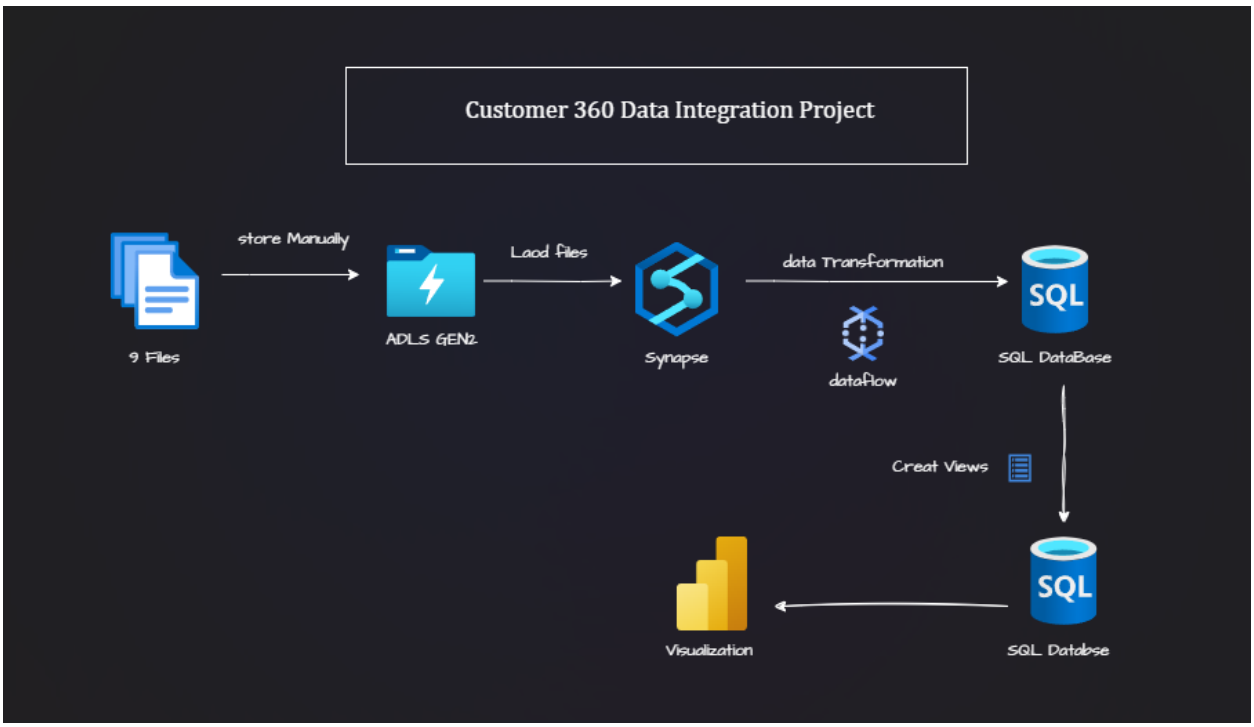
Customer 360 Data Integration Project Report

Overview

The Customer 360 Data Integration project aims to build a unified view of customer behavior by combining data from multiple sources such as online transactions, in-store purchases, customer service interactions, and loyalty programs. This project is implemented using Azure Data Lake Storage (ADLS), Azure Synapse Analytics, Azure SQL Database, and Power BI.

Architecture Diagram



Step 1: Ingest Data

Data Source:- Customer 360 Data

First Download dataset from link and then upload manually from local system to ADLS Gen 2 container project2 ----folder---bronze.

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: project3 / bronze

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status
[..]			
Agents.csv	4/29/2025, 8:52:38 PM	Hot (Inferred)	
Customers.csv	4/29/2025, 8:52:38 PM	Hot (Inferred)	
CustomerServiceInteractions.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	
InStoreTransactions.csv	4/29/2025, 8:52:38 PM	Hot (Inferred)	
LoyaltyAccounts.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	
LoyaltyTransactions.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	
OnlineTransactions.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	
Products.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	
Stores.csv	4/29/2025, 8:52:39 PM	Hot (Inferred)	

Step 2: Clean and Transform Data (Silver Layer – Curated) using Synapse Data Flow

First , I created 9 tables in SQL to save data.

```sql
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Address VARCHAR(255)
);

CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(100),
    Category VARCHAR(50),
    Price DECIMAL(10, 2)
);

CREATE TABLE OnlineTransactions (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    ProductID INT,
    DateTime DATETIME,
    PaymentMethod VARCHAR(50),
    Amount DECIMAL(10, 2),
    Status VARCHAR(20),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

```sql
CREATE TABLE Stores (
    StoreID INT PRIMARY KEY,
    Location VARCHAR(100),
    Manager VARCHAR(100),
    OpenHours VARCHAR(50)
);

CREATE TABLE InStoreTransactions (
    TransactionID INT PRIMARY KEY,
    CustomerID INT,
    StoreID INT,
    DateTime DATETIME,
    Amount DECIMAL(10, 2),
    PaymentMethod VARCHAR(50),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (StoreID) REFERENCES Stores(StoreID)
);

CREATE TABLE Agents (
    AgentID INT PRIMARY KEY,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Shift VARCHAR(50)
);
```

```sql
CREATE TABLE CustomerServiceInteractions (
    InteractionID INT PRIMARY KEY,
    CustomerID INT,
    DateTime DATETIME,
    AgentID INT,
    IssueType VARCHAR(50),
    ResolutionStatus VARCHAR(50),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (AgentID) REFERENCES Agents(AgentID)
);

