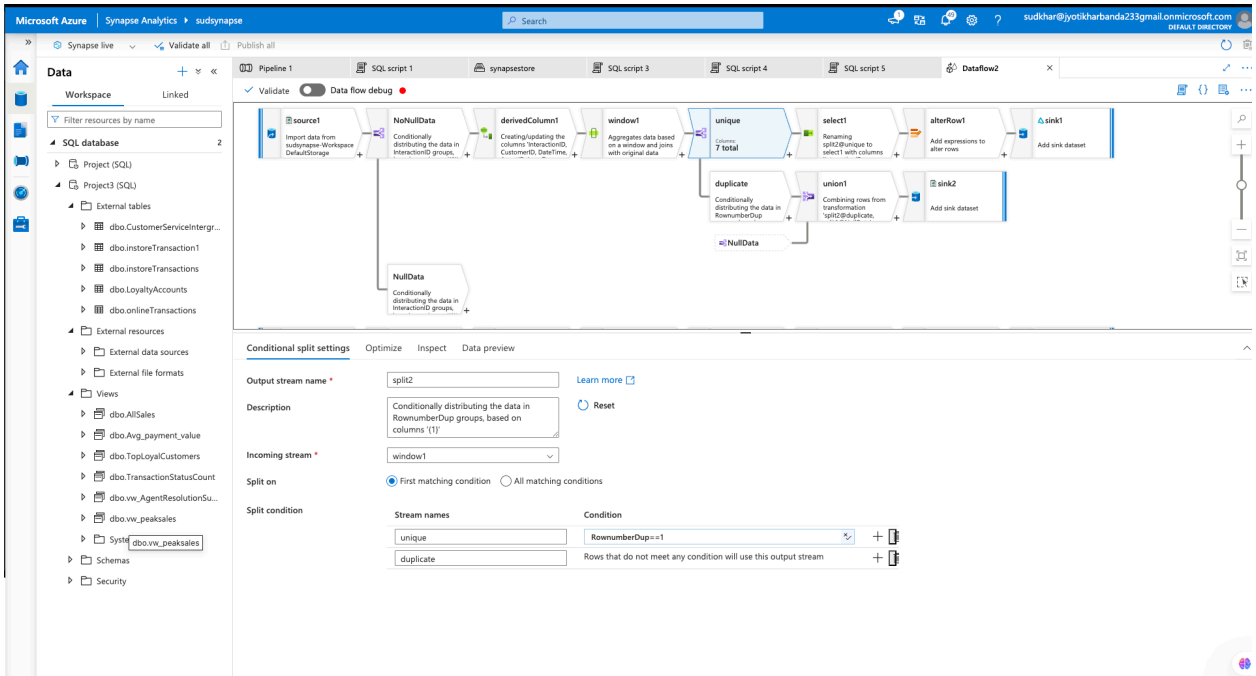
Replacing all the nulls with some specific values for all the other columns.



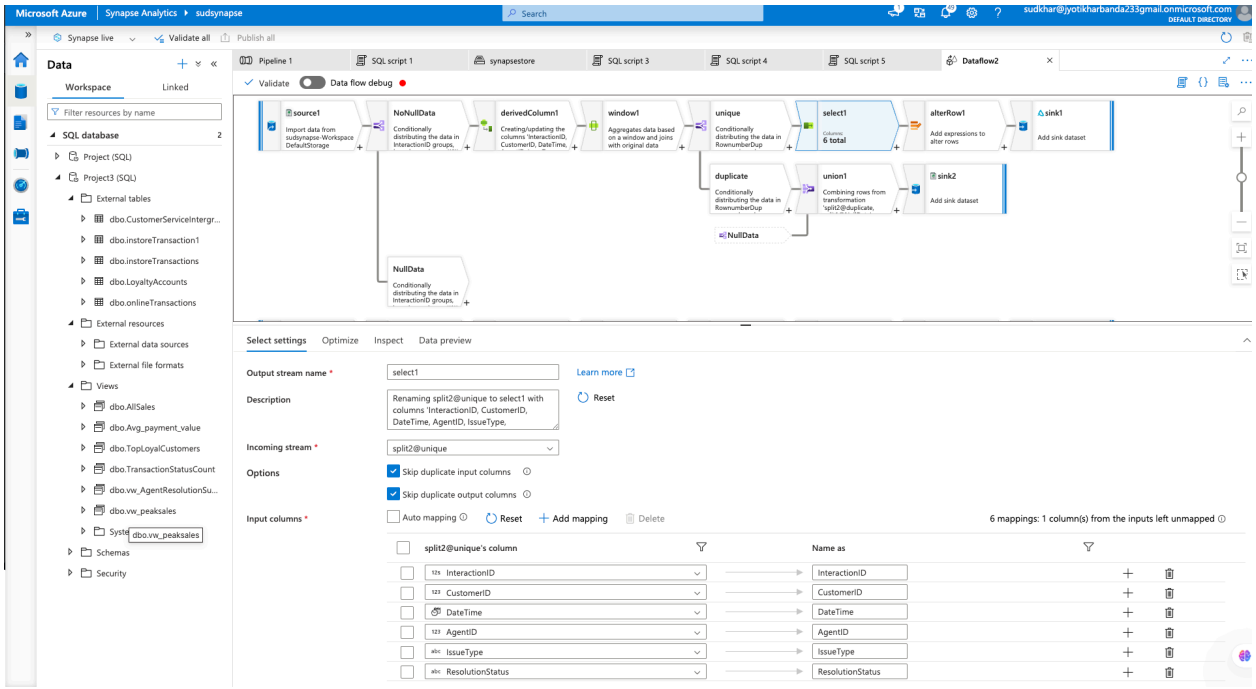
Using window activity for marking the duplicates.



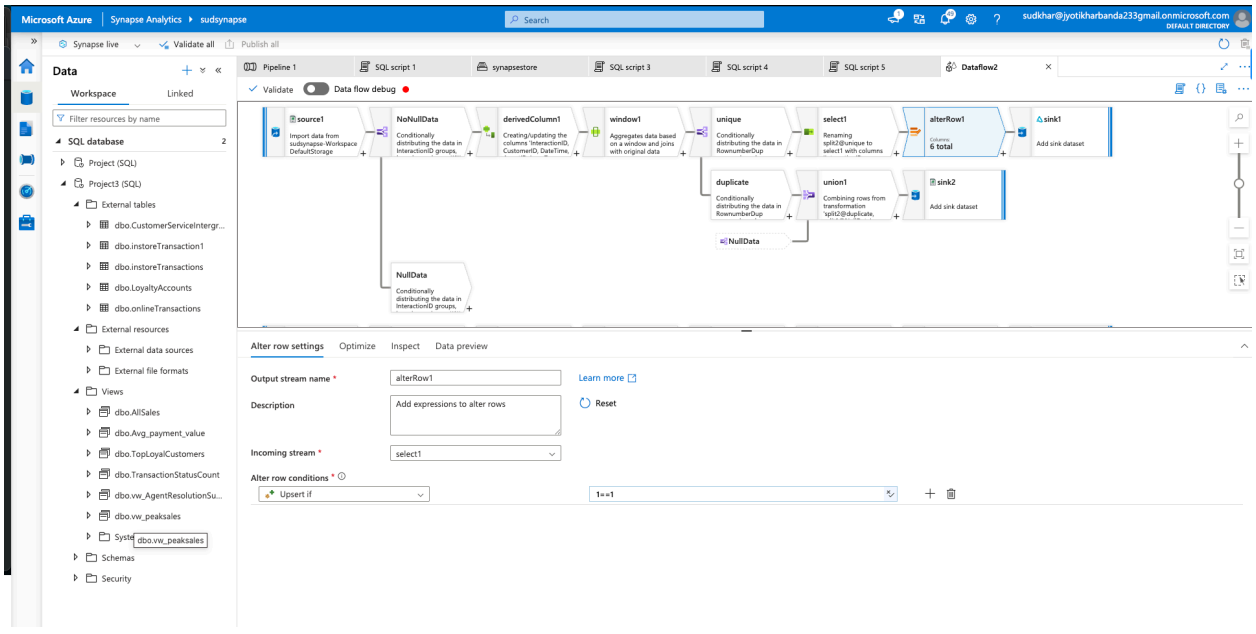
Using splitting activity for segregating unique and duplicates rows.



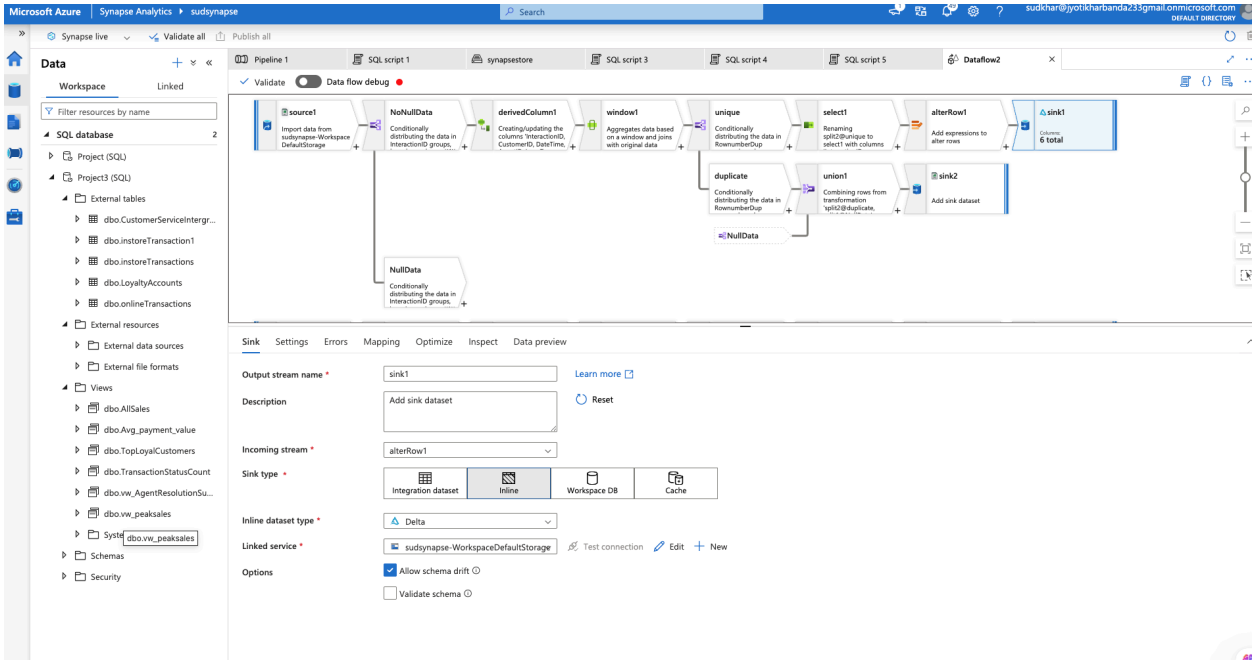
Here extra column rownumber is dropped using select activity.



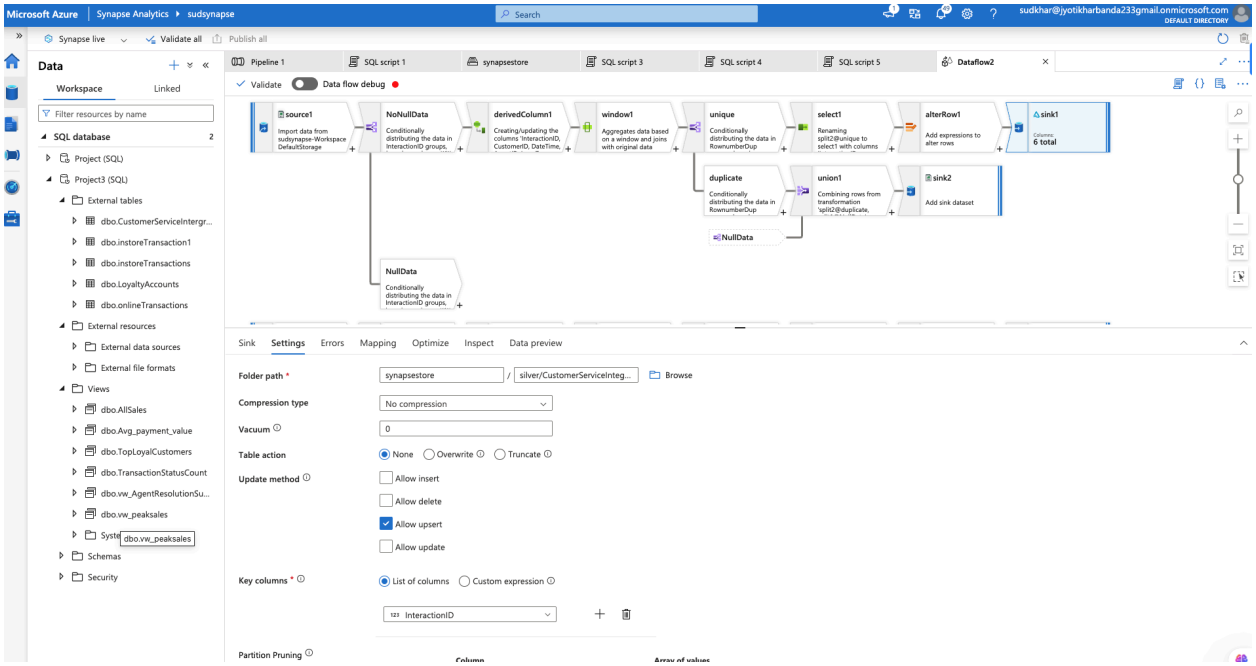
Adding alter activity for upsertimg data.



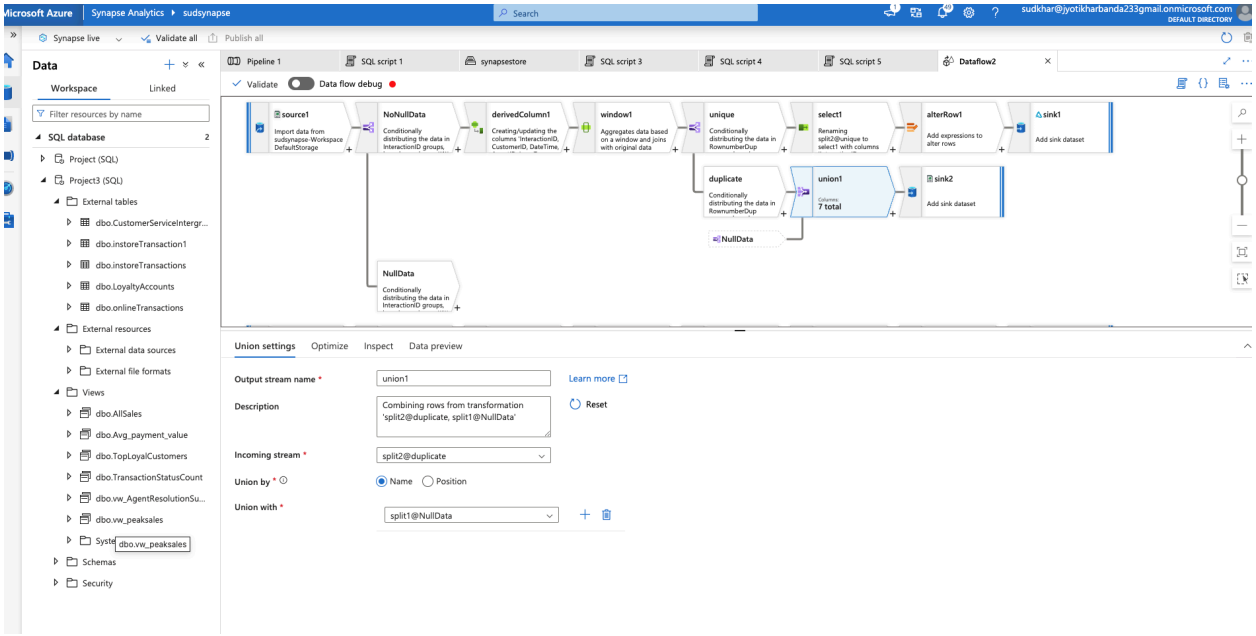
In sink activity providing dataset type as delta.



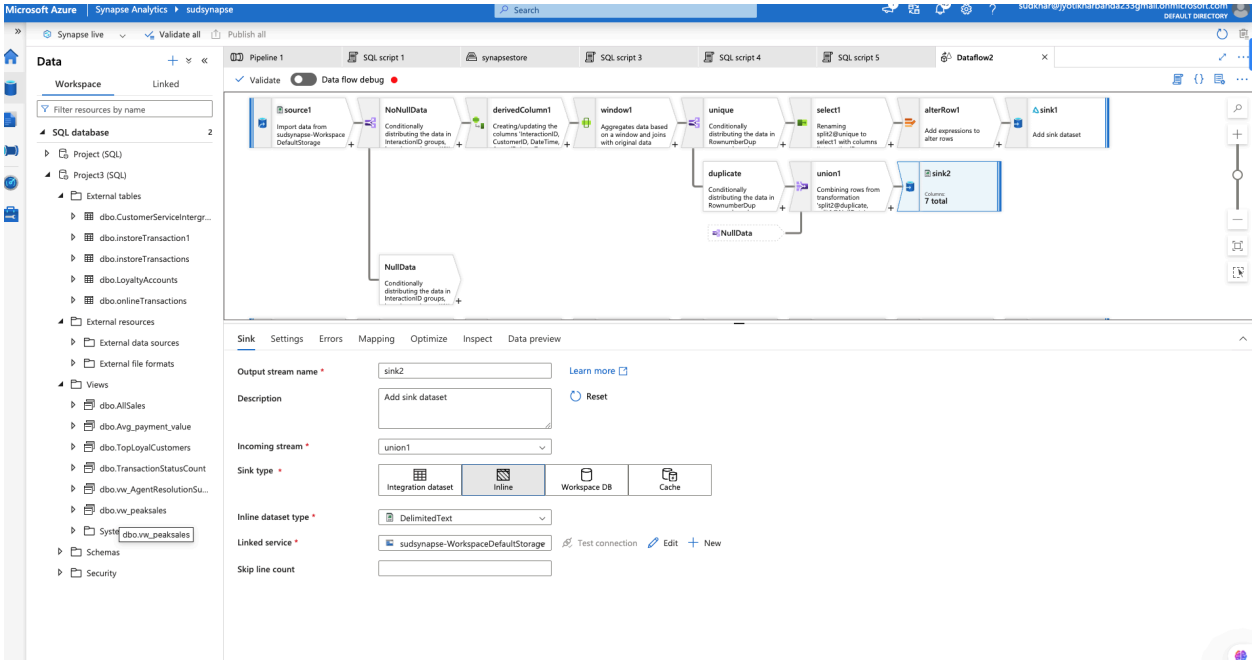
Providing the path for the silver layer.