CREATE TABLE LoyaltyAccounts (
    LoyaltyID INT PRIMARY KEY,
    CustomerID INT,
    PointsEarned INT,
    TierLevel VARCHAR(20),
    JoinDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE LoyaltyTransactions (
    LoyaltyID INT,
    DateTime DATETIME,
    PointsChange INT,
    Reason VARCHAR(100),
    PRIMARY KEY (LoyaltyID, DateTime),
    FOREIGN KEY (LoyaltyID) REFERENCES LoyaltyAccounts(LoyaltyID)
);
```

Tables created successfully.

CustomerID	Name	Email	Address

ProductID	Name	Category	Price

OrderID	CustomerID	ProductID	DateTime	PaymentMethod	Amount	Status

StoreID	Location	Manager	OpenHours

TransactionID	CustomerID	StoreID	DateTime	Amount	PaymentMethod

AgentID	Name	Department	Shift

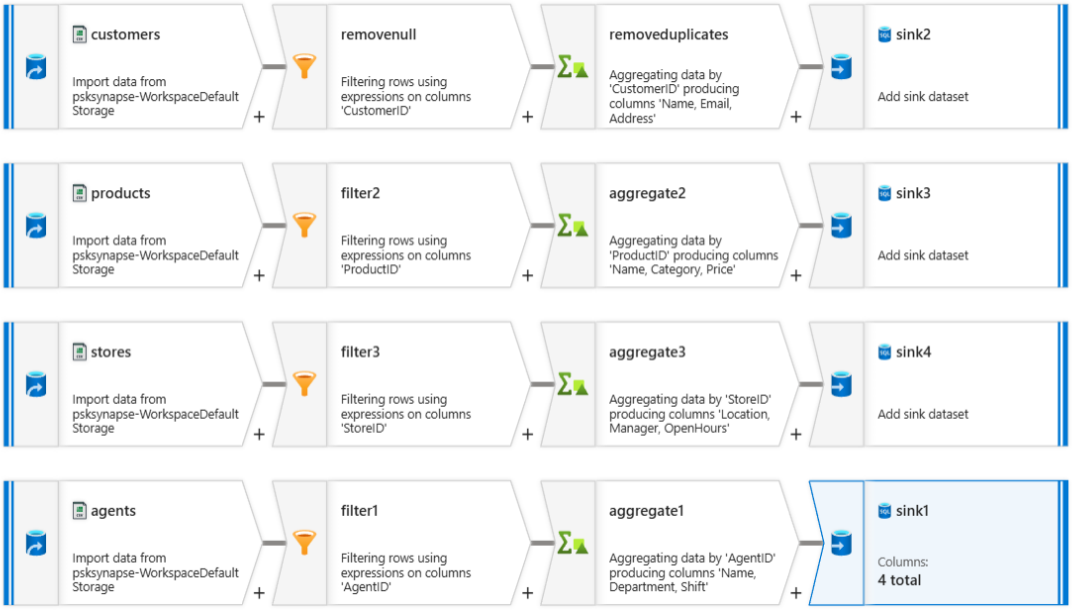
InteractionID	CustomerID	DateTime	AgentID	IssueType	ResolutionStatus

LoyaltyID	CustomerID	PointsEarned	TierLevel	JoinDate

LoyaltyID	DateTime	PointsChange	Reason

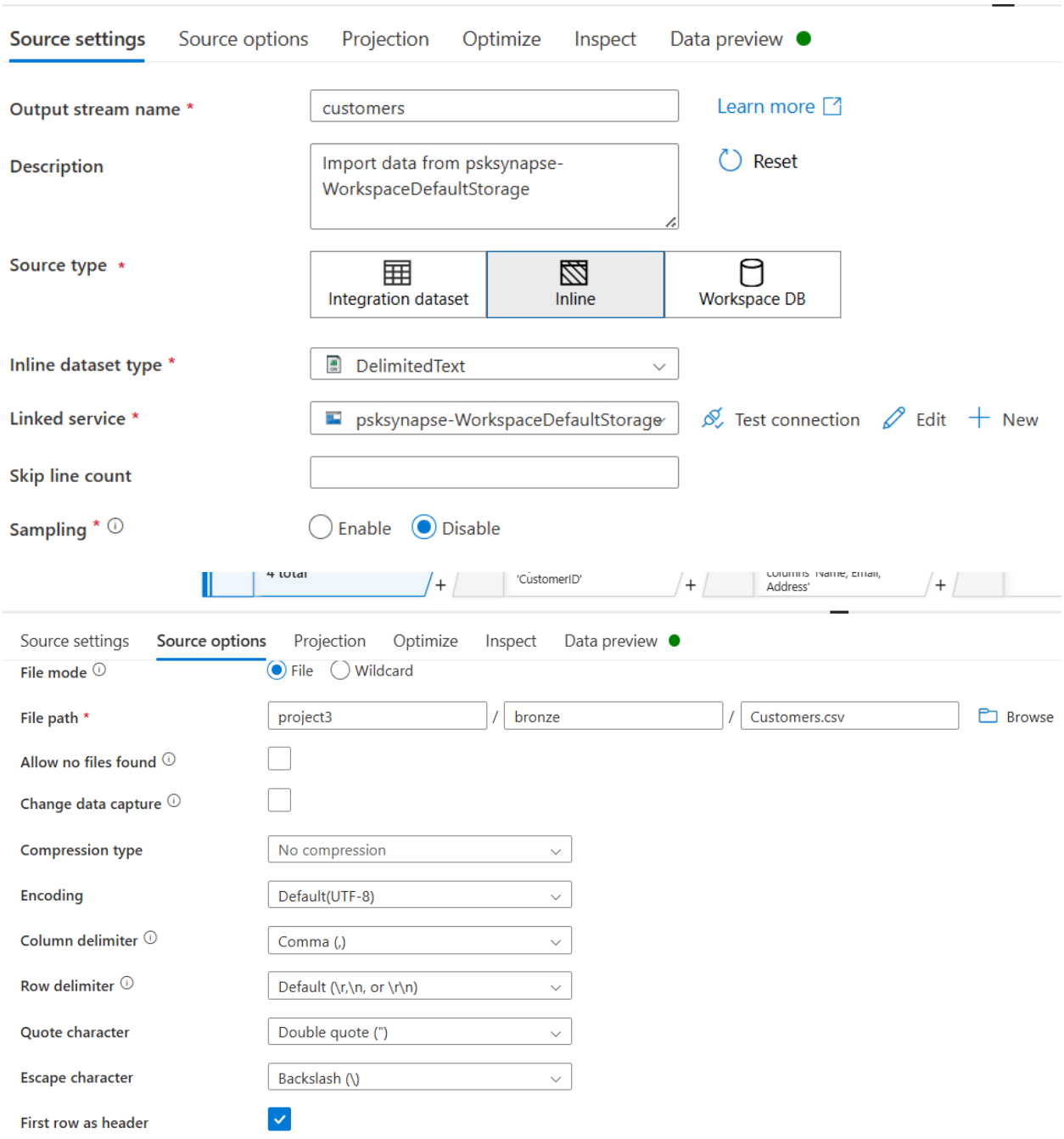
Dataflow -1

Here I selected 4 files which is customers , products , stores , agents to clean data process.



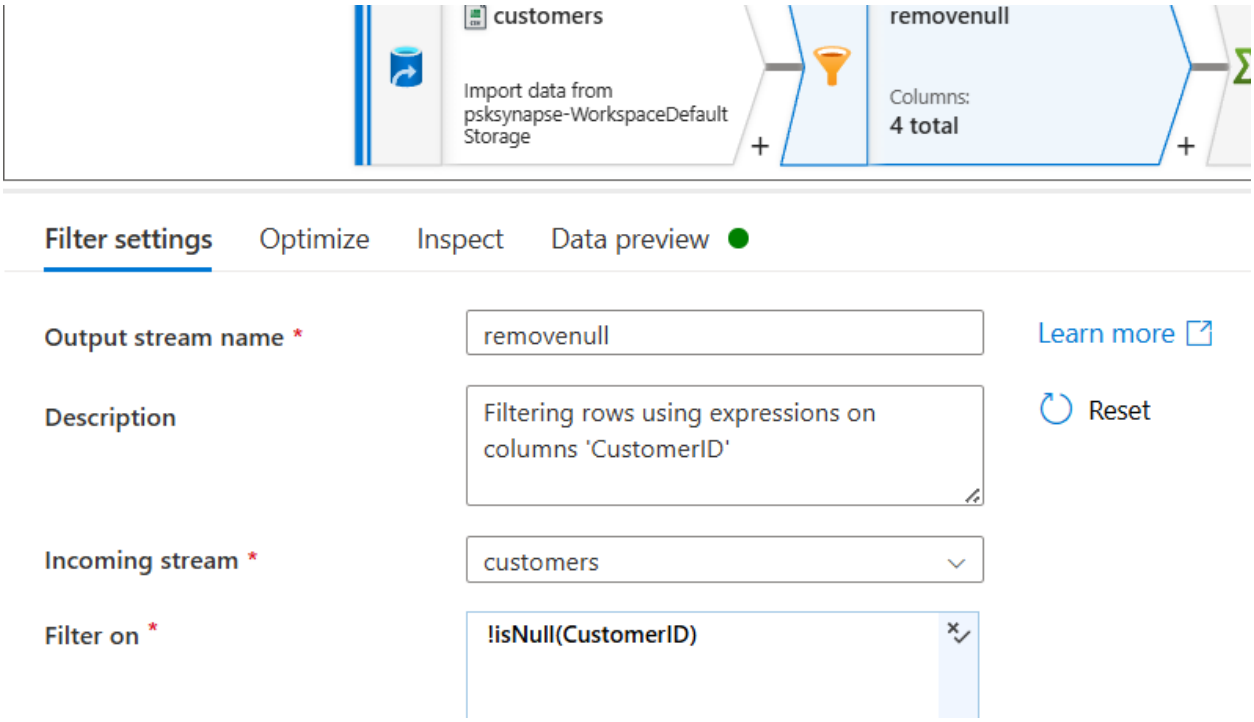
Step 1: Source (customers)

This step imports raw data from Azure Data Lake Storage Gen2 using the linked service `psksynapse-WorkspaceDefaultStorage`.



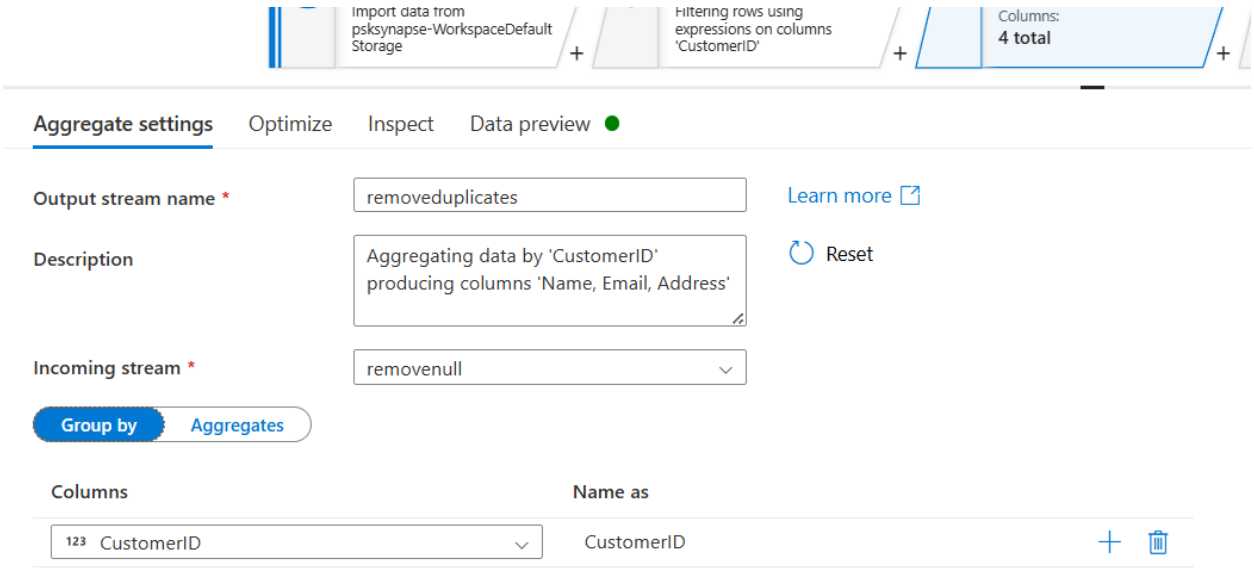
Step 2: Filter

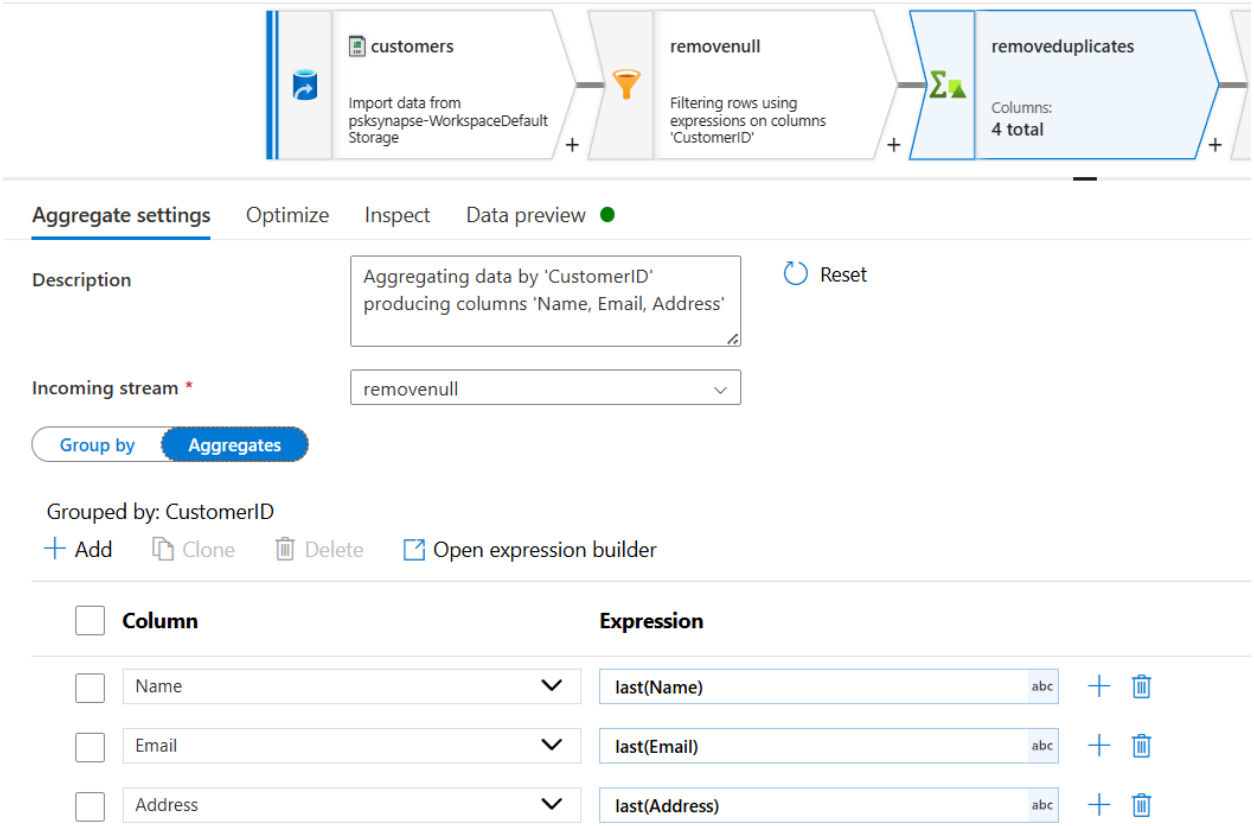
This transformation filters out rows where the primary key `CustomerID` is null. This ensures only valid records move to the next step.



Step 3: Aggregate (removeduplicates)

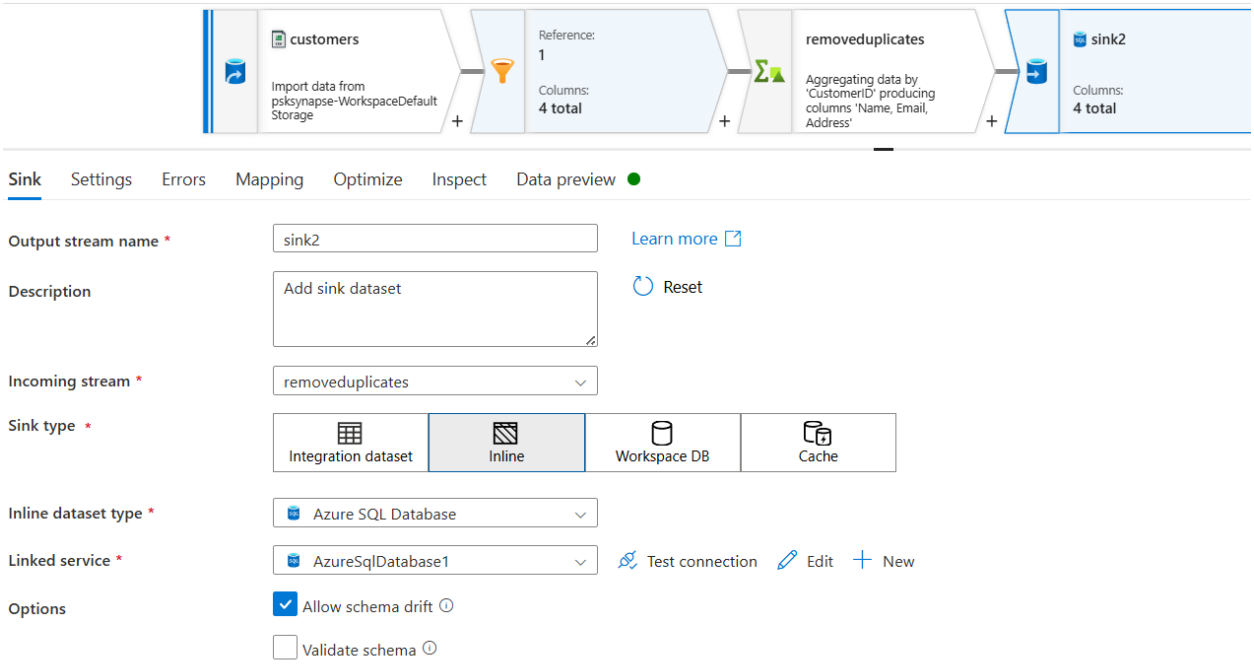
To remove duplicate records, the data is grouped by CustomerID and aggregation functions like `last()` are used for the other columns. This keeps the most recent value for Name, Email, and Address for each customer.

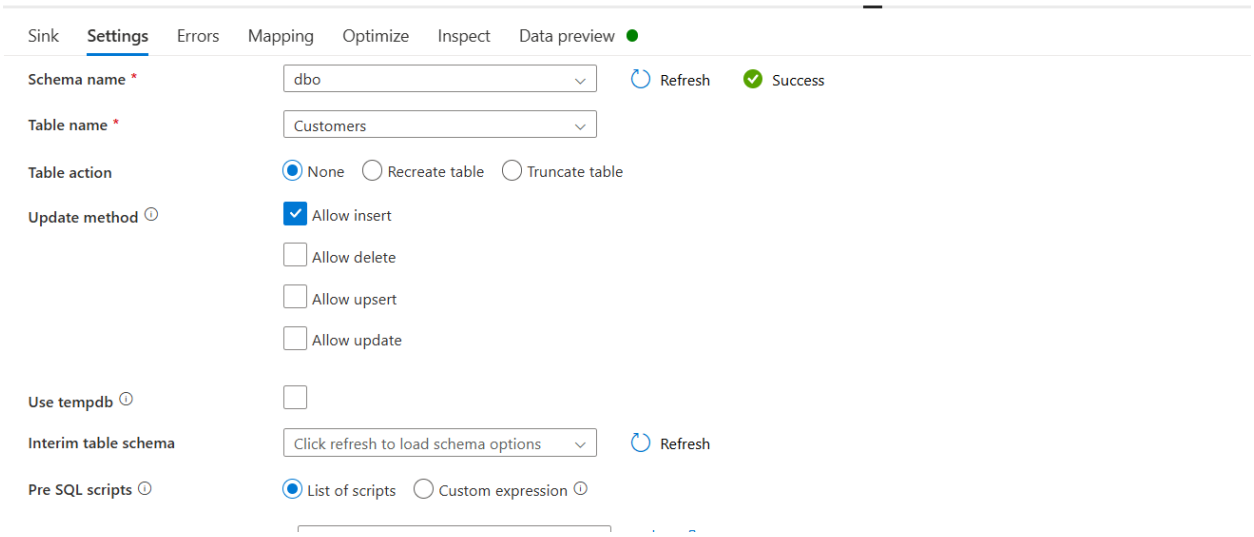




Step 4: Sink

The cleaned and deduplicated data is written into the SQL table `Customers` in the `dbo` schema.

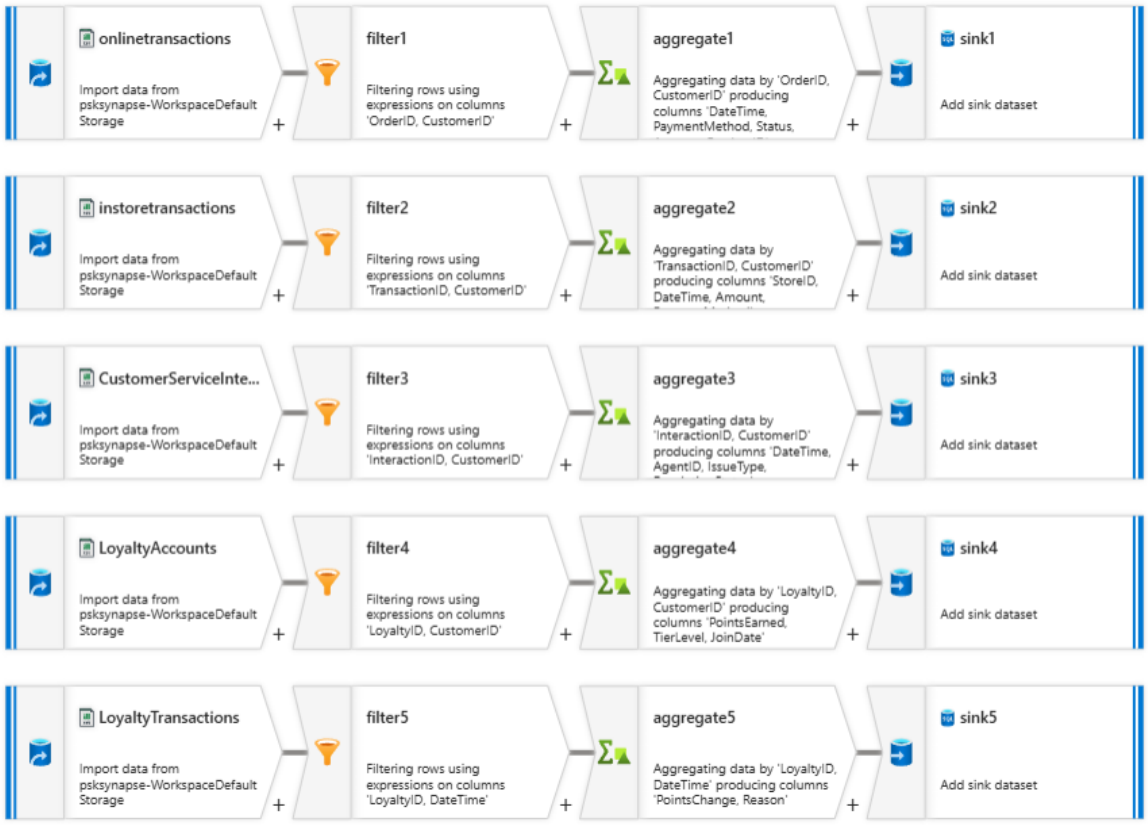




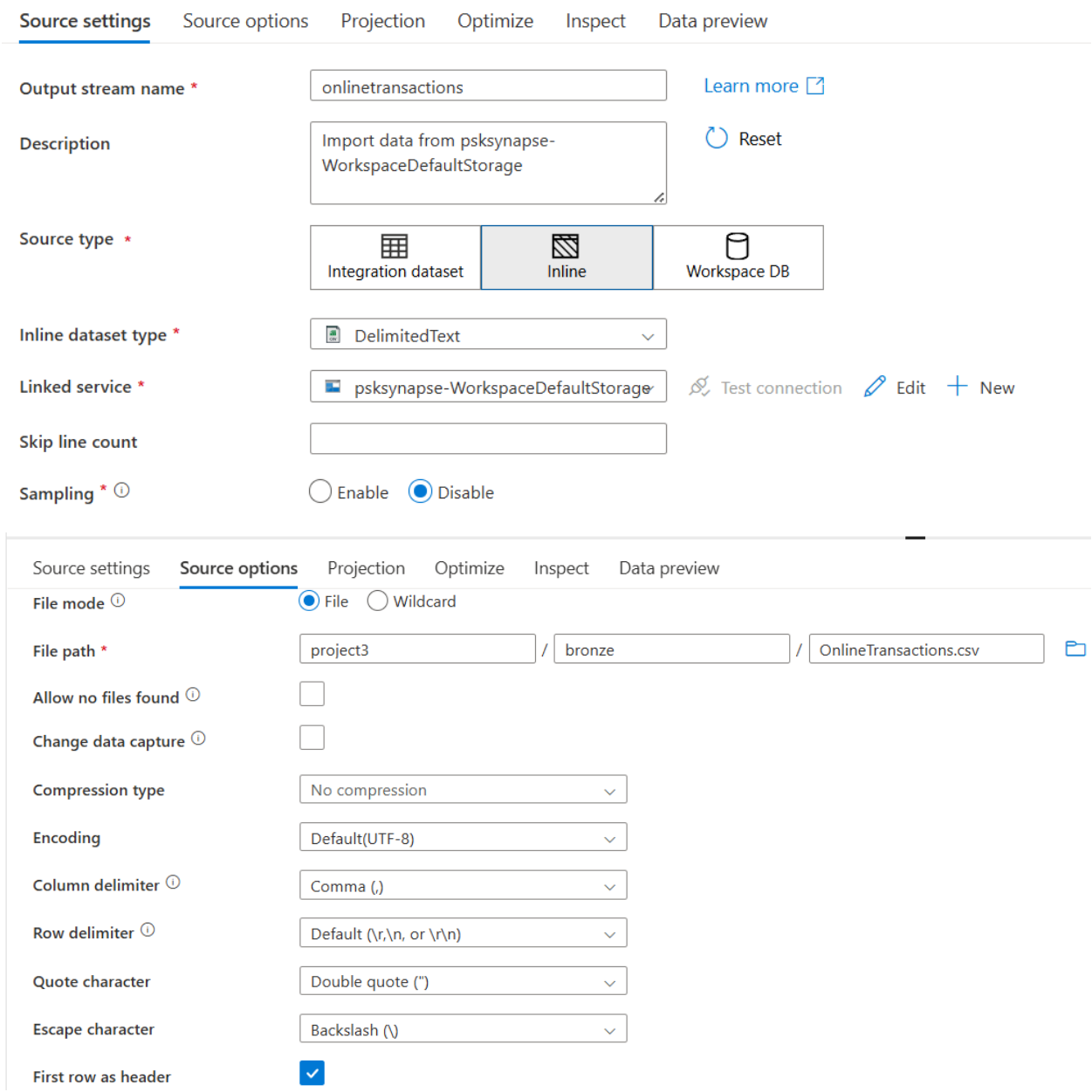
Similar steps are taken for this 3 files : products , stores , agents to clean data.

Dataflow-2

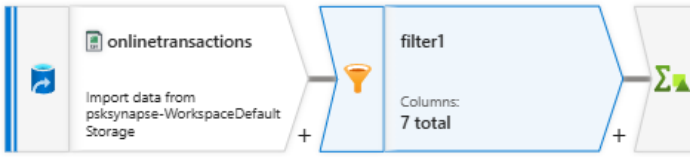
Here , I took 5 tables which are dependent on above tables data and perform same operations to clean data as well.

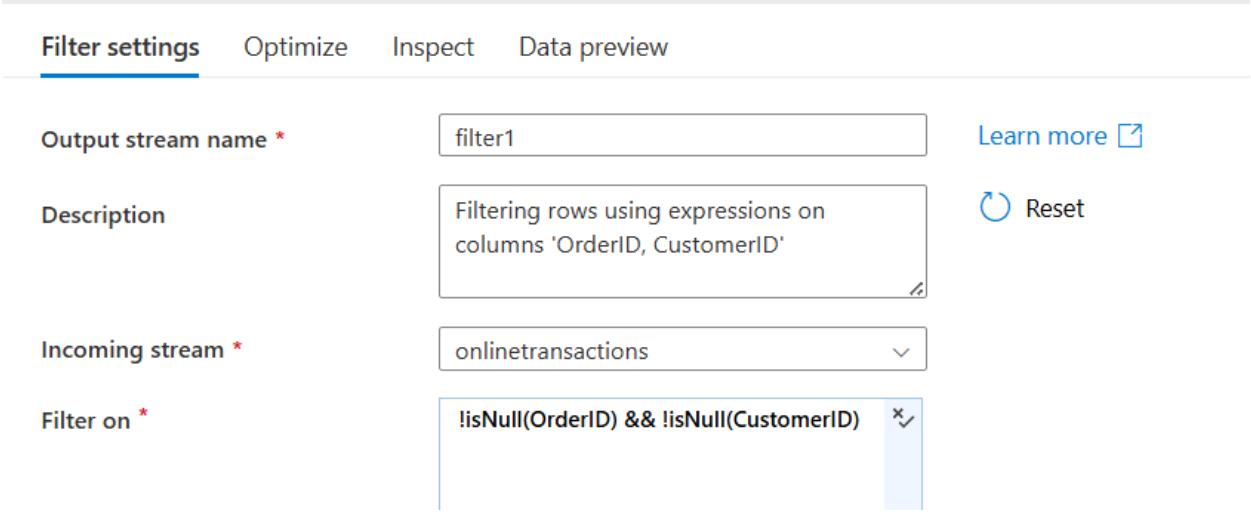


Step 1: Source (customers) : This step imports raw data from Azure Data Lake Storage.



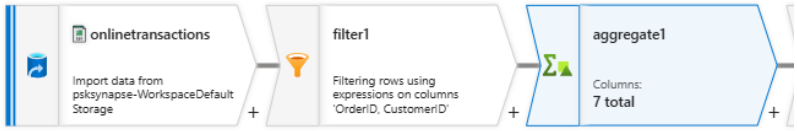
Step 2: Filter : remove null data from table

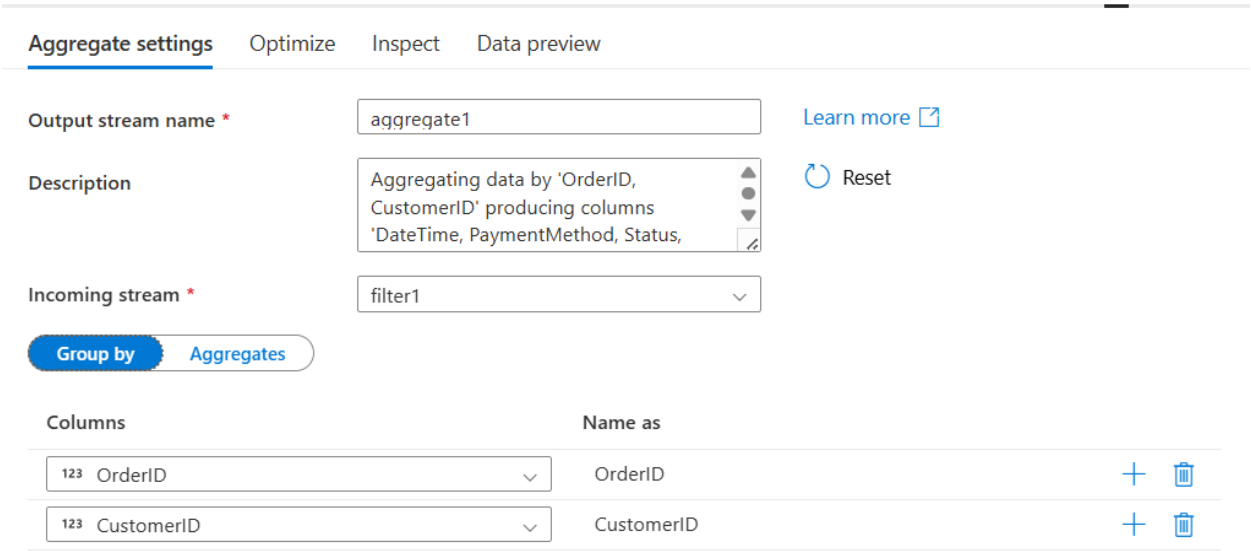


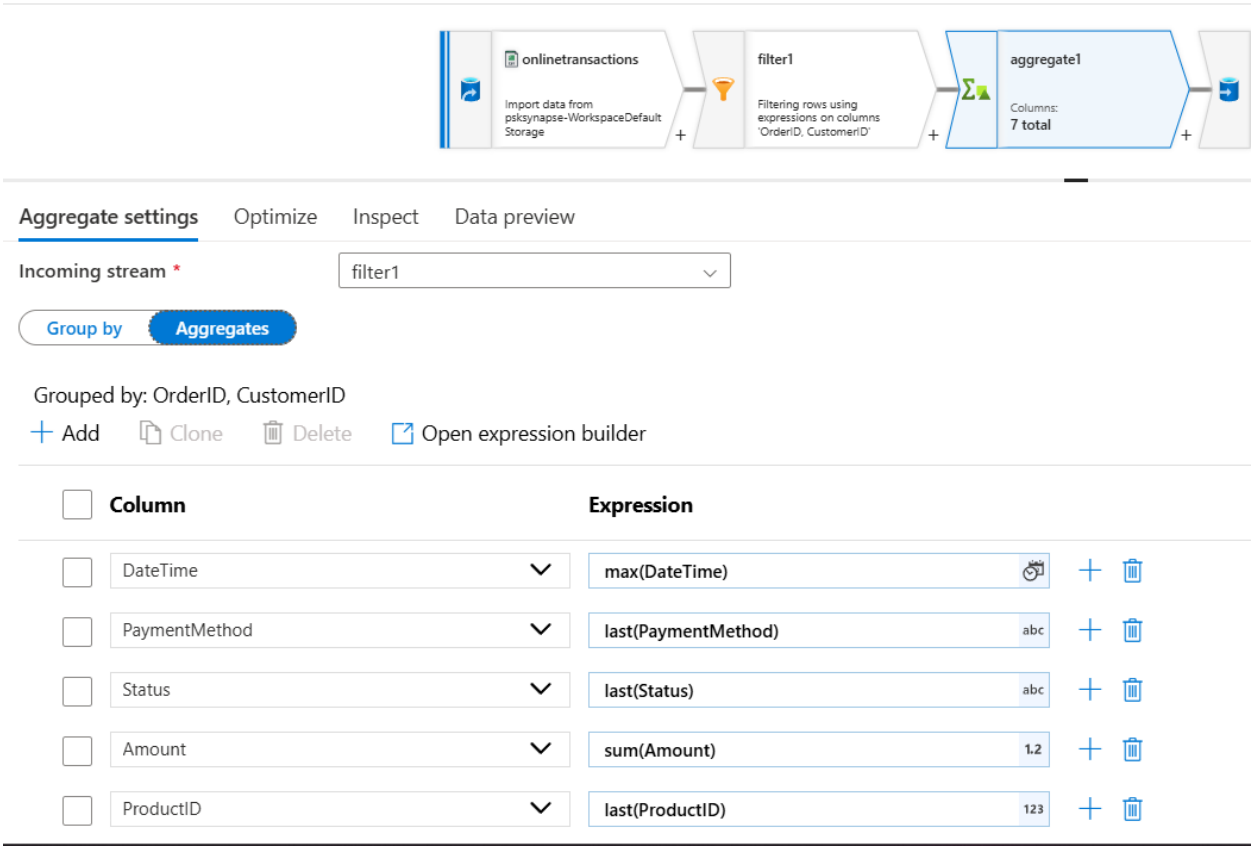