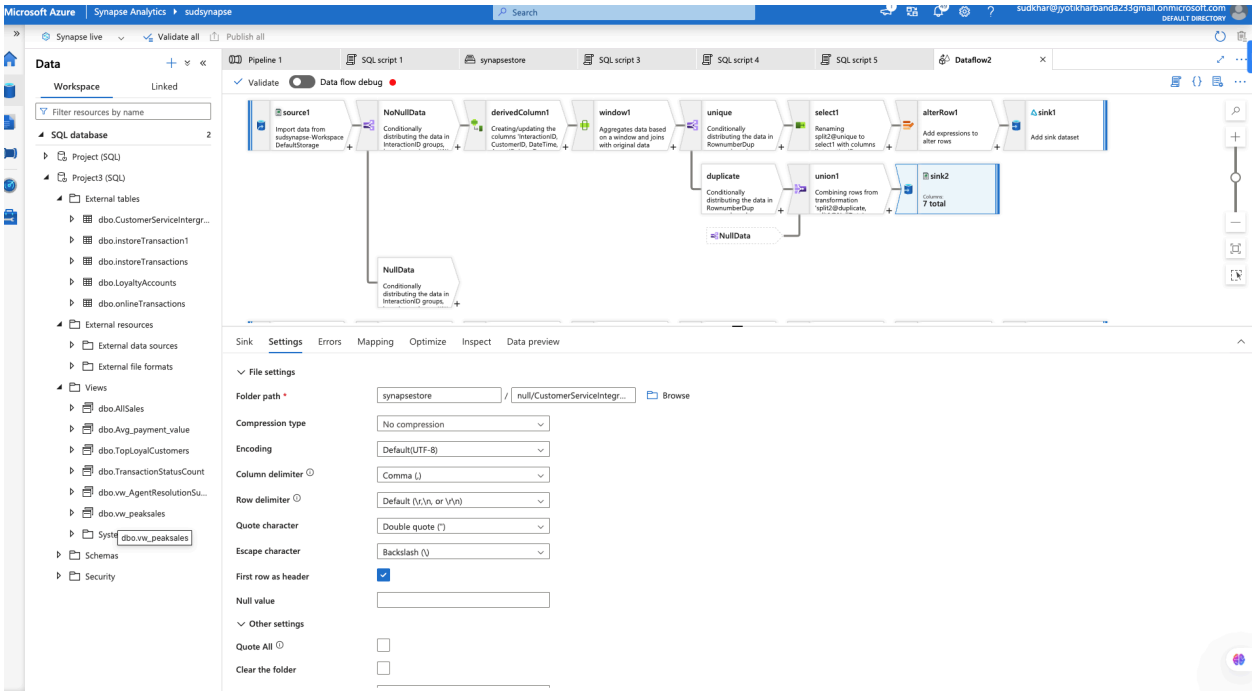
Here are adding all the duplicate and null data.



Saving all the duplicate and bull rows for sending the data back to client.

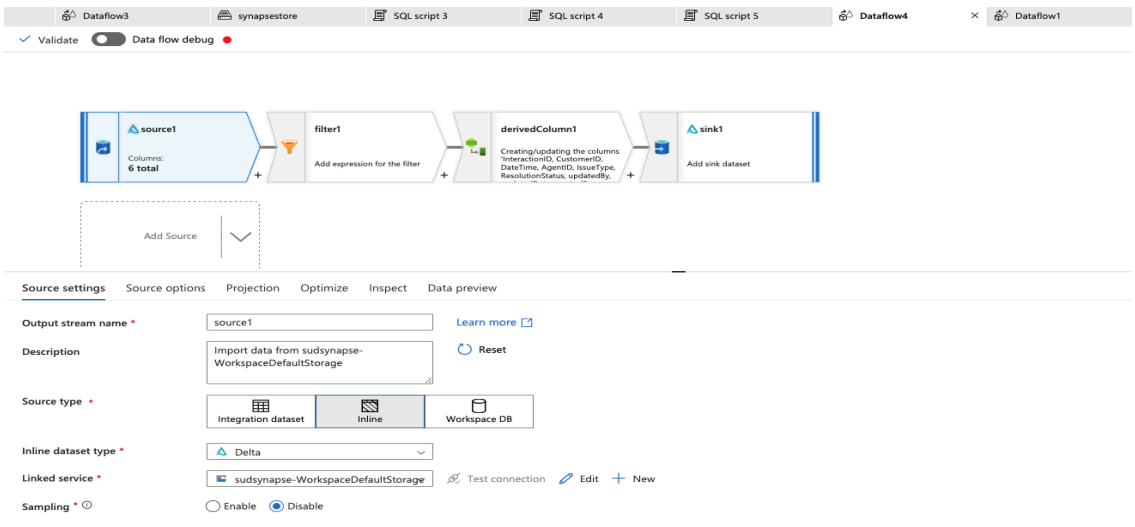


Providing the path for saving the data.

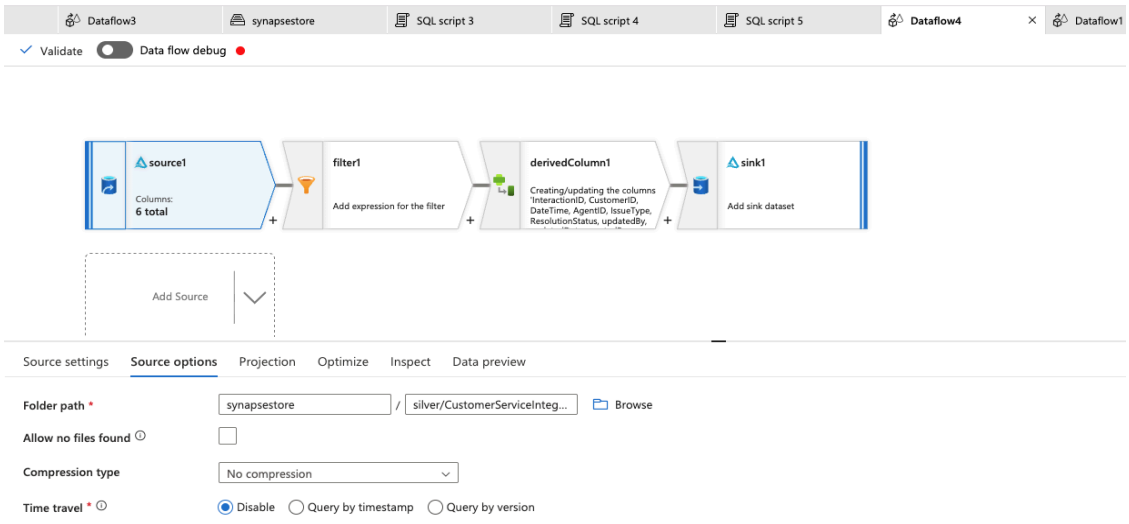


Creating Target Table dataflow

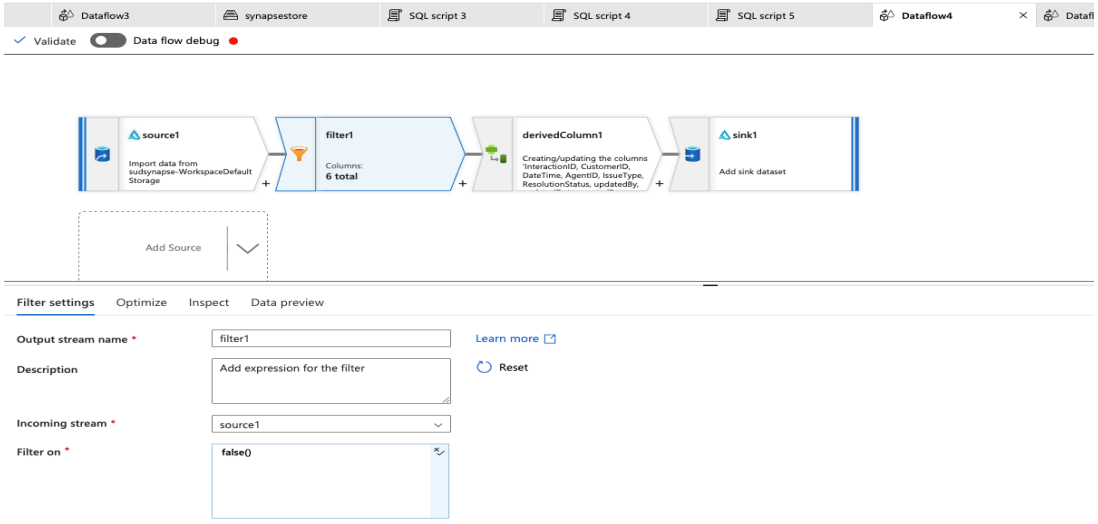
Creating a new data follow for creating the empty delta table which will be used as target while implementing SCD type-2



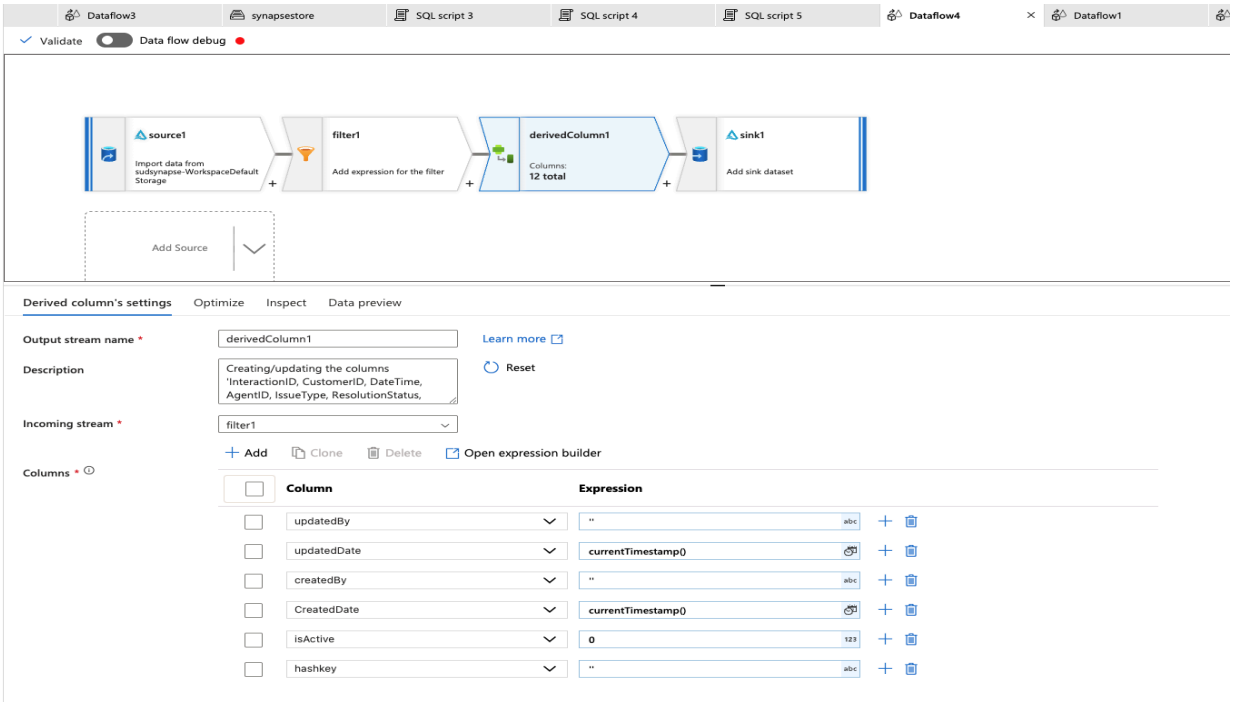
Providing the path.



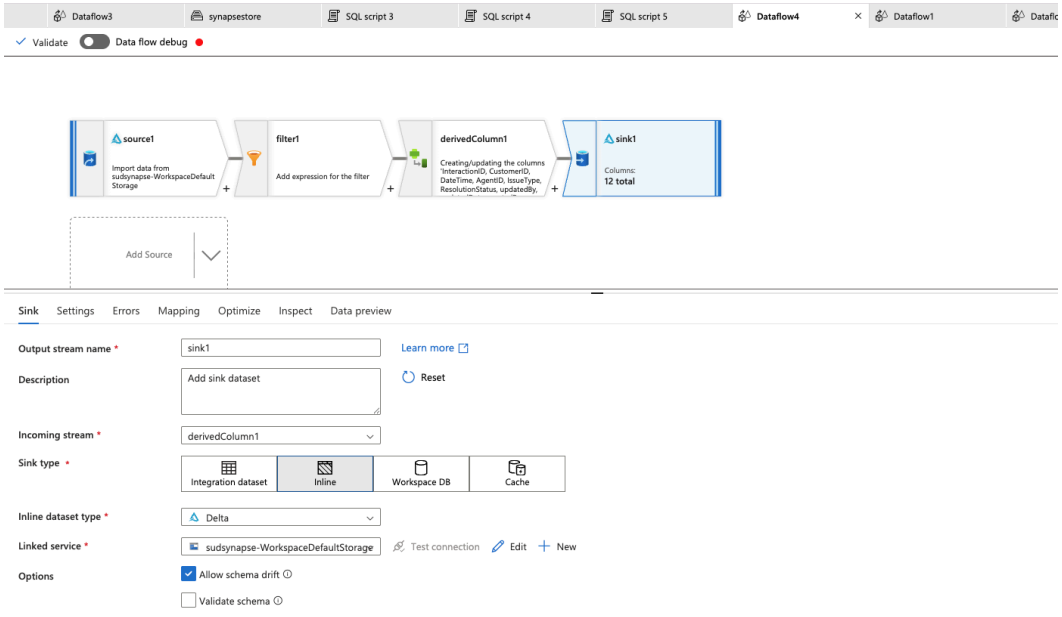
Adding the filter activity for removing all the data from the table.



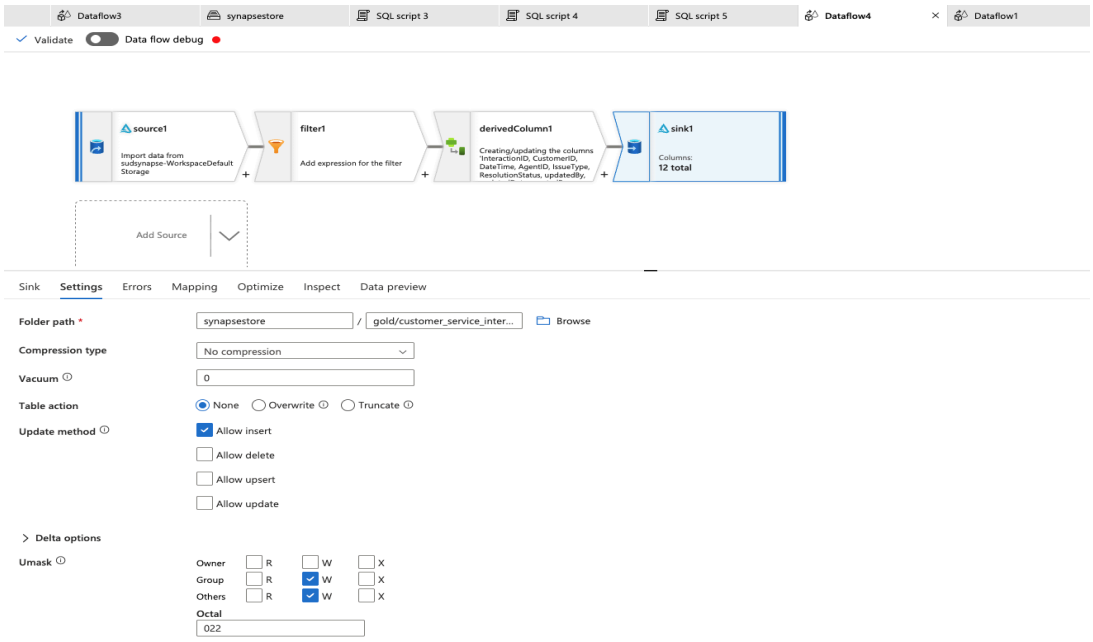
Adding the necessary column for the SCD type-2 table.



Adding sink activity for saving the empty scd type-2 table in the gold layer.

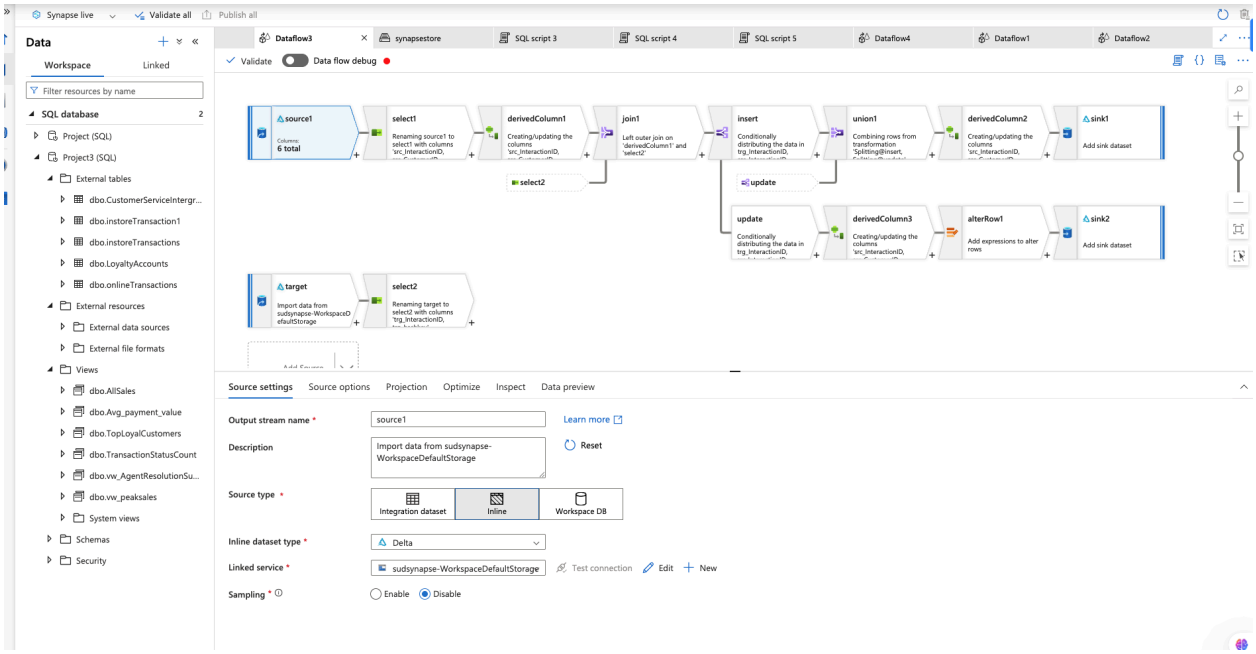


Providing the path

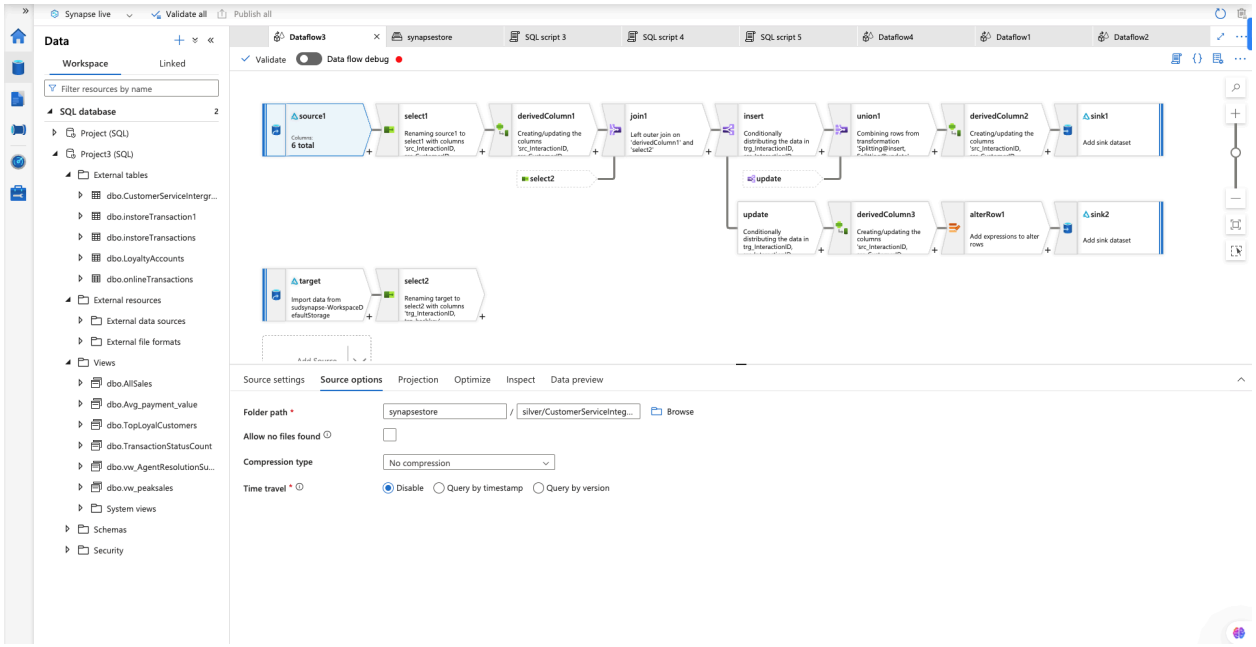


SCD Type-2 DataFlow

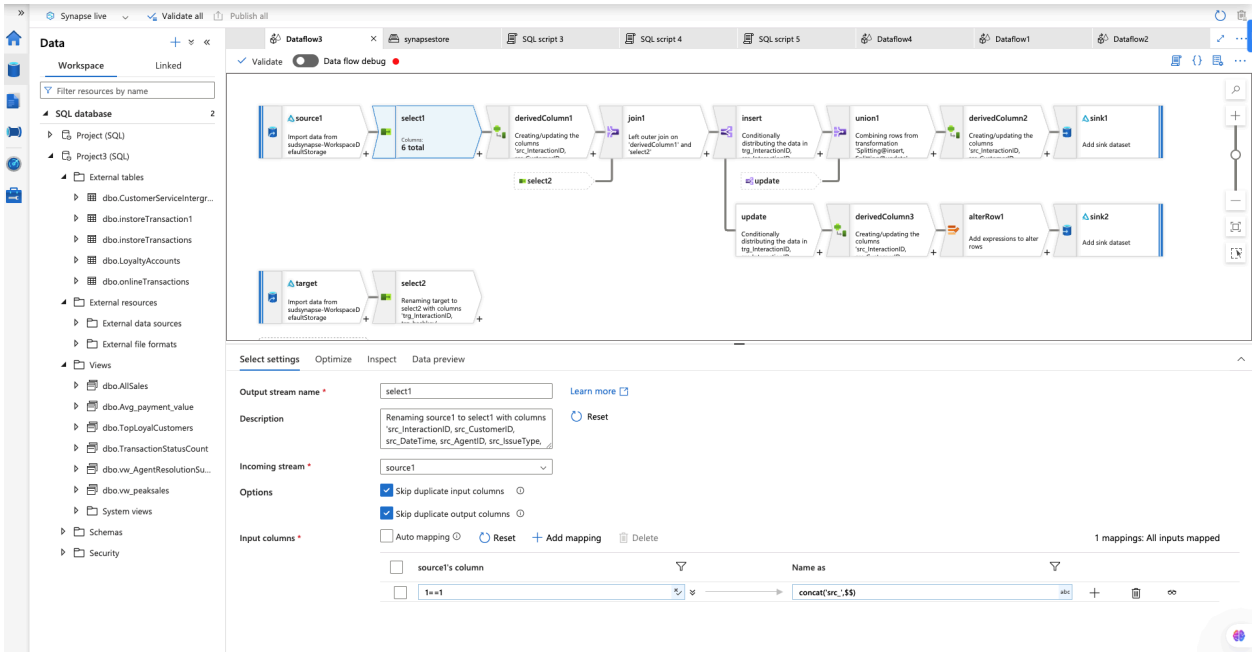
Adding source activity and selecting dataset type.



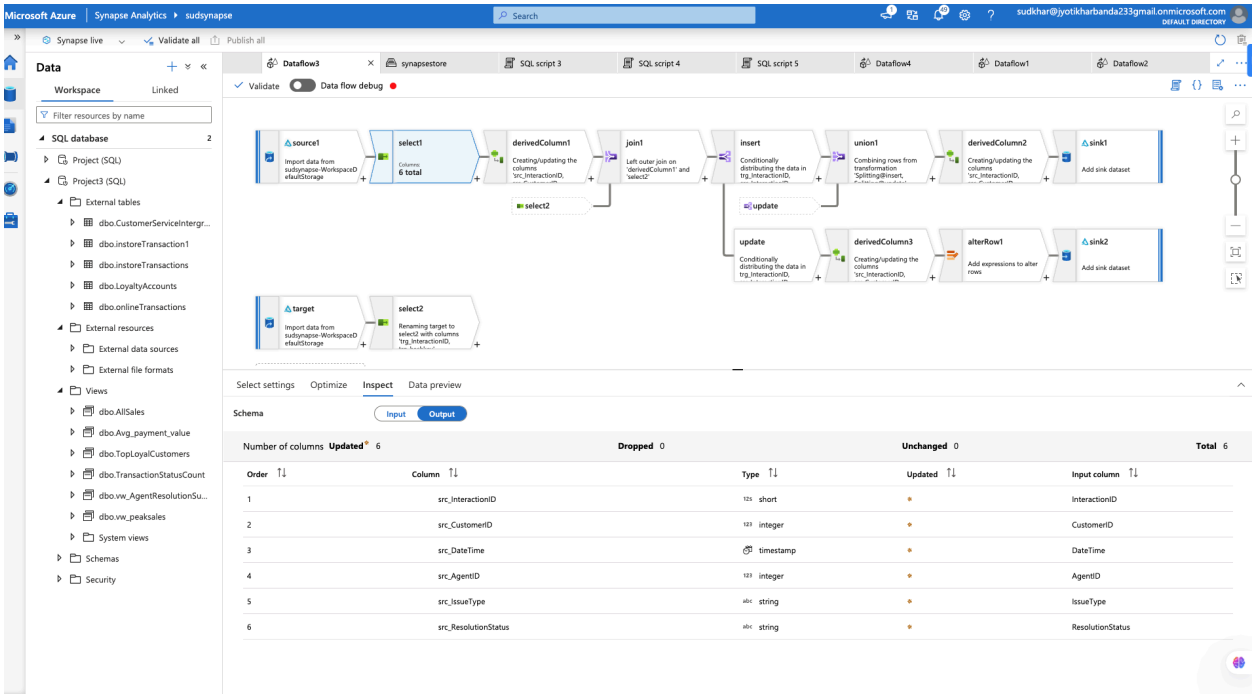
Providing the source path



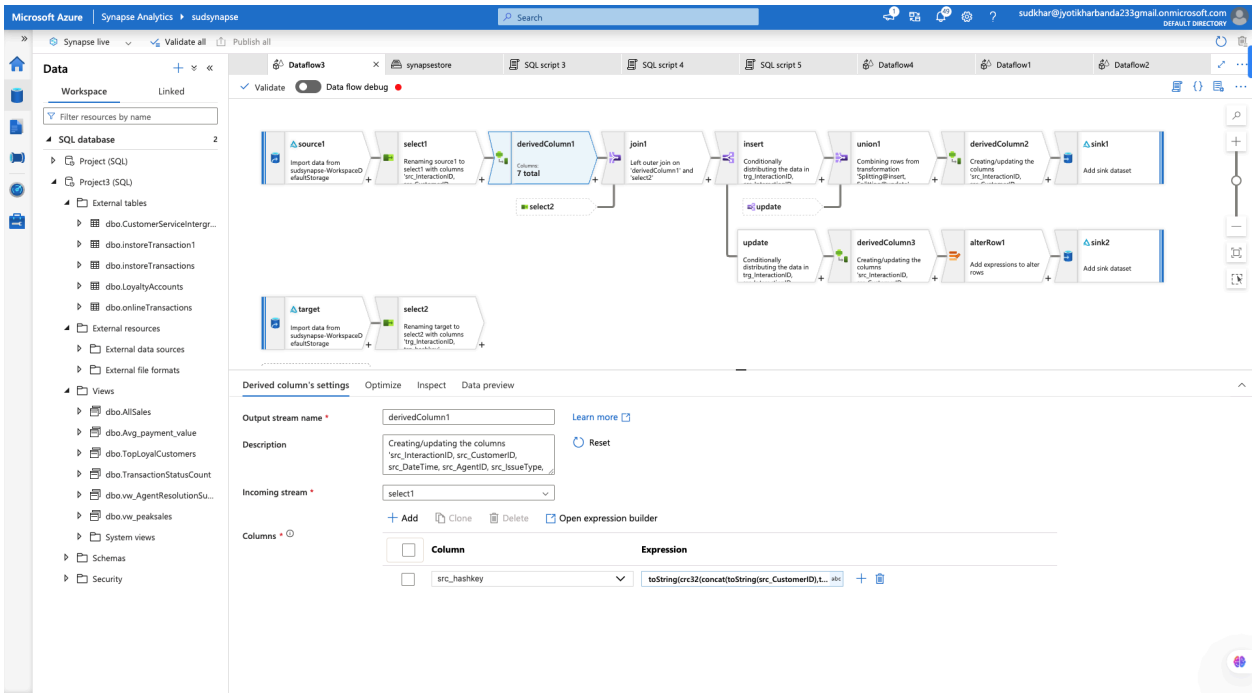
Using select activity for renaming the column.



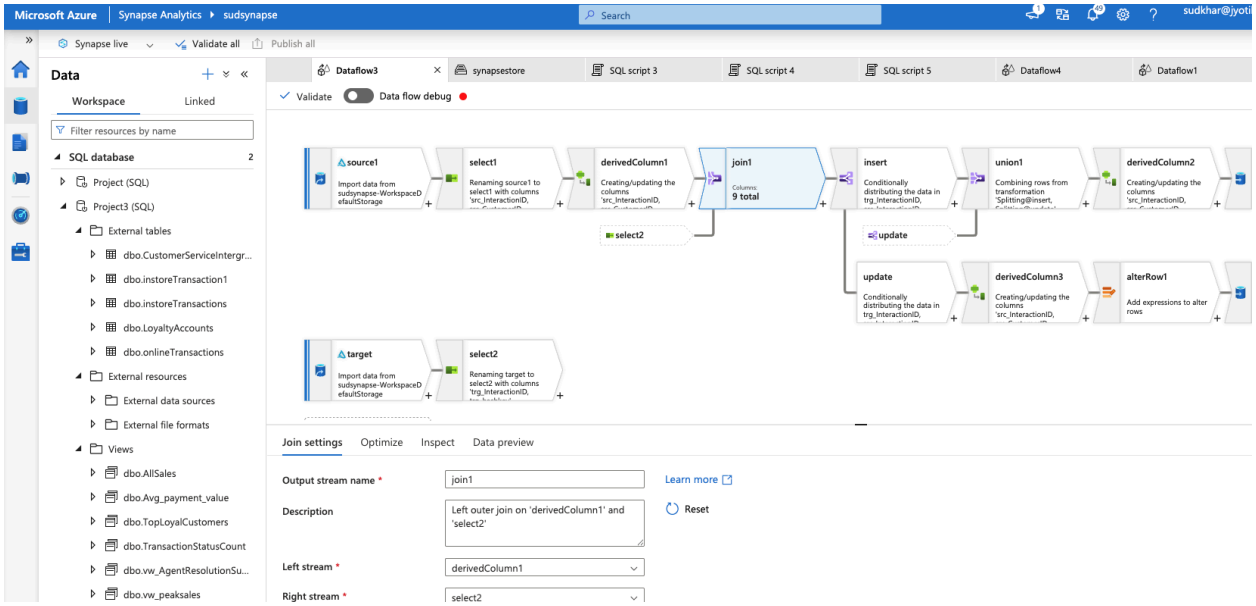
Checking whether the column has been renamed or not.



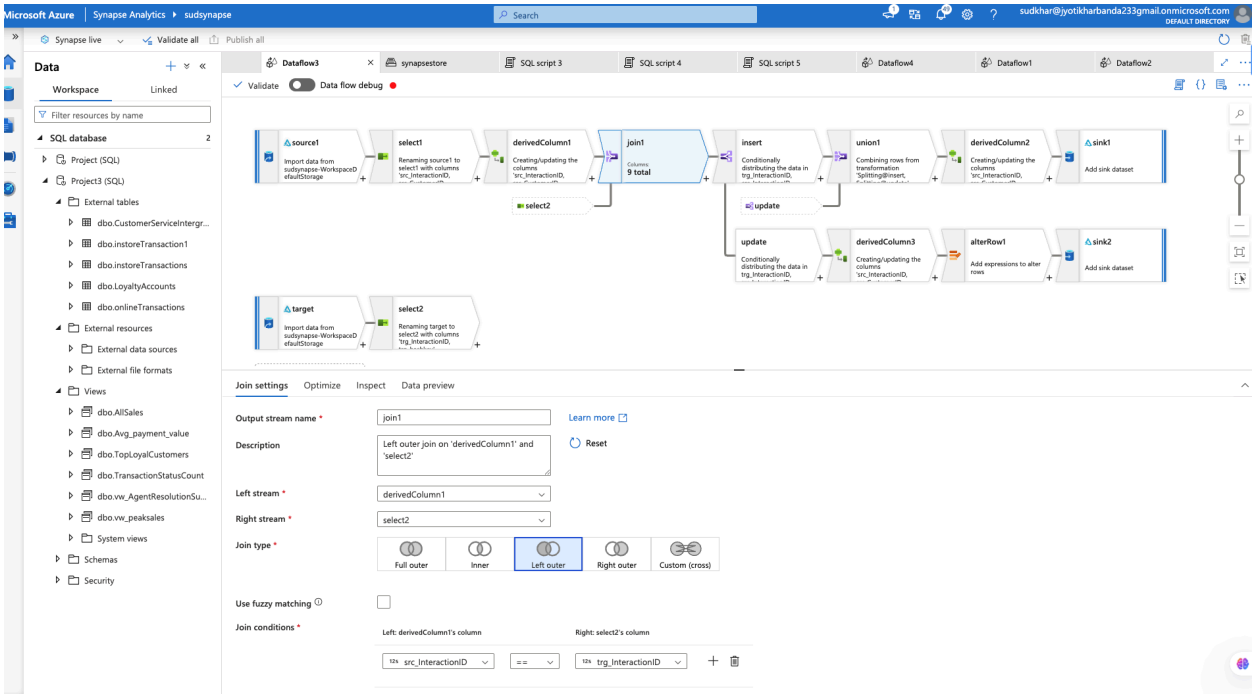
Creating a hashkey column using crc32.