Step 3: Aggregate (removeduplicates)

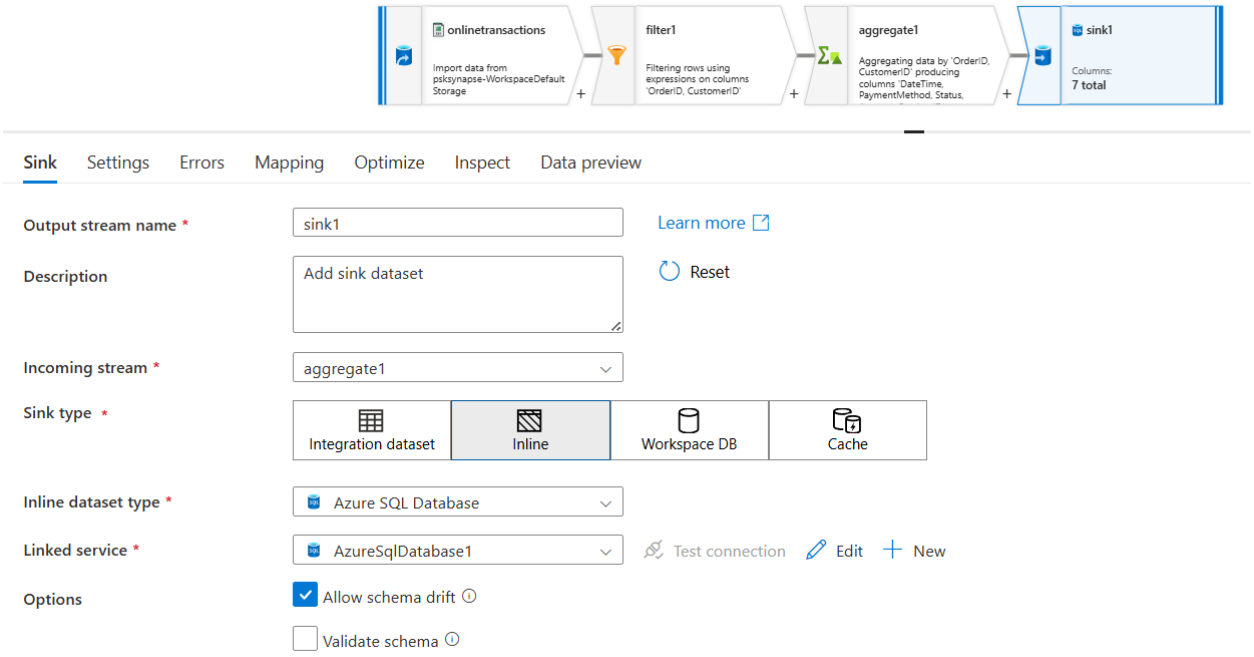
To remove duplicate records, the data is grouped by OrderID & CustomerID and aggregation functions like `last()` are used for the other columns.

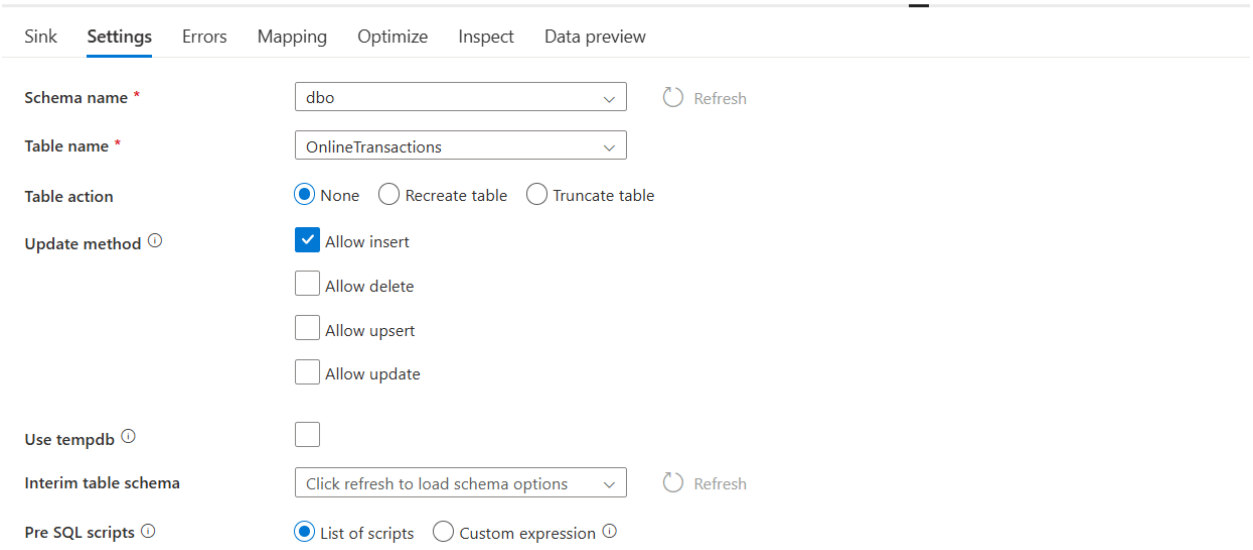






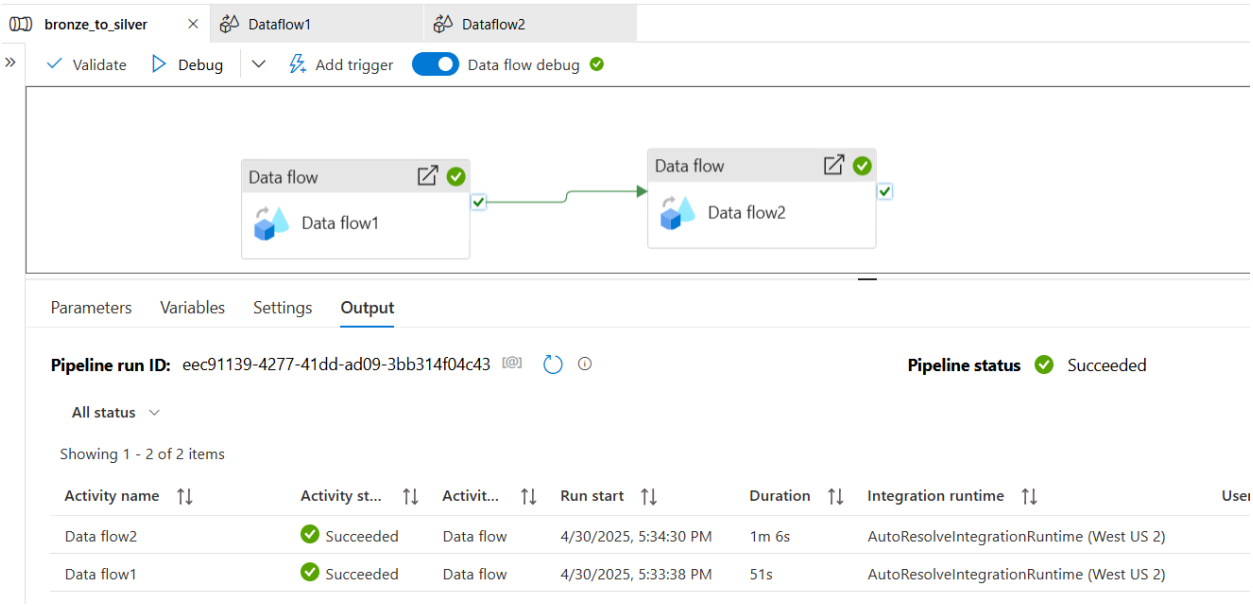
Step 4: Sink : The cleaned and deduplicated data is written into the SQL table.





similar steps are taken for other 4 files as per above mentioned dataflow.

Pipeline run Successfully.



All data added successfully in SQL tables :

110 %

Results | Messages

	CustomerID	Name	Email	Address
1	1	Mrs. Crystal Carroll	kaiserjacob@example.org	7566 Kelly Shoals Apt. 207, Port Joanne, FM 25401
2	2	Debra Newman ...	brownashley@example....	72713 Nelson Lodge Suite 286, Leonfort, NJ 49406
3	3	Thomas Mason	frankmark@example.com	3024 Riley Ferry Suite 573, Chadberg, MT 40822
4	4	Karla Hill	jnelson@example.org	9899 Hubbard Station, Lake Vanessa, CT 97851
5	5	Jeffrey Underwood	daniel48@example.org	11764 Hannah Plaza, Lake Vickichester, AR 85240
6	6	Justin Lowe	qwilson@example.org	494 Danielle Causeway, Lake Benjamin, NV 21217
7	7	Jason Daniels	elliottsonia@example.net	37283 Ramsey Light Suite 640, New Howard, IA 4...
8	8	Jennifer Hill	petersenrebecca@exam...	02411 Ruiz Corners Apt. 346, Oconnorberg, SD 9...

	ProductID	Name	Category	Price
1	1	Consider	Electronics	183.51
2	2	At	Toys	403.28
3	3	Risk	Clothing	186.53
4	4	Serious	Clothing	275.39
5	5	Discover	Clothing	23.39
6	6	Area	Books	316.84
7	7	Happy	Toys	137.58
8	8	Effect	Electronics	202.63

	OrderID	CustomerID	ProductID	DateTime	PaymentMethod	Amount	Status
1	1	34	91	2025-01-20 22:34:31.000	Gift Card	189.90	Completed
2	2	17	59	2025-02-10 20:42:13.000	Gift Card	193.25	Pending
3	3	13	57	2025-02-16 18:26:04.000	Debit Card	183.82	Pending
4	4	90	19	2025-02-21 05:04:59.000	Debit Card	175.37	Pending
5	5	61	99	2025-01-19 05:42:37.000	Credit Card	174.62	Completed
6	6	75	67	2025-02-26 09:47:16.000	Debit Card	144.48	Pending
7	7	63	44	2025-02-07 09:48:20.000	PayPal	91.74	Failed
8	8	55	6	2025-01-19 20:32:41.000	Credit Card	16.58	Failed

	StoreID	Location	Manager	OpenHours
4	4	Maytown	Frederick Dean	9:00 AM - 8:00 PM
5	5	Kathleenfort	Donald Ellis	10:00 AM - 7:00 ...
6	6	South Martin	Tamara Hernand...	10:00 AM - 6:00 ...
7	7	Rebeccastad	Amanda Diaz	9:00 AM - 8:00 PM
8	8	North Mariab...	Lisa Jackson	9:00 AM - 7:00 PM
9	9	Stephenfurt	Michael Hays	10:00 AM - 9:00 ...
10	10	Thompsonsh...	Mary Barker	9:00 AM - 9:00 PM
11	11	East Justin	Jose Brown	9:00 AM - 8:00 PM

	TransactionID	CustomerID	StoreID	DateTime	Amount	PaymentMethod