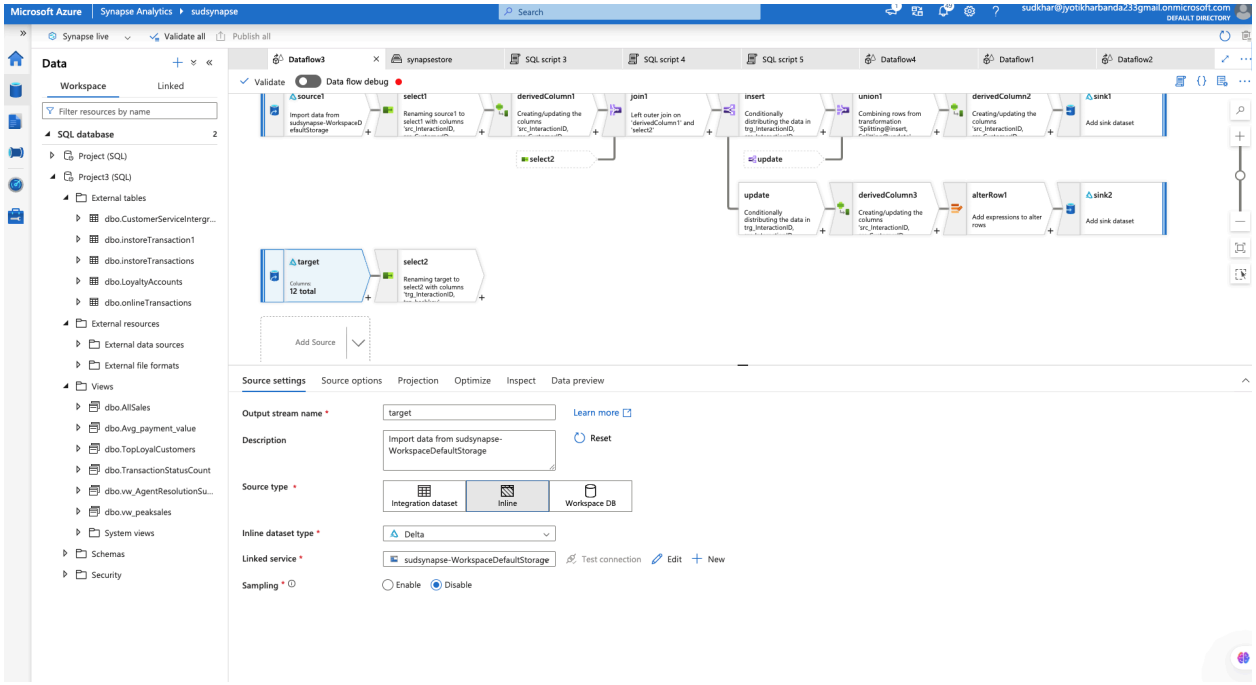
Now we are joining target table that contains only id and hashkey with the source table.



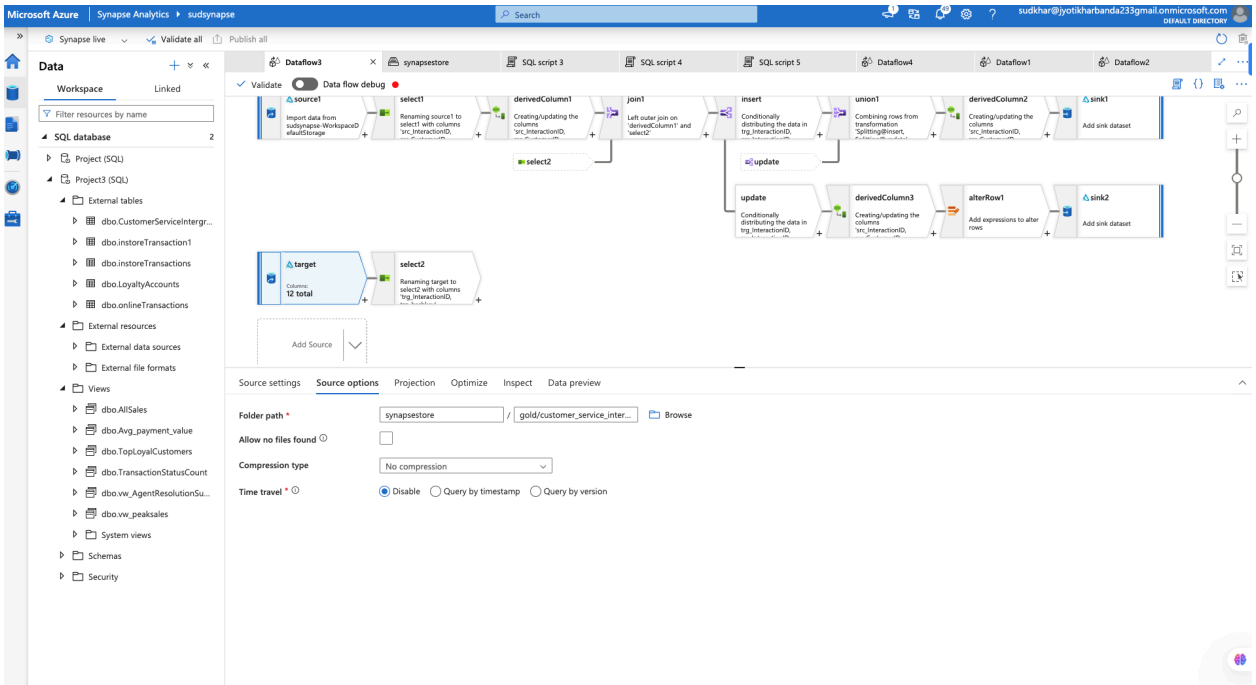
Doing a left outer join.



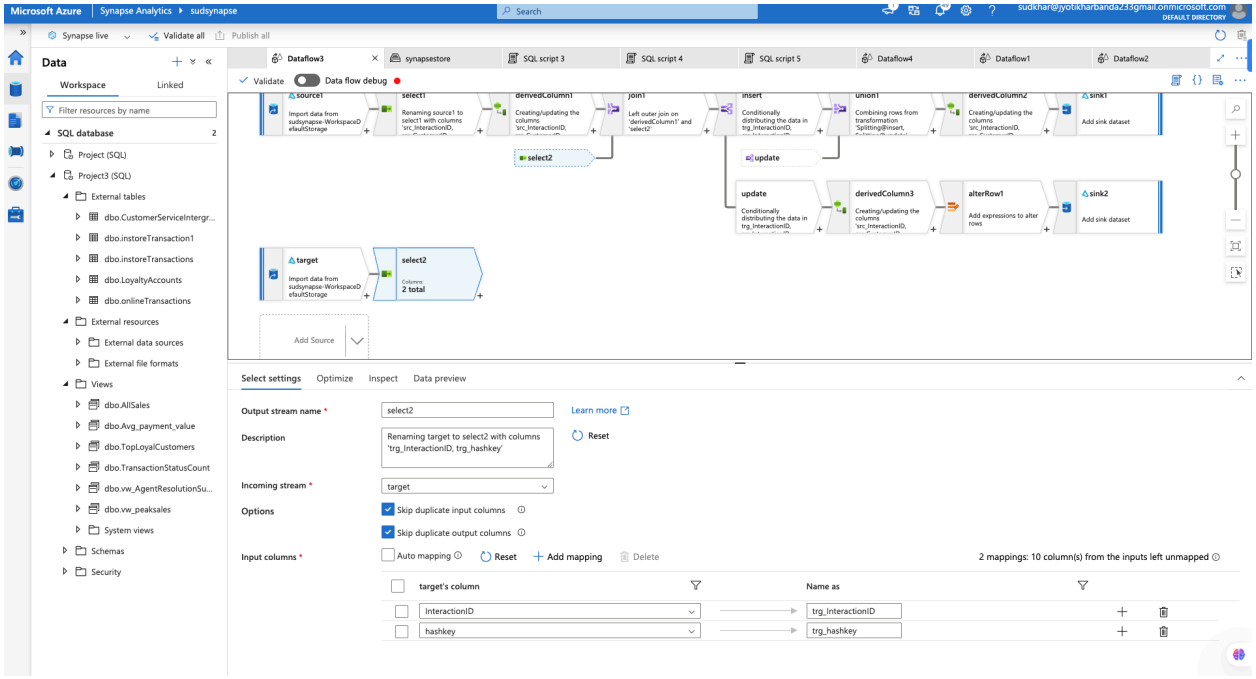
Defining target by using source activity that will scd type table from the gold layer.



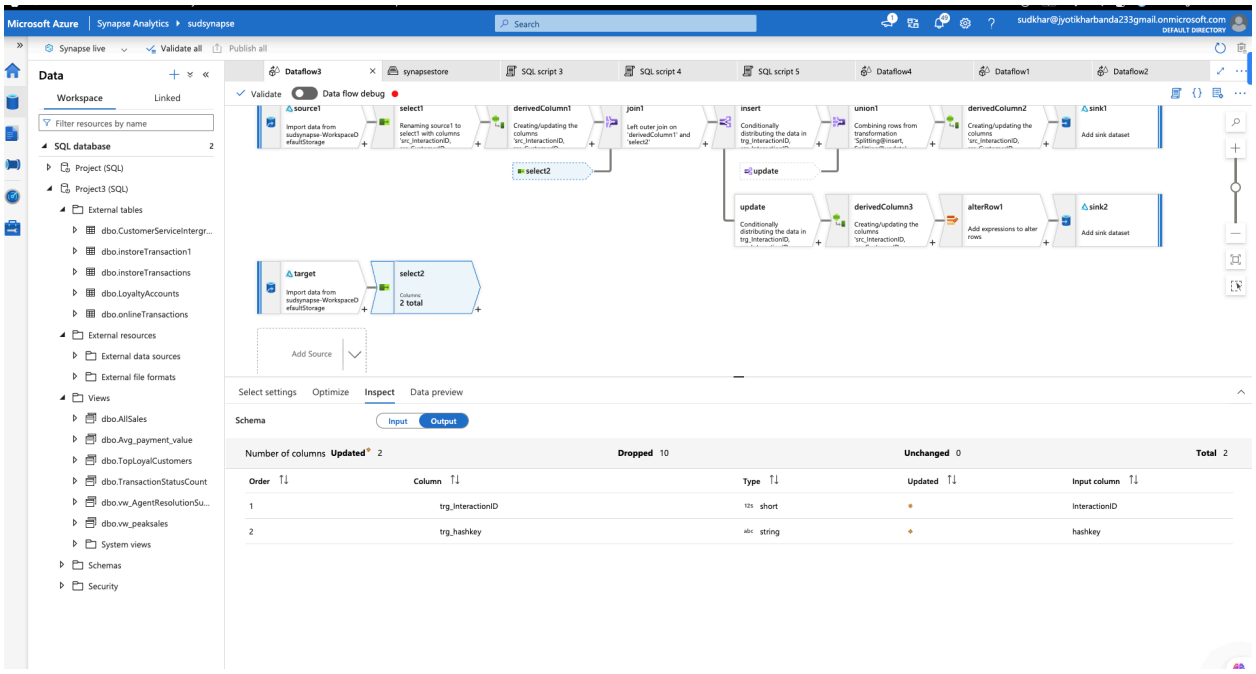
Providing the path of the table.



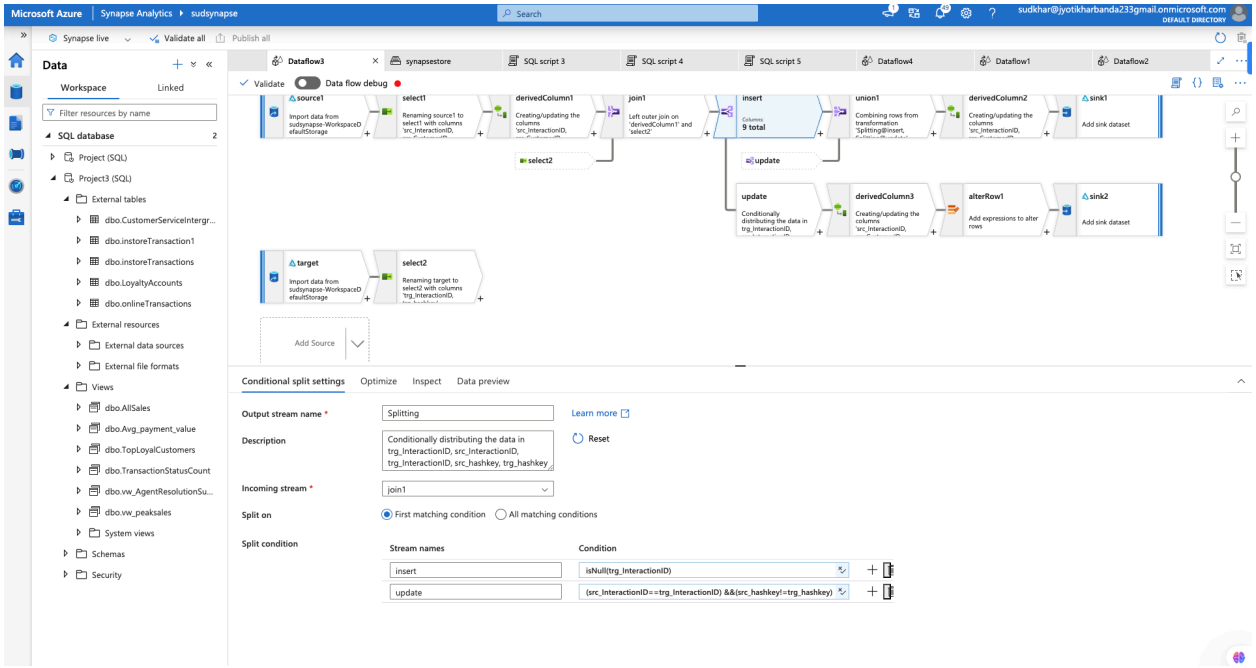
Removing the unnecessary columns from the target and adding trg in front in order to distinguish.



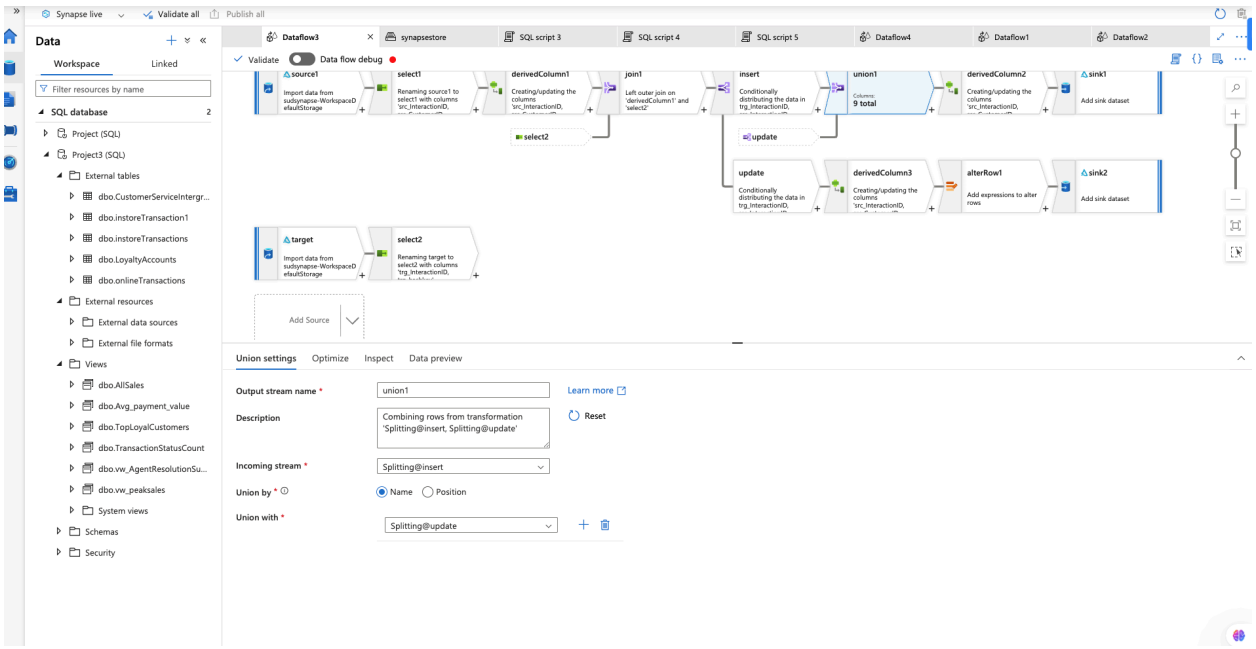
Inspecting the schema.



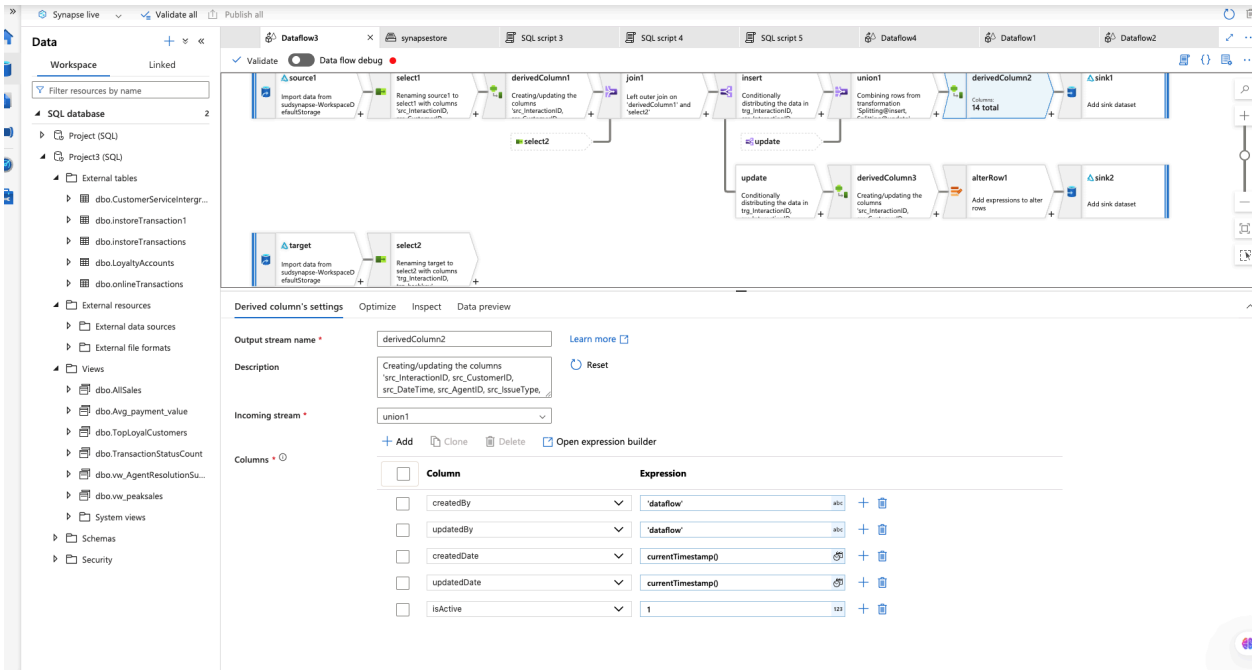
Adding a conditional split activity that we new data and updated data rows.