1	1	88	100	2025-03-12 06:49:37.000	135.74	Credit Card
2	2	60	12	2025-01-21 23:09:14.000	34.14	Credit Card
3	3	54	51	2025-01-29 22:12:51.000	111.42	Debit Card
4	4	32	94	2025-02-08 23:51:36.000	179.45	Debit Card
5	5	33	45	2025-01-18 04:05:21.000	177.96	Gift Card
6	6	92	91	2025-03-09 03:29:17.000	196.51	Debit Card
7	7	44	10	2025-02-19 19:28:55.000	198.12	Debit Card
8	8	91	51	2025-02-15 14:38:42.000	19.48	Gift Card

	AgentID	Name	Department	Shift
2	2	Terry Edwards	Billing	Evening
3	3	Garrett Knapp	Sales	Morning
4	4	Daryl Benjamin	Sales	Evening
5	5	Matthew Long	Sales	Morning
6	6	Patricia Rhodes	Sales	Morning
7	7	Elizabeth James	Technical Su...	Morning
8	8	Teresa Bennett	Technical Su...	Morning
9	9	Amber Ross	Billing	Afterno...

	InteractionID	CustomerID	DateTime	AgentID	IssueType	ResolutionStatus
4	4	30	2025-02-27 13:45:03.000	20	Other	Pending
5	5	49	2025-02-16 07:56:45.000	48	Technical Issue	Pending
6	6	42	2025-02-13 21:05:41.000	56	Product Inquiry	Resolved
7	7	73	2025-03-01 13:19:50.000	38	Other	Escalated
8	8	50	2025-01-15 16:44:36.000	66	Other	Escalated
9	9	63	2025-03-11 10:58:12.000	32	Complaint	Pending
10	10	21	2025-03-15 05:00:10.000	78	Product Inquiry	Escalated
11	11	69	2025-01-03 21:23:28.000	4	Complaint	Escalated

	LoyaltyID	CustomerID	PointsEarned	TierLevel	JoinDate
6	6	68	4291	Platinum	2020-09-25
7	7	90	2845	Bronze	2022-09-26
8	8	51	1697	Silver	2022-04-14
9	9	5	1912	Silver	2023-07-03
10	10	69	3846	Bronze	2024-09-28
11	11	53	2155	Silver	2022-04-24
12	12	86	4396	Bronze	2020-11-01
13	13	36	2411	Silver	2024-07-05

	LoyaltyID	DateTime	PointsChange	Reason
1	1	2025-01-13 20:04:21.000	166	Purchase
2	2	2025-02-28 18:14:23.000	190	Promotion
3	2	2025-03-04 21:11:47.000	-30	Promotion
4	6	2025-01-12 22:56:22.000	29	Referral
5	8	2025-02-21 03:46:32.000	122	Referral
6	9	2025-02-24 08:12:28.000	186	Correction

Step 3: Create 4 Key Views (Gold Layer Analytics)

View 1: Average Order Value

Show average money spent on each product, for each category and store location.

Explanation

1. Creates a new saved query named vw_AverageOrderValue
2. Selects columns from the Products and Stores tables to include product info and location.
3. Calculates average amount spent by dividing total amount (SUM) by total number of orders (COUNT).
4. Starts with the OnlineTransactions table as the base.
5. Joins to Products to get product name and category using ProductID.
6. Joins with in-store data using the same customer (optional depending on design).
7. Joins with Stores table to get store location using StoreID.
8. Groups results so the average is calculated per product and location.

```sql
CREATE VIEW vw_AverageOrderValue AS
SELECT
    p.ProductID,
    p.Name AS ProductName,
    p.Category,
    s.Location,
    SUM(o.Amount) / COUNT(o.OrderID) AS AverageOrderValue
FROM OnlineTransactions o
JOIN Products p ON o.ProductID = p.ProductID
JOIN InStoreTransactions i ON o.CustomerID = i.CustomerID
JOIN Stores s ON i.StoreID = s.StoreID
GROUP BY p.ProductID, p.Name, p.Category, s.Location;
```

110 %

Results | Messages

	ProductID	ProductName	Category	Location	AverageOrderValue
1	64	Subject	Electronics	Alexandraside	115.710000
2	86	Alone	Books	Alexandraside	187.640000
3	91	Drug	Clothing	Alexandraside	199.990000
4	23	Ahead	Home Goods	Ashleyfort	149.000000
5	67	Hand	Electronics	Ayersville	141.150000
6	14	Almost	Toys	Bakerberg	178.320000
7	34	Other	Clothing	Bakerberg	103.670000
8	13	Play	Home Goods	Bishopview	147.490000
9	41	Tend	Books	Chapmanhaven	17.410000
10	86	Alone	Books	Dariusberg	45.510000

View 2: Customer Segmentation

Group customers based on how much and how often they shop, and their loyalty level.