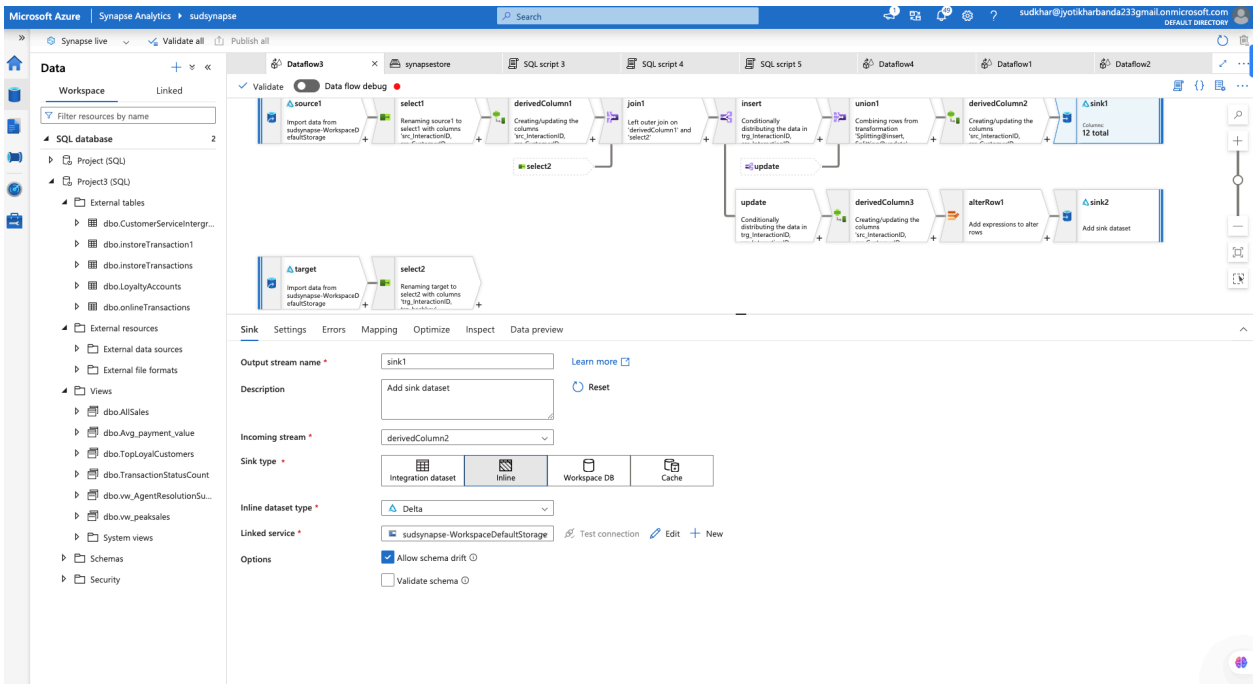
Adding union activity and will add updated rows as well we are going to insert those rows as new data.



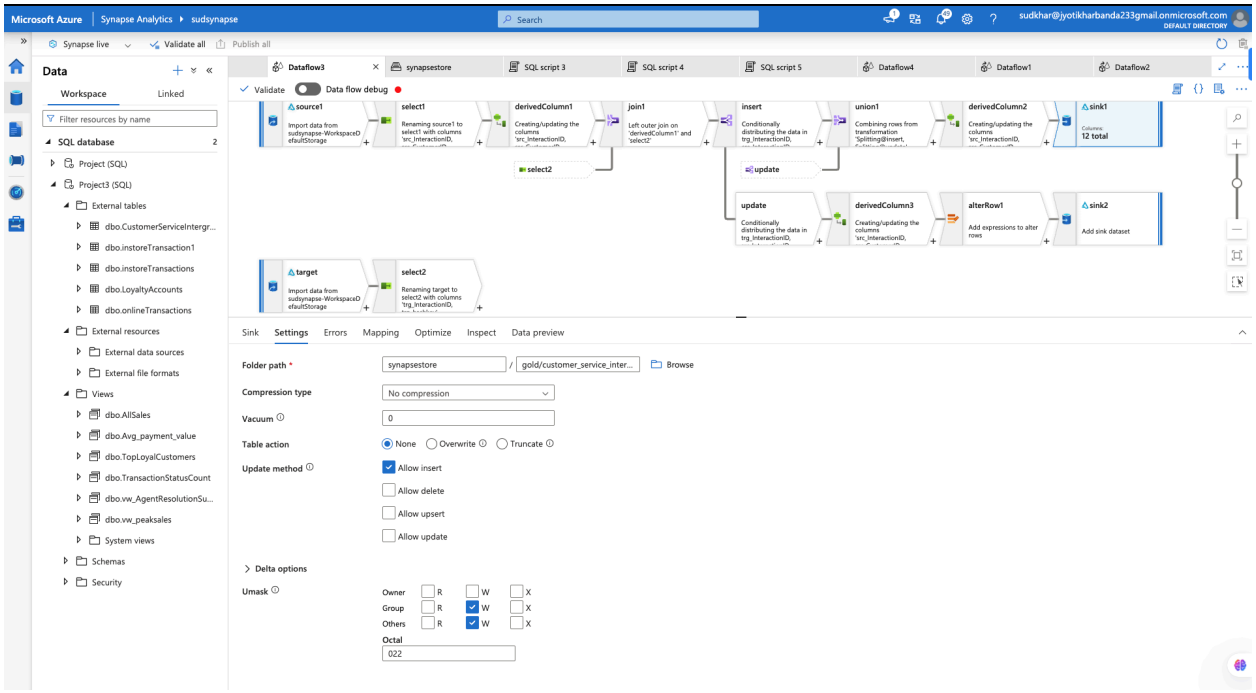
Using a derived column, we are adding scd type columns.



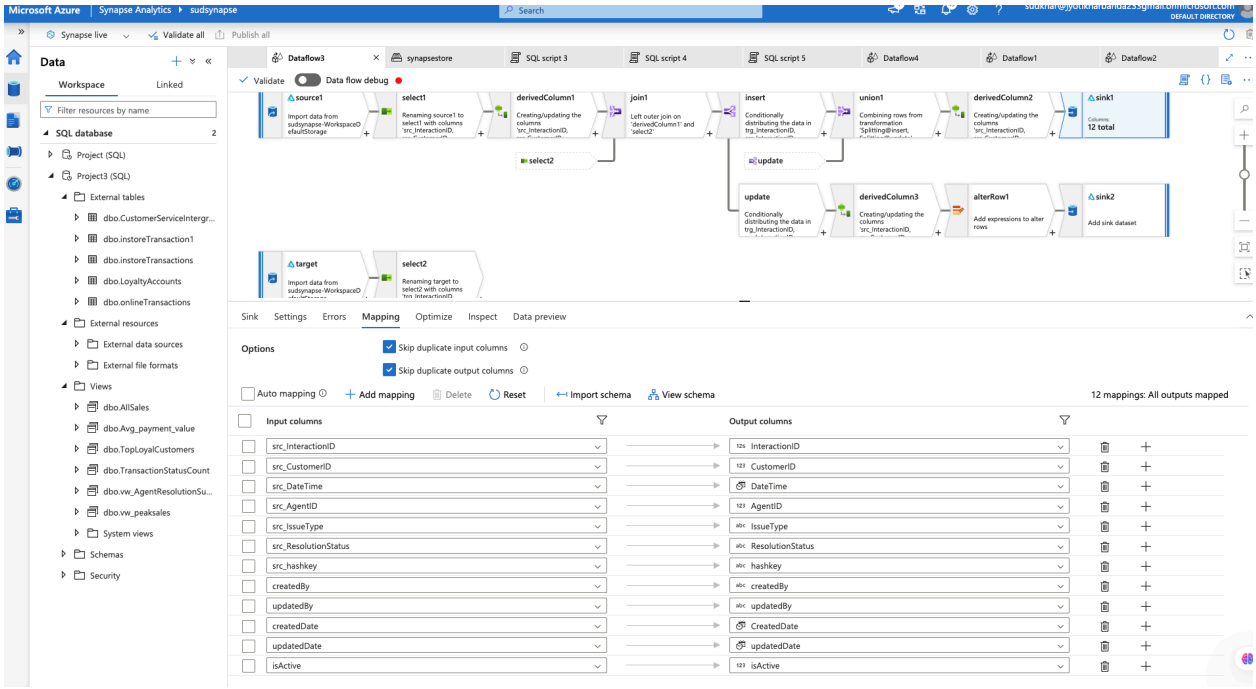
Adding sink activity selecting dataset type as Delta.



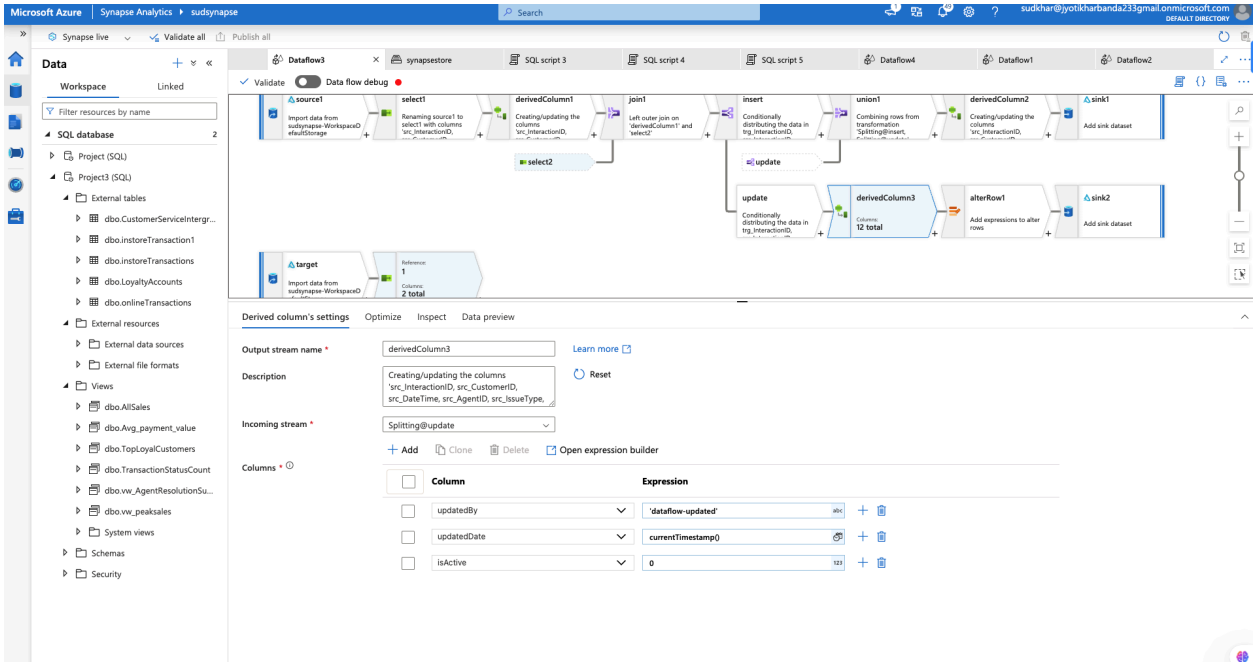
Providing the path where data will be saved.



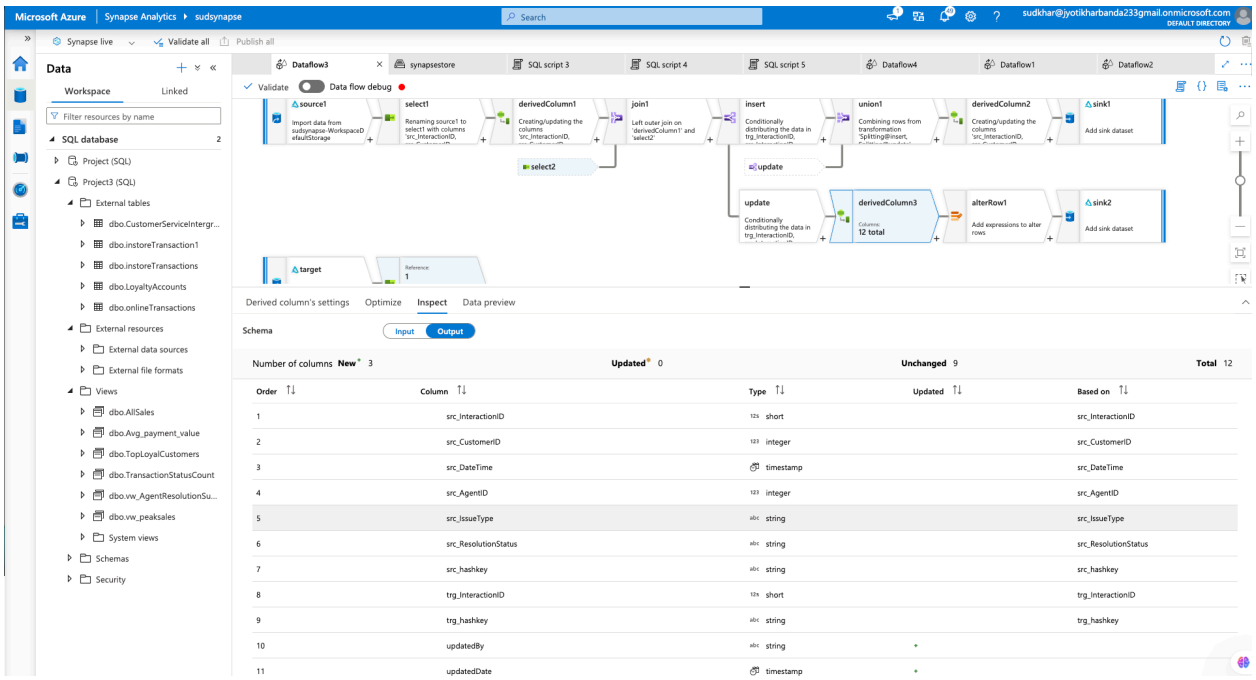
Here is mapping that we are doing inserting the data scd type-2 table.



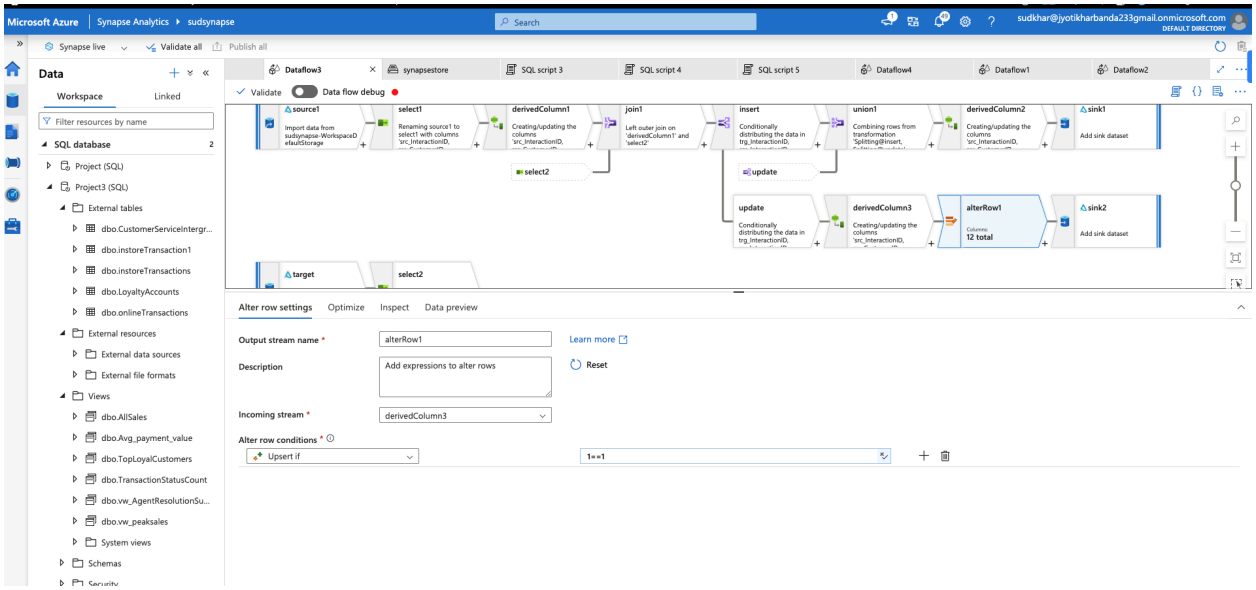
In the update section of the pipeline, a derived column has been added for adding scd type columns that are needed for us.



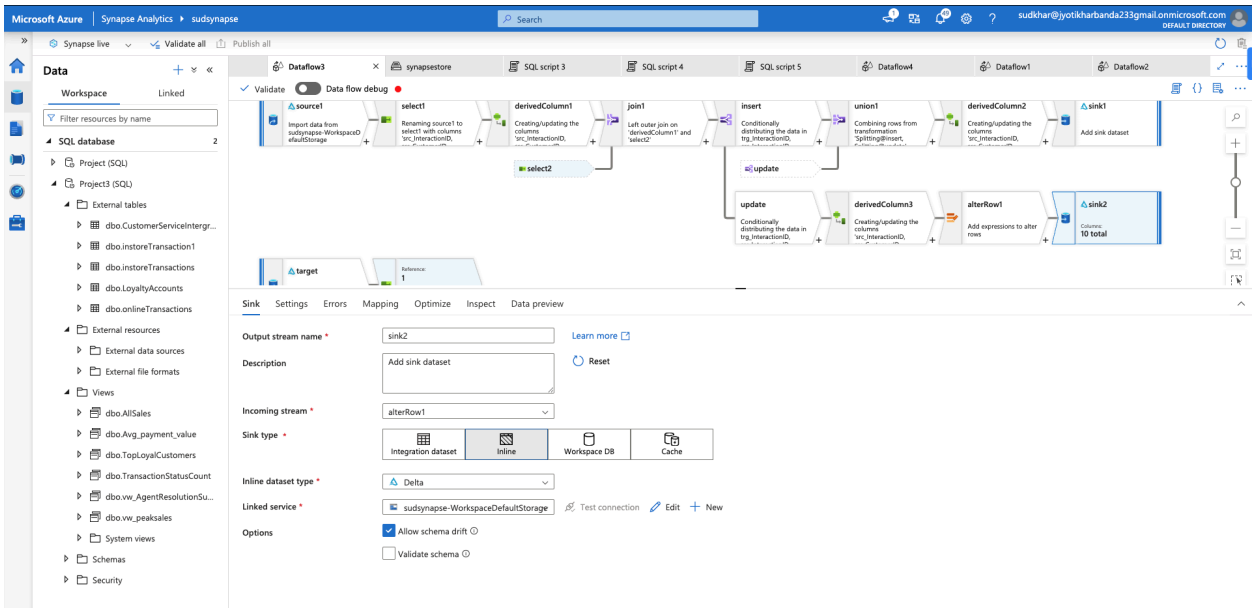
Schema of the data.



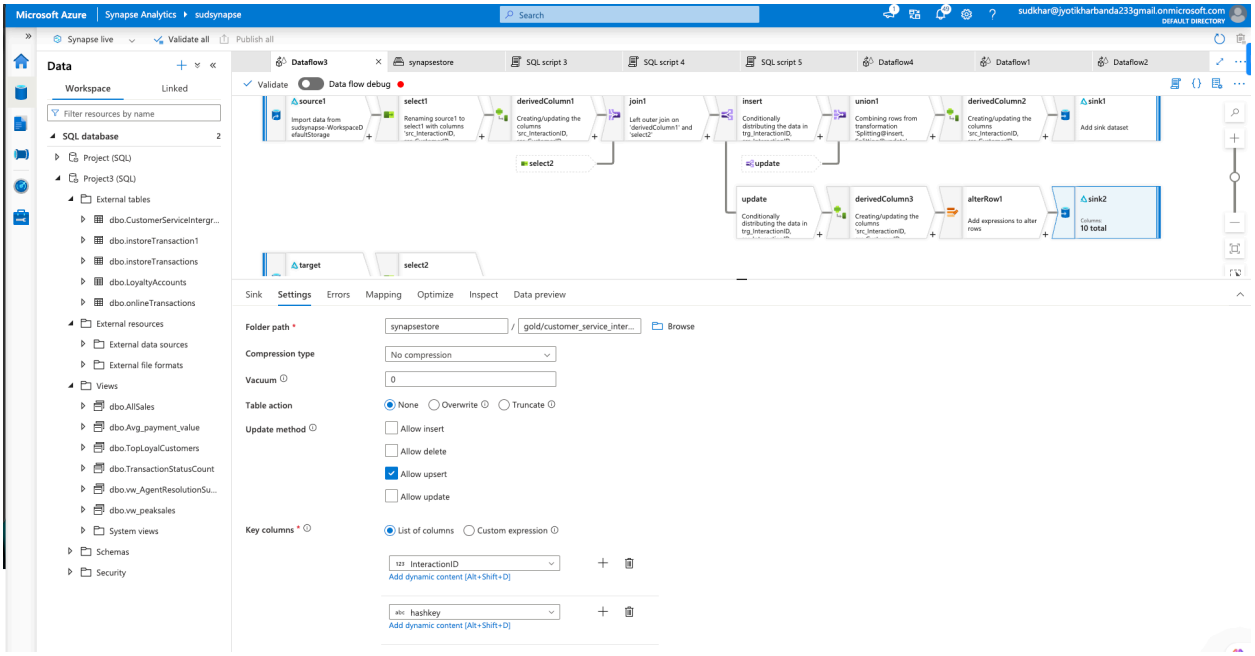
Adding an alter condition since we are upserting in sink.



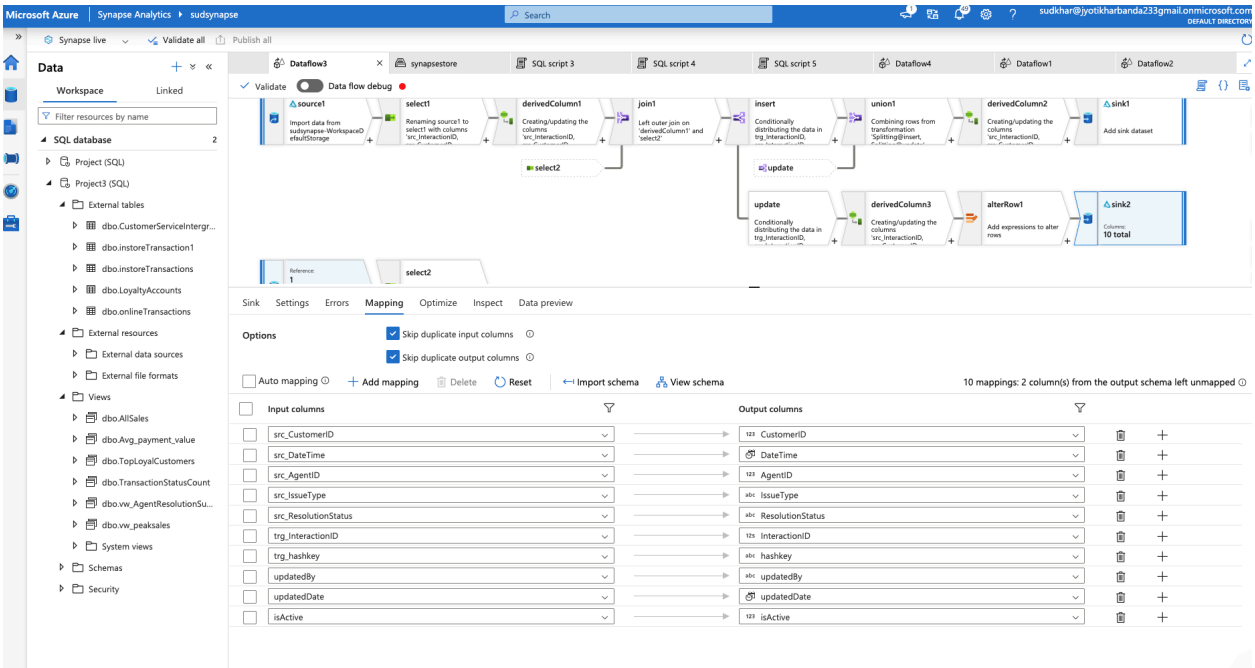
Adding sink activity, defining the dataset type as delta format.



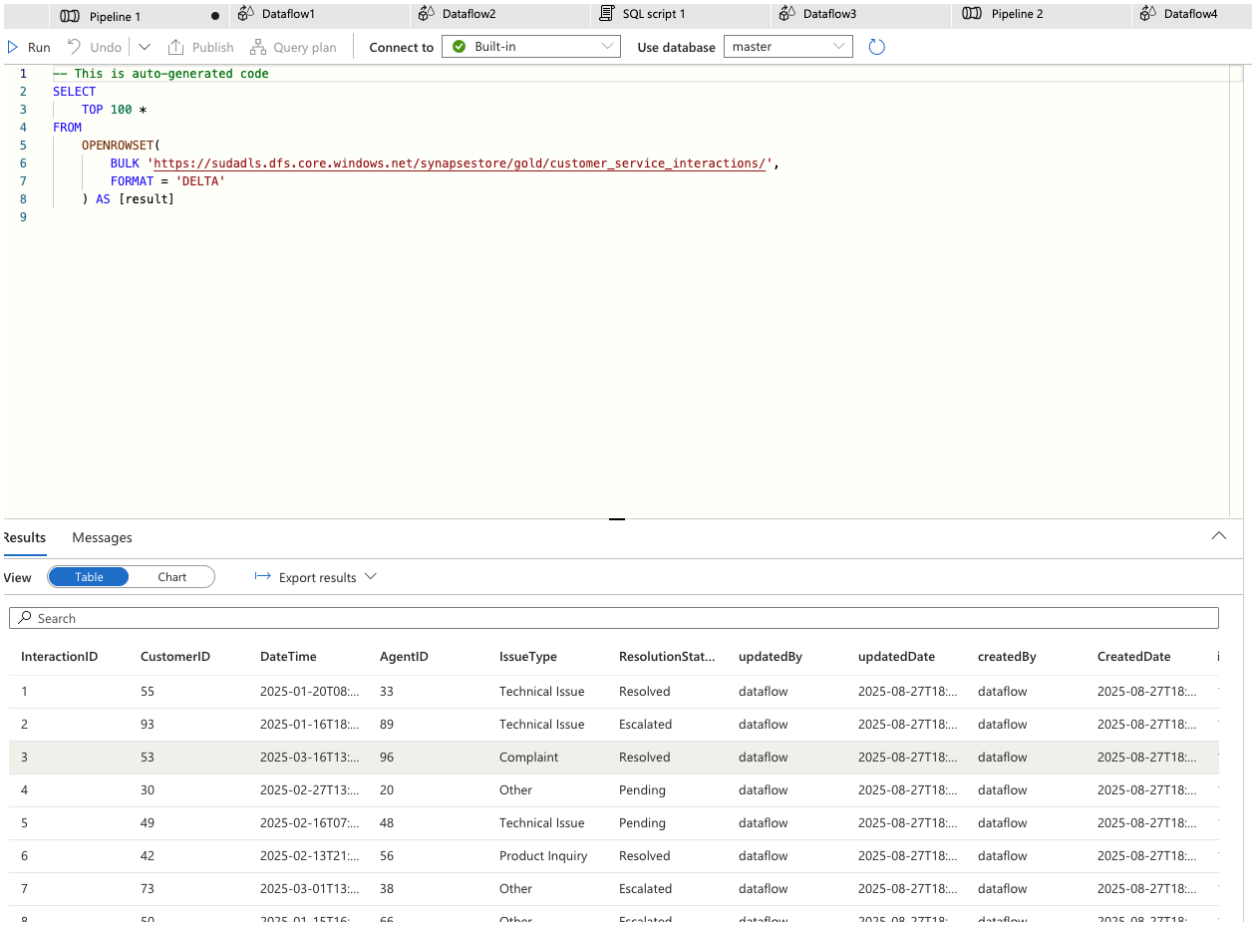
Adding an upsert condition and providing key columns as interaction ID and Hashkey.



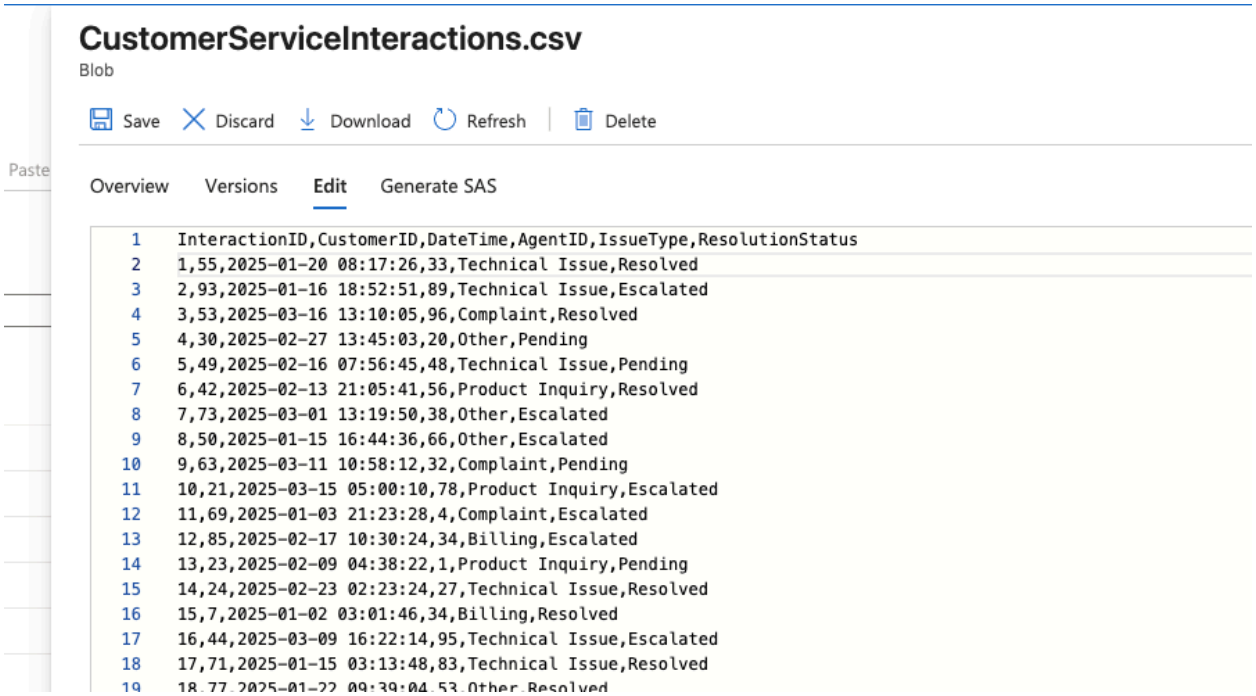
Performing the mapping for the update dataflow.



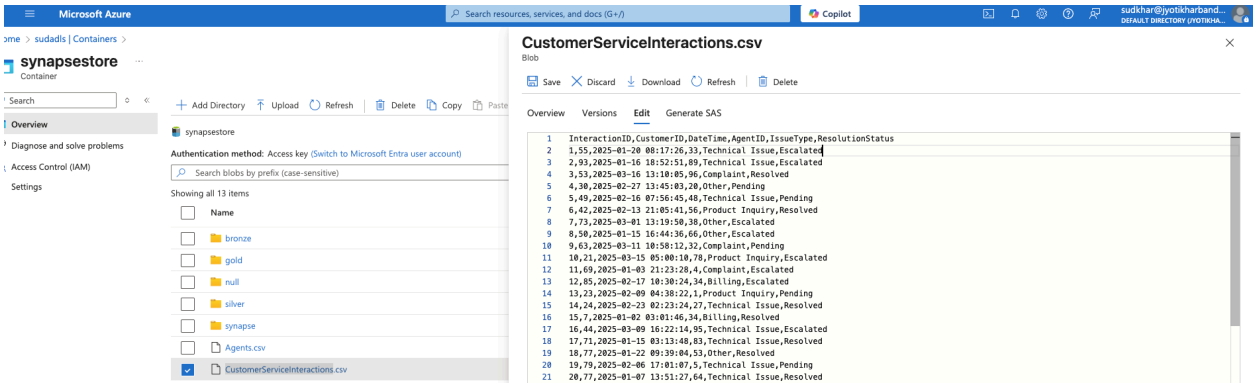
Run the pipeline for the first time and Checking the data in SQL Serverless Pool for the SCD type table.



This is the data without changes.



Changing the first row's Resolution status column to Escalated.



Now running all the dataflows from bronze to gold layer again, excluding the target table creating dataflow.

Checking scd type table. The data has been updated.

```
Select * FROM
 OPENROWSET(
       BULK 'https://sudadls.dfs.core.windows.net/synapsestore/silver/CustomerServiceIntegration/',
       FORMAT = 'DELTA'
    ) AS [result]
```

sults Messages

ew Table Chart Export results

Search

InteractionID	CustomerID	DateTime	AgentID	IssueType	ResolutionStat...
1	55	2025-01-20T08:17:26.0000000	33	Technical Issue	Escalated
2	93	2025-01-16T18:52:51.0000000	89	Technical Issue	Escalated
3	53	2025-03-16T13:10:05.0000000	96	Complaint	Resolved
4	30	2025-02-27T13:45:03.0000000	20	Other	Pending
5	49	2025-02-16T07:56:45.0000000	48	Technical Issue	Pending

Here we can see our old data has been retained, which is the main purpose of SCD type-2.