```sql
CREATE VIEW vw_CustomerSegments AS
SELECT
    c.CustomerID,
    SUM(o.Amount) AS TotalSpend,
    COUNT(o.OrderID) AS TotalOrders,
    l.TierLevel,
    CASE
        WHEN SUM(o.Amount) >= 1000 THEN 'High-Value'
        WHEN COUNT(o.OrderID) = 1 THEN 'One-Time Buyer'
        WHEN l.TierLevel = 'Gold' THEN 'Loyalty Champion'
        ELSE 'Regular'
    END AS Segment
FROM Customers c
LEFT JOIN OnlineTransactions o ON c.CustomerID = o.CustomerID
LEFT JOIN LoyaltyAccounts l ON c.CustomerID = l.CustomerID
GROUP BY c.CustomerID, l.TierLevel;
```

Explanation

1. Creates a view called vw_CustomerSegments.
2. Starts by selecting customer ID.
3. Adds up total amount spent by each customer.
4. Counts how many orders each customer placed.
5. Includes the customer's loyalty tier from the loyalty table.
6. Uses CASE to group customers into segments based on spend, frequency, or loyalty tier.
7. The main table is Customers.
8. Joins with OnlineTransactions so we can sum order amounts.
9. Joins with loyalty table to get tier info.
10. Groups by customer so that SUM/COUNT and segment label work per person.

	CustomerID	TotalSpend	TotalOrders	TierLevel	Segment
34	98	NULL	0	NULL	Regular
35	99	353.54	2	NULL	Regular
36	1	84.75	1	Bronze	One-Time Buyer
37	10	NULL	0	Bronze	Regular
38	12	315.70	2	Bronze	Regular
39	17	193.25	1	Bronze	One-Time Buyer
40	28	NULL	0	Bronze	Regular
41	29	NULL	0	Bronze	Regular
42	34	211.99	2	Bronze	Regular
43	41	29.44	2	Bronze	Regular
44	43	17.38	1	Bronze	One-Time Buyer
45	49	27.55	1	Bronze	One-Time Buyer
46	51	460.89	3	Bronze	Regular
47	69	NULL	0	Bronze	Regular
48	73	295.12	2	Bronze	Regular
49	83	49.44	1	Bronze	One-Time Buyer
50	86	NULL	0	Bronze	Regular
51	90	350.74	2	Bronze	Regular
52	94	104.86	1	Bronze	One-Time Buyer
53	3	NULL	0	Gold	Loyalty Champ...
54	5	147.49	1	Gold	One-Time Buyer
55	6	94.31	1	Gold	One-Time Buyer
56	8	90.37	1	Gold	One-Time Buyer
57	10	NULL	0	Gold	Loyalty Champ...

View 3: Peak Days and Hours

Shows when (day & hour) customers shop the most online and in-store.