Run Undo Publish Query plan Connect to Built-in Use database master

```
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://sudadls.dfs.core.windows.net/synapsestore/gold/customer_service_interactions/',
        FORMAT = 'DELTA'
    ) AS [result] WHERE InteractionID =1;
```

esults Messages

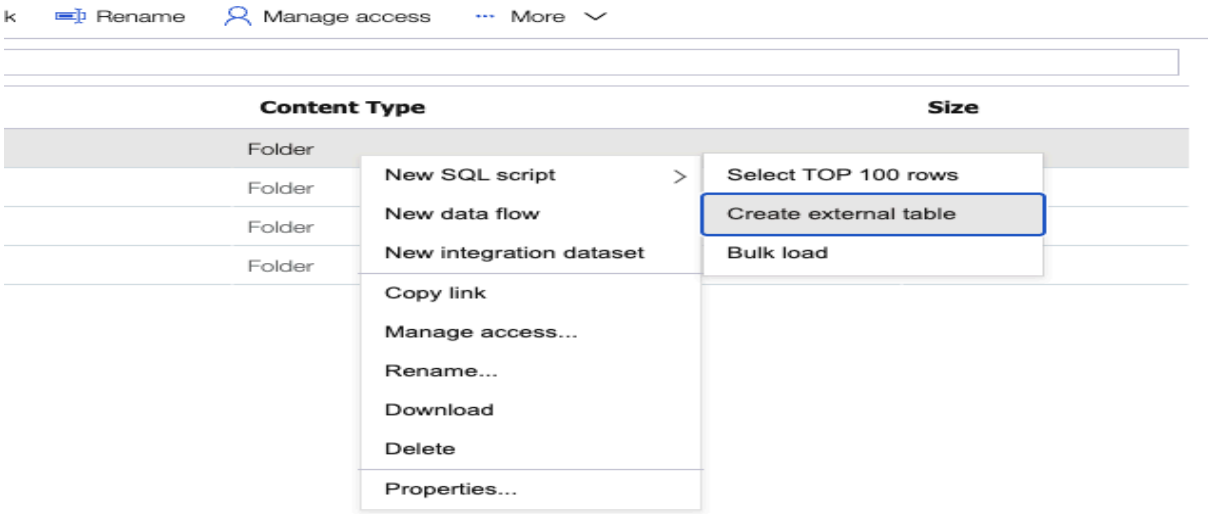
ew Table Chart Export results

Search

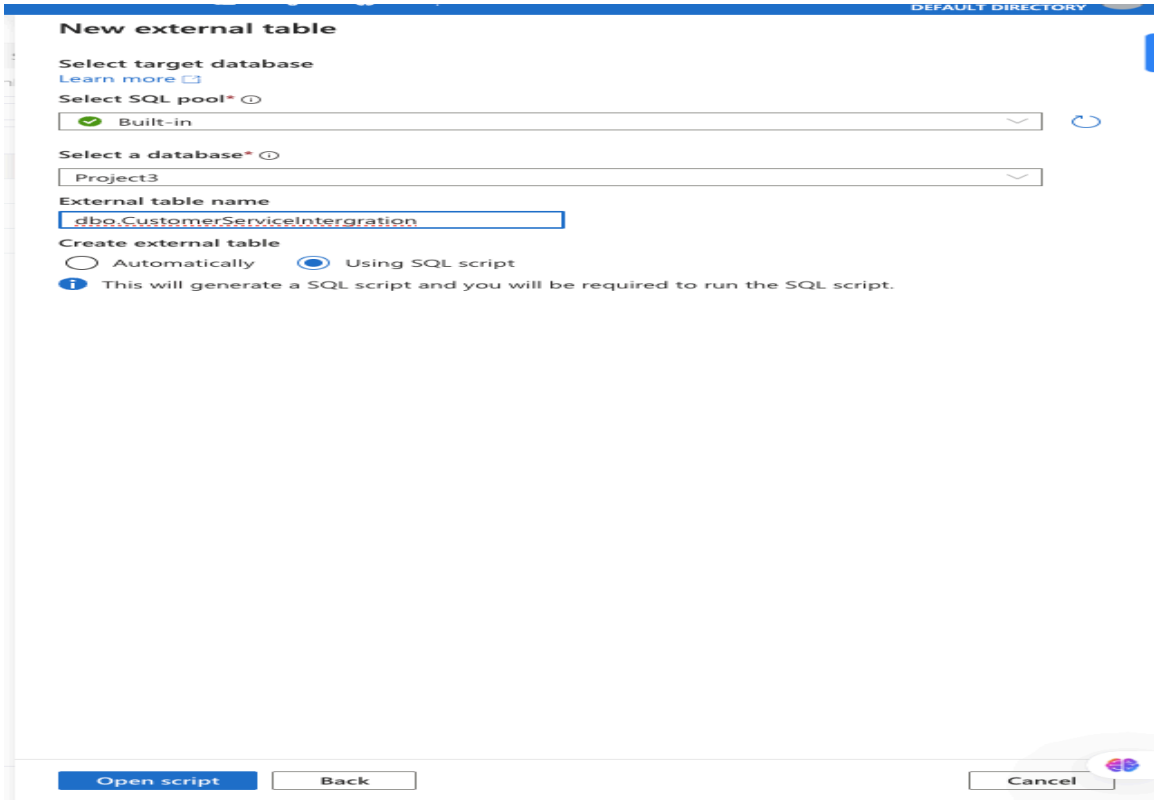
InteractionID	CustomerID	DateTime	AgentID	IssueType	ResolutionStat...	updatedBy	updatedDate	createdBy	CreatedDate
1	55	2025-01-20T08:...	33	Technical Issue	Escalated	dataflow-updat...	2025-08-27T19:...	dataflow	2025-08-27T18:...
1	55	2025-01-20T08:...	33	Technical Issue	Escalated	dataflow	2025-08-27T19:...	dataflow	2025-08-27T19:...

Views

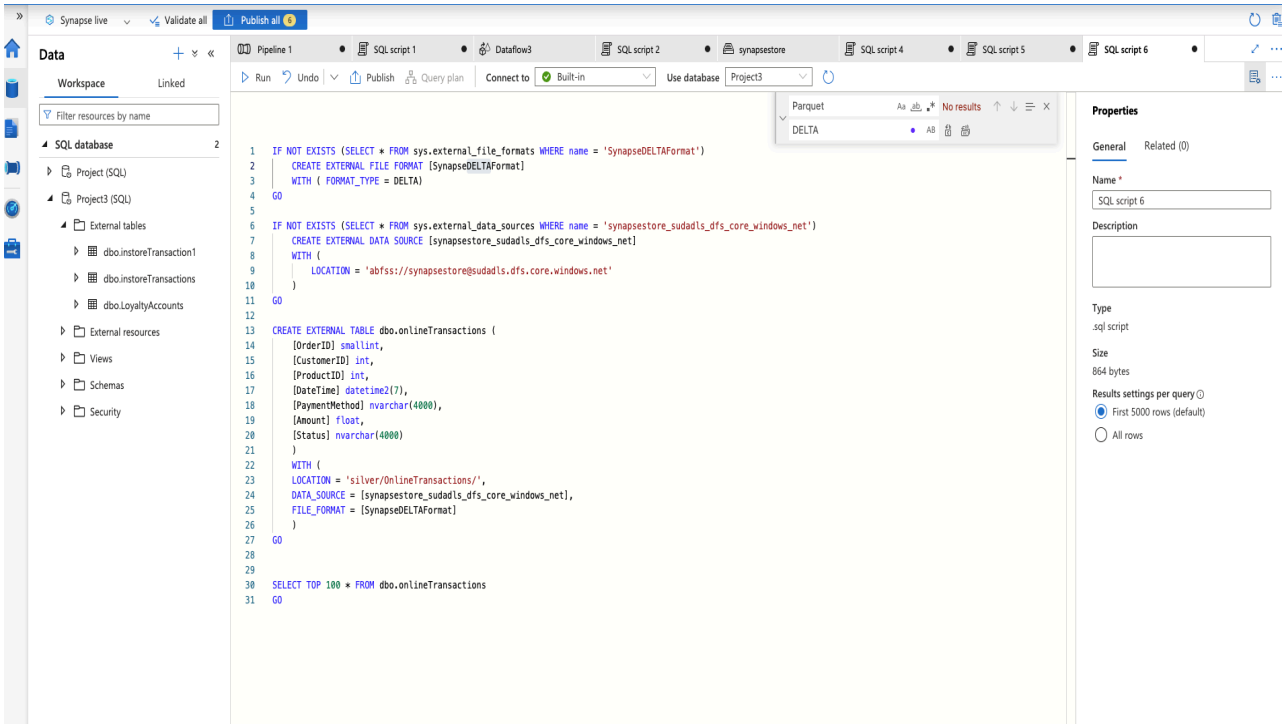
Go to the folder where your Delta File is stored.



Select the database and provide the name of the external table.



We we will replace all the places where parquet is written to Delta in order to create Delta table.

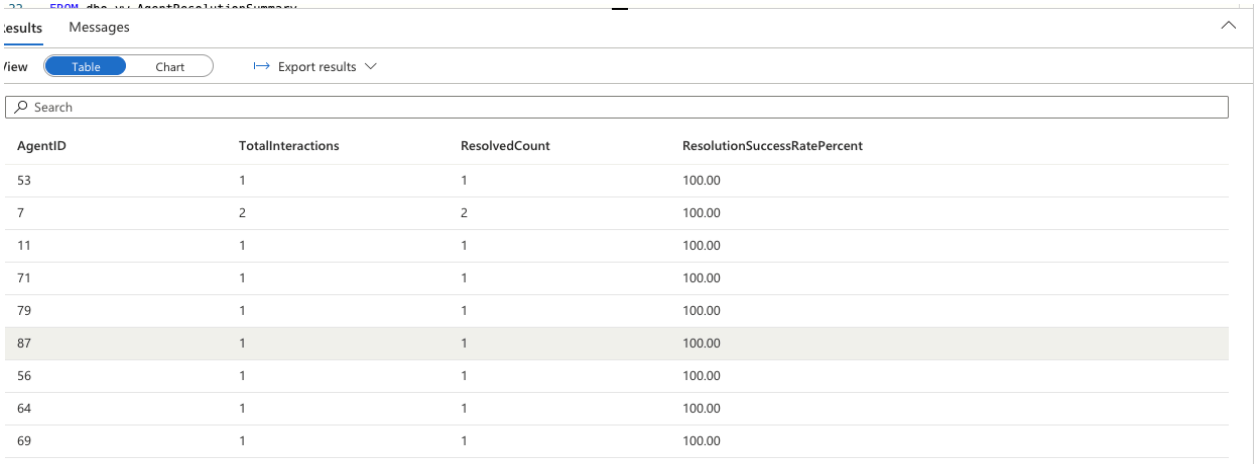


In the same way create external table for all the datafiles stored in silver layer. After table creating views creating will be implemented.

resolution success rates per agent

```
CREATE OR ALTER VIEW dbo.vw_AgentResolutionSummary
AS
SELECT
    AgentID,
    COUNT(*) AS TotalInteractions,
    SUM(CASE WHEN ResolutionStatus = 'Resolved' THEN 1 ELSE 0 END) AS ResolvedCount,
    CAST(SUM(CASE WHEN ResolutionStatus = 'Resolved' THEN 1 ELSE 0 END) * 100.0
        / COUNT(*) AS DECIMAL(5,2)) AS ResolutionSuccessRatePercent
FROM
    OPENROWSET(
        BULK 'https://sudadls.dfs.core.windows.net/synapsestore/silver/CustomerServiceIntegration/',
        FORMAT = 'DELTA'
    ) AS [result]
GROUP BY AgentID
-- ORDER BY ResolutionSuccessRatePercent DESC;

SELECT *
FROM dbo.vw_AgentResolutionSummary
ORDER BY ResolutionSuccessRatePercent DESC;
```



esults Messages

iew Table Chart Export results

Search

AgentID	TotalInteractions	ResolvedCount	ResolutionSuccessRatePercent
53	1	1	100.00
7	2	2	100.00
11	1	1	100.00
71	1	1	100.00
79	1	1	100.00
87	1	1	100.00
56	1	1	100.00
64	1	1	100.00
69	1	1	100.00

for Analyze DateTime to find peak days and times in-store vs. online

```sql
SELECT * FROM dbo.AllSales

SELECT
    DATENAME(WEEKDAY, TransactionDateTime) AS DayOfWeek,
    Channel,
    COUNT(*) AS Transactions
FROM dbo.AllSales
GROUP BY DATENAME(WEEKDAY, TransactionDateTime), Channel
ORDER BY Channel, Transactions DESC;

SELECT
    DATEPART(HOUR, TransactionDateTime) AS HourOfDay,
    Channel,
    COUNT(*) AS Transactions
FROM dbo.AllSales
GROUP BY DATEPART(HOUR, TransactionDateTime), Channel
ORDER BY Channel, Transactions DESC;

CREATE OR ALTER VIEW vw_peaksales
AS
SELECT
    Channel,
    DATENAME(WEEKDAY, TransactionDateTime) AS DayOfWeek,
    DATEPART(HOUR, TransactionDateTime) AS HourOfDay,
    COUNT(*) AS Transactions
FROM dbo.AllSales
GROUP BY Channel, DATENAME(WEEKDAY, TransactionDateTime), DATEPART(HOUR, TransactionDateTime);

SELECT * FROM vw_peaksales
```

Results Messages

View Table Chart Export results

Search

Channel	DayOfWeek	HourOfDay	Transactions
Instore	Tuesday	23	2
Instore	Sunday	22	2
Online	Friday	19	1
Online	Friday	1	1
Online	Monday	9	1
Online	Friday	15	1
Online	Thursday	6	1
Online	Friday	17	1
Online	Tuesday	18	1
Online	Saturday	19	1
Online	Wednesday	12	2

Top 10% spenders per Tier Level

Pipeline 1 | SQL script 1 | synapsestore | SQL script 3 | SQL script 4 | SQL script 5 | Dataflow4

Run Undo Publish Query plan Connect to Built-in Use database Project3

```
SELECT TOP (100) [LoyaltyID]
,[CustomerID]
,[PointsEarned]
,[TierLevel]
,[JoinDate]
 FROM [dbo].[LoyaltyAccounts]

CREATE OR ALTER VIEW TopLoyalCustomers
AS
 WITH RankedCustomers AS (
    SELECT
        LoyaltyID,
        CustomerID,
        PointsEarned,
        TierLevel,
        JoinDate,
        NTILE(10) OVER (
            PARTITION BY TierLevel
            ORDER BY PointsEarned DESC
        ) AS decile
    FROM
        [dbo].[LoyaltyAccounts]
)
SELECT
        LoyaltyID,
        CustomerID,
        PointsEarned,
        TierLevel,
        JoinDate
FROM RankedCustomers
WHERE decile = 1;

Select * FROM dbo.TopLoyalCustomers;
```

Results Messages

View Table Chart Export results

Search

LoyaltyID	CustomerID	PointsEarned	TierLevel	JoinDate
30	41	4934	Bronze	2024-07-05
68	1	4706	Bronze	2022-02-02
100	73	4601	Bronze	2020-08-26
28	61	4616	Platinum	2020-10-04
62	9	4603	Platinum	2024-07-07
99	11	4338	Platinum	2023-06-01
80	63	4863	Gold	2021-10-19
3	13	4782	Gold	2023-04-11
90	47	4720	Gold	2021-08-27
73	35	4532	Gold	2024-09-16
43	22	4632	Silver	2023-12-11
94	23	4603	Silver	2023-03-30

Average payment in each payment method.

```
1
2   CREATE OR ALTER VIEW dbo.Avg_payment_value
3   AS
4     SELECT
5       PaymentMethod,
6       ROUND(AVG(AMOUNT), 2) AS AveragePaymentValue
7   FROM
8       [dbo].[instoreTransactions]
9   GROUP BY
10      PaymentMethod;
11
12
13  Select * FROM dbo.Avg_payment_value
```

Results Messages

View Table Chart Export results

Search

PaymentMethod	AveragePaymentValue
Gift Card	89.43
PayPal	100.66
Debit Card	113.52
Credit Card	111.32

Json Code of the pipeline and data flows

Pipeline


```
{
   "name": "Pipeline 1",
   "properties": {
       "activities": [
           {
               "name": "Data flow1",
               "type": "ExecuteDataFlow",
               "state": "Inactive",
               "onInactiveMarkAs": "Succeeded",
               "dependsOn": [],
               "policy": {
                   "timeout": "0.12:00:00",
                   "retry": 0,
                   "retryIntervalInSeconds": 30,
                   "secureOutput": false,
                   "secureInput": false
               },
               "userProperties": [],
               "typeProperties": {
                   "dataflow": {
                       "referenceName": "Dataflow1",
                       "type": "DataFlowReference"
                   },
                   "compute": {
                       "coreCount": 8,
                       "computeType": "General"
```

```
            },
            "traceLevel": "Fine"
        }
    },
    {
        "name": "Data flow2",
        "type": "ExecuteDataFlow",
        "state": "Inactive",
        "onInactiveMarkAs": "Succeeded",
        "dependsOn": [
            {
                "activity": "Data flow1",
                "dependencyConditions": [
                    "Succeeded"
                ]
            }
        ],
        "policy": {
            "timeout": "0.12:00:00",
            "retry": 0,
            "retryIntervalInSeconds": 30,
            "secureOutput": false,
            "secureInput": false
        },
        "userProperties": [],
        "typeProperties": {
            "dataflow": {
                "referenceName": "Dataflow2",
```

```
                "type": "DataFlowReference"
            },
            "compute": {
                "coreCount": 8,
                "computeType": "General"
            },
            "traceLevel": "Fine"
        }
    },
    {
        "name": "CreatingTargetTable",
        "type": "ExecuteDataFlow",
        "state": "Inactive",
        "onInactiveMarkAs": "Succeeded",
        "dependsOn": [],
        "policy": {
            "timeout": "0.12:00:00",
            "retry": 0,
            "retryIntervalInSeconds": 30,
            "secureOutput": false,
            "secureInput": false
        },
        "userProperties": [],
        "typeProperties": {
            "dataflow": {
                "referenceName": "Dataflow4",
                "type": "DataFlowReference"
            },
```

```
            "compute": {
                "coreCount": 8,
                "computeType": "General"
            },
            "traceLevel": "Fine"
        }
    },
    {
        "name": "SCDType-2",
        "type": "ExecuteDataFlow",
        "dependsOn": [],
        "policy": {
            "timeout": "0.12:00:00",
            "retry": 0,
            "retryIntervalInSeconds": 30,
            "secureOutput": false,
            "secureInput": false
        },
        "userProperties": [],
        "typeProperties": {
            "dataflow": {
                "referenceName": "Dataflow3",
                "type": "DataFlowReference"
            },
            "compute": {
                "coreCount": 8,
                "computeType": "General"
            },
```

```
                    "traceLevel": "Fine"
                }
            }
        ],
        "annotations": []
    }
}
```


Dataflow for bronze to silver


```
{
    "name": "Dataflow1",
    "properties": {
        "type": "MappingDataFlow",
        "typeProperties": {
            "sources": [
                {
                    "linkedService": {
                        "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                        "type": "LinkedServiceReference"
                    },
                    "name": "source1"
                },
                {
                    "linkedService": {
                        "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                        "type": "LinkedServiceReference"
                    },
                    "name": "source2"
                },
```

```
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "source3"
            },
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "source4"
            }
        ],
        "sinks": [
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "sink1"
            },
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
```

```
            "name": "sink2"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink3"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink4"
        }
    ],
    "transformations": [],
    "scriptLines": [
        "source(useSchema: false,",
        "     allowSchemaDrift: true,",
        "     validateSchema: false,",
        "     ignoreNoFilesFound: false,",
        "     format: 'delimited',",
        "     fileSystem: 'synapsestore',",
        "     fileName: 'CustomerServiceInteractions.csv',",
        "     columnDelimiter: ',',",
        "     escapeChar: '\\\\',",
```

```
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source1",
"source(output(",
"          TransactionID as short,",
"          CustomerID as short,",
"          StoreID as short,",
"          DateTime as timestamp,",
"          Amount as double,",
"          PaymentMethod as string",
"     ),",
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     fileName: 'InStoreTransactions.csv',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source2",
"source(output(",
"          LoyaltyID as short,",
"          CustomerID as short,",
"          PointsEarned as short,",
"          TierLevel as string,",
"          JoinDate as date",
"     ),",
```

```
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     fileName: 'LoyaltyAccounts.csv',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source3",
"source(output(",
"          OrderID as short,",
"          CustomerID as short,",
"          ProductID as short,",
"          DateTime as timestamp,",
"          PaymentMethod as string,",
"          Amount as double,",
"          Status as string",
"     ),",
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     fileName: 'OnlineTransactions.csv',",
"     columnDelimiter: ',',",
```

```
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source4",
"source1 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/CustomerServiceIntegration',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink1",
"source2 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/instoretransactions',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
```

```
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink2",
"source3 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/LoyaltyAccounts',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink3",
"source4 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/OnilneTransactions',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
```

```
                "     postCommands: [],",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> sink4"
            ]
        }
    }
}
```