```sql
CREATE VIEW vw_PeakTimes AS
SELECT
    'Online' AS Channel,
    DATENAME(WEEKDAY, DateTime) AS DayName,
    DATEPART(HOUR, DateTime) AS Hour,
    COUNT(*) AS TotalTransactions
FROM OnlineTransactions
GROUP BY DATENAME(WEEKDAY, DateTime), DATEPART(HOUR, DateTime)

UNION ALL

SELECT
    'InStore' AS Channel,
    DATENAME(WEEKDAY, DateTime) AS DayName,
    DATEPART(HOUR, DateTime) AS Hour,
    COUNT(*) AS TotalTransactions
FROM InStoreTransactions
GROUP BY DATENAME(WEEKDAY, DateTime), DATEPART(HOUR, DateTime);
```

Results | Messages

	Channel	DayName	Hour	TotalTransactions
1	Online	Monday	0	1
2	Online	Sunday	0	1
3	Online	Tuesday	0	1
4	Online	Friday	1	1
5	Online	Saturday	1	1
6	Online	Tuesday	1	1
7	Online	Monday	2	2
8	Online	Saturday	2	1
9	Online	Wednesday	2	1
10	Online	Saturday	3	2
11	Online	Sunday	3	1
12	Online	Tuesday	3	1
13	Online	Friday	4	1
14	Online	Thursday	4	2
15	Online	Tuesday	4	1
16	Online	Wednesday	4	2

Explanation

1. Creates a view called vw_PeakTimes.
2. Labels these rows as coming from the online channel.
3. Extracts day of the week (e.g., Monday) from date.
4. Extracts the hour from the time to show which hour was busiest.
5. Counts how many transactions happened during that hour and day.
6. Groups by both day and hour.
7. Adds results from in-store transactions below, combining both sources.
8. Same logic as above, just from InStoreTransactions, labeled as 'InStore'

View 4: Agent Performance

Count how many customer issues each agent handled and how many they resolved.

```sql
CREATE VIEW vw_AgentPerformance AS
SELECT
    a.AgentID,
    a.Name,
    COUNT(c.InteractionID) AS TotalInteractions,
    SUM(CASE WHEN c.ResolutionStatus = 'Resolved' THEN 1 ELSE 0 END) AS ResolvedInteractions
FROM CustomerServiceInteractions c
JOIN Agents a ON c.AgentID = a.AgentID
GROUP BY a.AgentID, a.Name;

select *from vw_AgentPerformance;
```

110 %

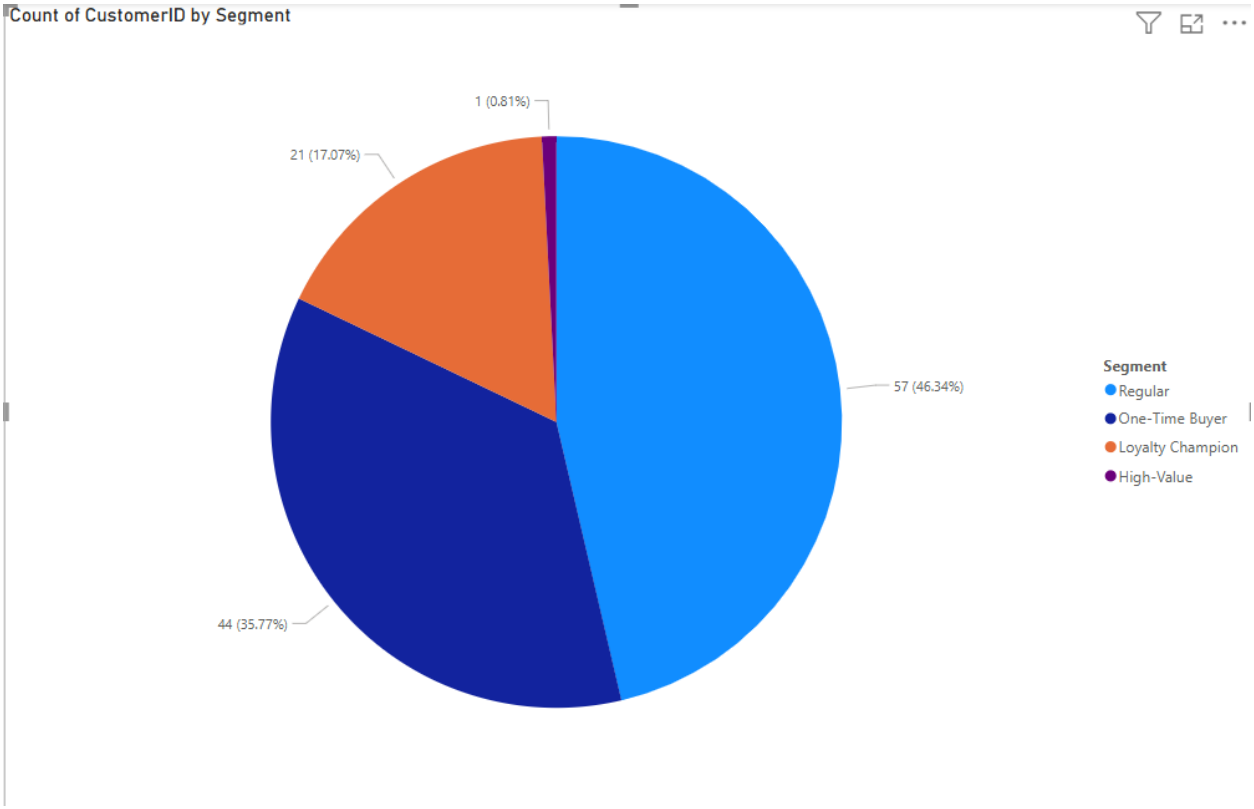
Results | Messages

	AgentID	Name	TotalInteractions	ResolvedInteractions
1	1	Jonathan Williams	1	0
2	2	Terry Edwards	1	0
3	4	Daryl Benjamin	2	0
4	5	Matthew Long	1	0
5	7	Elizabeth James	2	2
6	8	Teresa Bennett	1	0
7	9	Amber Ross	2	1
8	10	Tonya Jones	1	0
9	11	Curtis Mcbride	1	1
10	13	Scott Flowers	1	0
11	15	Kristen Crawford	1	1
12	17	Brandon Jimenez	1	0
13	18	Melissa White	2	0
14	19	Eddie Pierce	1	0
15	20	Julia Owens	1	0
16	21	Cindy Gomez	3	2
17	27	Shawn Gill	2	1
18	28	Ricky Davenport	1	0
19	29	Jessica Mora	2	1
20	31	Kimberly Chamb...	2	0
21	32	Debbie Stewart	2	1
22	33	Jimmy Weber	1	1
23	34	Heather Harrison	4	1

Explanation

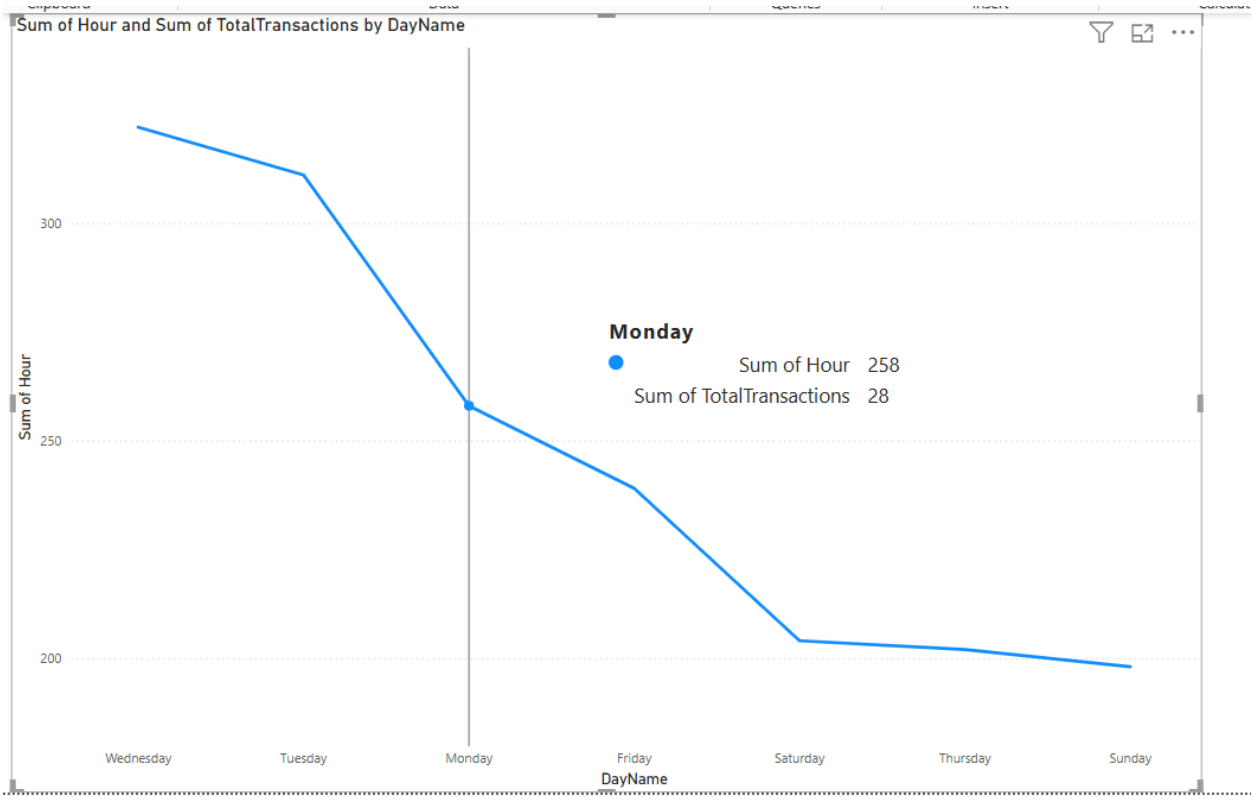
1. Creates a view called vw_AgentPerformance.
2. Gets agent ID and name.
3. Counts how many customer service cases each agent worked on.
4. Counts how many cases were marked as "Resolved".
5. Joins the CustomerServiceInteractions with the Agents table.
6. Groups by agent to get total and resolved counts per person.

Power BI Visuals



It groups customers by their **segment** (based on their behavior/spending/loyalty).

The chart shows **how many customers** fall into each segment and their **percentage share** of the total.



It's helpful for analyzing **peak days** and **transaction intensity**.

Day	Interpretation
Wednesday & Tuesday	Highest activity — peak days in terms of overall engagement.
Monday	Start of noticeable drop in activity.
Friday to Sunday	Low activity — suggests quiet end to the week.