Cleaning dataflow Bronze to Silver

```
{
    "name": "Dataflow2",
    "properties": {
        "type": "MappingDataFlow",
        "typeProperties": {
            "sources": [
                {
                    "linkedService": {
                        "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                        "type": "LinkedServiceReference"
                    },
                    "name": "source1"
                },
                {
                    "linkedService": {
                        "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                        "type": "LinkedServiceReference"
                    },
                    "name": "source2"
                },
```

```
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "source3"
            },
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "source4"
            }
        ],
        "sinks": [
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
                "name": "sink1"
            },
            {
                "linkedService": {
                    "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                    "type": "LinkedServiceReference"
                },
```

```
            "name": "sink2"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink3"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink4"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink5"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
```

```
            "name": "sink6"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink7"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink8"
        }
    ],
    "transformations": [
        {
            "name": "split1"
        },
        {
            "name": "derivedColumn1"
        },
        {
            "name": "window1"
        },
        {
```

```
        "name": "split2"
    },
    {
        "name": "alterRow1"
    },
    {
        "name": "select1"
    },
    {
        "name": "union1"
    },
    {
        "name": "split"
    },
    {
        "name": "split4"
    },
    {
        "name": "split5"
    },
    {
        "name": "derivedColumn2"
    },
    {
        "name": "window2"
    },
    {
        "name": "split6"
```

```
},
{
    "name": "select2"
},
{
    "name": "alterRow2"
},
{
    "name": "derivedColumn3"
},
{
    "name": "window3"
},
{
    "name": "split7"
},
{
    "name": "select3"
},
{
    "name": "alterRow3"
},
{
    "name": "derivedColumn4"
},
{
    "name": "window4"
},
```

```
        {
            "name": "split8"
        },
        {
            "name": "select4"
        },
        {
            "name": "alterRow4"
        },
        {
            "name": "union2"
        },
        {
            "name": "union3"
        },
        {
            "name": "union4"
        }
    ],
    "scriptLines": [
        "source(output(",
        "          InteractionID as short,",
        "          CustomerID as short,",
        "          DateTime as timestamp,",
        "          AgentID as short,",
        "          IssueType as string,",
        "          ResolutionStatus as string",
        "     ),",
```

```
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/CustomerServiceIntegration',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source1",
"source(output(",
"          LoyaltyID as short,",
"          CustomerID as short,",
"          PointsEarned as short,",
"          TierLevel as string,",
"          JoinDate as date",
"     ),",
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/LoyaltyAccounts',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
```

```
"     columnNamesAsHeader: true) ~> source2",
"source(output(",
"          OrderID as short,",
"          CustomerID as short,",
"          ProductID as short,",
"          DateTime as timestamp,",
"          PaymentMethod as string,",
"          Amount as double,",
"          Status as string",
"     ),",
"     useSchema: false,",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'bronze/OnilneTransactions',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true) ~> source3",
"source(output(",
"          TransactionID as short,",
"          CustomerID as short,",
"          StoreID as short,",
"          DateTime as timestamp,",
"          Amount as double,",
"          PaymentMethod as string",
```

```
                "     ),",
                "     useSchema: false,",
                "     allowSchemaDrift: true,",
                "     validateSchema: false,",
                "     ignoreNoFilesFound: false,",
                "     format: 'delimited',",
                "     fileSystem: 'synapsestore',",
                "     folderPath: 'bronze/instoretransactions',",
                "     columnDelimiter: ',',",
                "     escapeChar: '\\\\',",
                "     quoteChar: '\\\"',",
                "     columnNamesAsHeader: true) ~> source4",
                "source1 split(!isNull(InteractionID),",
                "     disjoint: false) ~> split1@(NoNullData, NullData)",
                "split1@NoNullData derive(CustomerID = iifNull(CustomerID, 0),",
                "          DateTime = iifNull(DateTime, currentTimestamp()),",
                "          AgentID = iifNull(AgentID, 0),",
                "          IssueType = iifNull(IssueType, 'NA'),",
                "          ResolutionStatus = iifNull(ResolutionStatus, 'NA')) ~>
derivedColumn1",
                "derivedColumn1 window(over(InteractionID),",
                "     asc(InteractionID, true),",
                "     RownumberDup = rowNumber()) ~> window1",
                "window1 split(RownumberDup==1,",
                "     disjoint: false) ~> split2@(unique, duplicate)",
                "select1 alterRow(upsertIf(1==1)) ~> alterRow1",
                "split2@unique select(mapColumn(",
                "          InteractionID,",
                "          CustomerID,",
```

```
                "          DateTime,",
                "          AgentID,",
                "          IssueType,",
                "          ResolutionStatus",
                "     ),",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> select1",
                "split2@duplicate, split1@NullData union(byName: true)~> union1",
                "source2 split(!isNull(LoyaltyID),",
                "     disjoint: false) ~> split@(NotNull, Null)",
                "source3 split(!isNull(OrderID),",
                "     disjoint: false) ~> split4@(NoNull, nullData)",
                "source4 split(!isNull(TransactionID),",
                "     disjoint: false) ~> split5@(NotNull1, NullData1)",
                "split@NotNull derive(CustomerID = iifNull(CustomerID, 0),",
                "          PointsEarned = iifNull(PointsEarned, 0),",
                "          TierLevel = iifNull(TierLevel, 'NA'),",
                "          JoinDate = iifNull(JoinDate, currentDate())) ~>
derivedColumn2",
                "derivedColumn2 window(over(LoyaltyID),",
                "     asc(LoyaltyID, true),",
                "     RowNumber = rowNumber()) ~> window2",
                "window2 split(RowNumber==1,",
                "     disjoint: false) ~> split6@(unique1, duplicate1)",
                "split6@unique1 select(mapColumn(",
                "          LoyaltyID,",
                "          CustomerID,",
                "          PointsEarned,",
                "          TierLevel,",
```

```
"          JoinDate",
"     ),",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> select2",
"select2 alterRow(upsertIf(1==1)) ~> alterRow2",
"split4@NoNull derive(CustomerID = iifNull(CustomerID, 0),",
"          ProductID = iifNull(ProductID, 0),",
"          DateTime = iifNull(DateTime, currentTimestamp()),",
"          PaymentMethod = iifNull(PaymentMethod, 'NA'),",
"          Amount = iifNull(Amount, 0),",
"          Status = iifNull(Status, 'NA')) ~> derivedColumn3",
"derivedColumn3 window(over(OrderID),",
"     asc(OrderID, true),",
"     RowNumber = rowNumber()) ~> window3",
"window3 split(RowNumber==1,",
"     disjoint: false) ~> split7@(unique2, duplicate2)",
"split7@unique2 select(mapColumn(",
"          OrderID,",
"          CustomerID,",
"          ProductID,",
"          DateTime,",
"          PaymentMethod,",
"          Amount,",
"          Status",
"     ),",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> select3",
"select3 alterRow(upsertIf(1==1)) ~> alterRow3",
```

```
                "split5@NotNull1 derive(CustomerID = iifNull(CustomerID, 0),",
                "          StoreID = iifNull(StoreID, 0),",
                "          DateTime = iifNull(DateTime, currentTimestamp()),",
                "          Amount = iifNull(Amount, 0),",
                "          PaymentMethod = iifNull(PaymentMethod, 'NA')) ~>
derivedColumn4",
                "derivedColumn4 window(over(TransactionID),",
                "     asc(TransactionID, true),",
                "     RowNumber = rowNumber()) ~> window4",
                "window4 split(RowNumber==1,",
                "     disjoint: false) ~> split8@(unique3, duplicate3)",
                "split8@unique3 select(mapColumn(",
                "          TransactionID,",
                "          CustomerID,",
                "          StoreID,",
                "          DateTime,",
                "          Amount,",
                "          PaymentMethod",
                "     ),",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> select4",
                "select4 alterRow(upsertIf(1==1)) ~> alterRow4",
                "split6@duplicate1, split@Null union(byName: true)~> union2",
                "split7@duplicate2, split4@nullData union(byName: true)~> union3",
                "split8@duplicate3, split5@NullData1 union(byName: true)~> union4",
                "alterRow1 sink(allowSchemaDrift: true,",
                "     validateSchema: false,",
                "     format: 'delta',",
                "     fileSystem: 'synapsestore',",
```

```
"     folderPath: 'silver/CustomerServiceIntegration',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: false,",
"     updateable: false,",
"     upsertable: true,",
"     keys:['InteractionID'],",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink1",
"union1 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'null/CustomerServiceIntegration',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
```

```
"     skipDuplicateMapOutputs: true) ~> sink2",
"alterRow2 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'silver/LoyaltyAccounts',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: false,",
"     updateable: false,",
"     upsertable: true,",
"     keys:['LoyaltyID'],",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink3",
"union2 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'null/LoyaltyAccounts',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
```

```
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink4",
"alterRow3 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'silver/OnlineTransactions',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: false,",
"     updateable: false,",
"     upsertable: true,",
"     keys:['OrderID'],",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink5",
"union3 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delimited',",
```

```
"     fileSystem: 'synapsestore',",
"     folderPath: 'null/OnlineTransactions',",
"     columnDelimiter: ',',",
"     escapeChar: '\\\\',",
"     quoteChar: '\\\"',",
"     columnNamesAsHeader: true,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> sink6",
"alterRow4 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'silver/InStoreTransactions',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: false,",
"     updateable: false,",
"     upsertable: true,",
"     keys:['TransactionID'],",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
```

```
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> sink7",
                "union4 sink(allowSchemaDrift: true,",
                "     validateSchema: false,",
                "     format: 'delimited',",
                "     fileSystem: 'synapsestore',",
                "     folderPath: 'null/InStoreTransactions',",
                "     columnDelimiter: ',',",
                "     escapeChar: '\\\\',",
                "     quoteChar: '\\\"',",
                "     columnNamesAsHeader: true,",
                "     umask: 0022,",
                "     preCommands: [],",
                "     postCommands: [],",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> sink8"
            ]
        }
    }
}
```

Target table creation

```
{
    "name": "Dataflow4",
    "properties": {
        "type": "MappingDataFlow",
        "typeProperties": {
            "sources": [
                {
```

```
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "source1"
        }
    ],
    "sinks": [
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink1"
        }
    ],
    "transformations": [
        {
            "name": "filter1"
        },
        {
            "name": "derivedColumn1"
        }
    ],
    "scriptLines": [
        "source(output(",
        "          InteractionID as short,",
        "          CustomerID as integer,",
```

```
"          DateTime as timestamp,",
"          AgentID as integer,",
"          IssueType as string,",
"          ResolutionStatus as string",
"     ),",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'silver/CustomerServiceIntegration') ~> source1",
"source1 filter(false()) ~> filter1",
"filter1 derive(updatedBy = '',",
"          updatedDate = currentTimestamp(),",
"          createdBy = '',",
"          CreatedDate = currentTimestamp(),",
"          isActive = 0,",
"          hashkey = '') ~> derivedColumn1",
"derivedColumn1 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'gold/customer_service_interactions',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
```

```
                "     insertable: true,",
                "     updateable: false,",
                "     upsertable: false,",
                "     umask: 0022,",
                "     preCommands: [],",
                "     postCommands: [],",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> sink1"
            ]
        }
    }
}
```

Silver to gold

SCD type -2 Implemenation

```
{
    "name": "Dataflow3",
    "properties": {
        "type": "MappingDataFlow",
        "typeProperties": {
            "sources": [
                {
                    "linkedService": {
                        "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                        "type": "LinkedServiceReference"
                    },
                    "name": "source1"
                },
```

```
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "target"
        }
    ],
    "sinks": [
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink1"
        },
        {
            "linkedService": {
                "referenceName": "sudsynapse-WorkspaceDefaultStorage",
                "type": "LinkedServiceReference"
            },
            "name": "sink2"
        }
    ],
    "transformations": [
        {
            "name": "select1"
        },
```

```
        {
            "name": "derivedColumn1"
        },
        {
            "name": "join1"
        },
        {
            "name": "Splitting"
        },
        {
            "name": "union1"
        },
        {
            "name": "derivedColumn2"
        },
        {
            "name": "derivedColumn3"
        },
        {
            "name": "alterRow1"
        },
        {
            "name": "select2"
        }
    ],
    "scriptLines": [
        "source(output(",
        "          InteractionID as short,",
```

```
"          CustomerID as integer,",
"          DateTime as timestamp,",
"          AgentID as integer,",
"          IssueType as string,",
"          ResolutionStatus as string",
"     ),",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
"     ignoreNoFilesFound: false,",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'silver/CustomerServiceIntegration') ~> source1",
"source(output(",
"          InteractionID as short,",
"          CustomerID as integer,",
"          DateTime as timestamp,",
"          AgentID as integer,",
"          IssueType as string,",
"          ResolutionStatus as string,",
"          updatedBy as string,",
"          updatedDate as timestamp,",
"          createdBy as string,",
"          CreatedDate as timestamp,",
"          isActive as integer,",
"          hashkey as string",
"     ),",
"     allowSchemaDrift: true,",
"     validateSchema: false,",
```

```
                "     ignoreNoFilesFound: false,",
                "     format: 'delta',",
                "     fileSystem: 'synapsestore',",
                "     folderPath: 'gold/customer_service_interactions') ~> target",
                "source1 select(mapColumn(",
                "          each(match(1==1),",
                "               concat('src_',$$) = $$)",
                "     ),",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true) ~> select1",
                "select1 derive(src_hashkey =
toString(crc32(concat(toString(src_CustomerID),toString(src_DateTime),toString(src_Age
ntID),src_IssueType,src_ResolutionStatus)))) ~> derivedColumn1",
                "derivedColumn1, select2 join(src_InteractionID == trg_InteractionID,",
                "     joinType:'left',",
                "     matchType:'exact',",
                "     ignoreSpaces: false,",
                "     broadcast: 'auto')~> join1",
                "join1 split(isNull(trg_InteractionID),",
                "     (src_InteractionID==trg_InteractionID)
&&(src_hashkey!=trg_hashkey),",
                "     disjoint: false) ~> Splitting@(insert, update)",
                "Splitting@insert, Splitting@update union(byName: true)~> union1",
                "union1 derive(createdBy = 'dataflow',",
                "          updatedBy = 'dataflow',",
                "          createdDate = currentTimestamp(),",
                "          updatedDate = currentTimestamp(),",
                "          isActive = 1) ~> derivedColumn2",
                "Splitting@update derive(updatedBy = 'dataflow-updated',",
```

```
"          updatedDate = currentTimestamp(),",
"          isActive = 0) ~> derivedColumn3",
"derivedColumn3 alterRow(upsertIf(1==1)) ~> alterRow1",
"target select(mapColumn(",
"          trg_InteractionID = InteractionID,",
"          trg_hashkey = hashkey",
"     ),",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true) ~> select2",
"derivedColumn2 sink(allowSchemaDrift: true,",
"     validateSchema: false,",
"     input(",
"          InteractionID as short,",
"          CustomerID as integer,",
"          DateTime as timestamp,",
"          AgentID as integer,",
"          IssueType as string,",
"          ResolutionStatus as string,",
"          updatedBy as string,",
"          updatedDate as timestamp,",
"          createdBy as string,",
"          CreatedDate as timestamp,",
"          isActive as integer,",
"          hashkey as string",
"     ),",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'gold/customer_service_interactions',",
```

```
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: true,",
"     updateable: false,",
"     upsertable: false,",
"     umask: 0022,",
"     preCommands: [],",
"     postCommands: [],",
"     skipDuplicateMapInputs: true,",
"     skipDuplicateMapOutputs: true,",
"     mapColumn(",
"          InteractionID = src_InteractionID,",
"          CustomerID = src_CustomerID,",
"          DateTime = src_DateTime,",
"          AgentID = src_AgentID,",
"          IssueType = src_IssueType,",
"          ResolutionStatus = src_ResolutionStatus,",
"          hashkey = src_hashkey,",
"          createdBy,",
"          updatedBy,",
"          CreatedDate = createdDate,",
"          updatedDate,",
"          isActive",
"     )) ~> sink1",
"alterRow1 sink(allowSchemaDrift: true,",
```

```
"     validateSchema: false,",
"     input(",
"          InteractionID as short,",
"          CustomerID as integer,",
"          DateTime as timestamp,",
"          AgentID as integer,",
"          IssueType as string,",
"          ResolutionStatus as string,",
"          updatedBy as string,",
"          updatedDate as timestamp,",
"          createdBy as string,",
"          CreatedDate as timestamp,",
"          isActive as integer,",
"          hashkey as string",
"     ),",
"     format: 'delta',",
"     fileSystem: 'synapsestore',",
"     folderPath: 'gold/customer_service_interactions',",
"     mergeSchema: false,",
"     autoCompact: false,",
"     optimizedWrite: false,",
"     vacuum: 0,",
"     deletable: false,",
"     insertable: false,",
"     updateable: false,",
"     upsertable: true,",
"     keys:['InteractionID','hashkey'],",
"     umask: 0022,",
```

```
                "     preCommands: [],",
                "     postCommands: [],",
                "     skipDuplicateMapInputs: true,",
                "     skipDuplicateMapOutputs: true,",
                "     mapColumn(",
                "          CustomerID = src_CustomerID,",
                "          DateTime = src_DateTime,",
                "          AgentID = src_AgentID,",
                "          IssueType = src_IssueType,",
                "          ResolutionStatus = src_ResolutionStatus,",
                "          InteractionID = trg_InteractionID,",
                "          hashkey = trg_hashkey,",
                "          updatedBy,",
                "          updatedDate,",
                "          isActive",
                "     )) ~> sink2"
            ]
        }
    }
}
```