

Bootcamp Project - 1

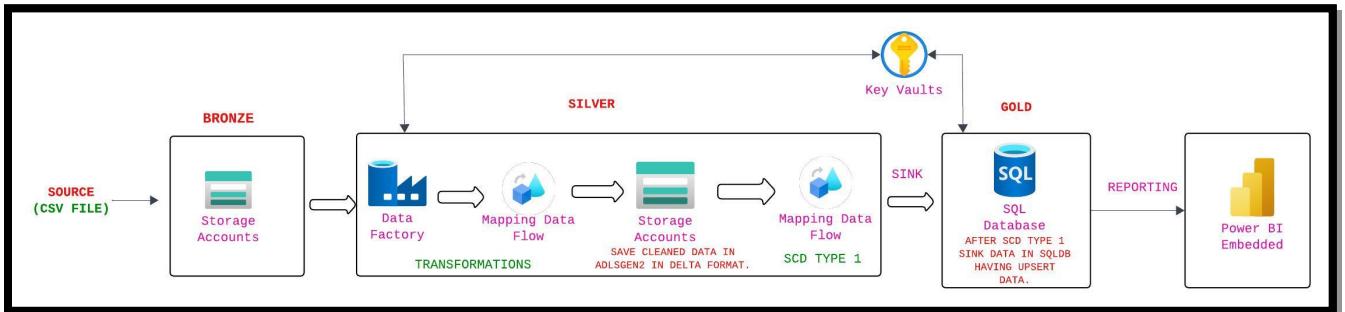
Data Pipeline for Customer Account Analysis

Objective

The project aims to design and implement a robust data pipeline for processing customer account data. This includes copying data from a backend team's storage account, performing necessary transformations using datafactory, and upserting (inserting or updating) (SCD-1) data from a file stored in Azure Data Lake Storage (ADLS) GOLD Storage into a SQL DB. The pipeline aims to ensure efficient, accurate, and scalable data processing to support downstream analytics and reporting needs.

NANDINI RATHORE

ARCHITECTURE OF PROJECTÀ



This screenshot shows the Microsoft Azure Storage accounts interface. The user is viewing the **nandiniadls | Containers** page for the storage account **nandiniadls**. On the left, there is a sidebar with various options like Create, Restore, and a message about the new browser experience. The main area displays a table of containers:

Name	Last modified	Anonymous access level	Lease state
<input checked="" type="checkbox"/> bronze	23/07/2025, 13:34:45	Private	Available
<input checked="" type="checkbox"/> silver	23/07/2025, 13:35:17	Private	Available
<input checked="" type="checkbox"/> gold	07/08/2025, 22:24:59	Private	Available

The table is sorted by Name in ascending order. The container names **bronze**, **silver**, and **gold** are highlighted with blue checkboxes. The interface also includes a search bar, upload and refresh buttons, and other navigation links like Overview, Activity log, Tags, and Diagnose and solve problems.

Creating 3 container in adlsgen2.-->bronze,silver,gold.

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar shows the 'bronze' container under 'Containers'. The main area displays a list of blobs with the following details:

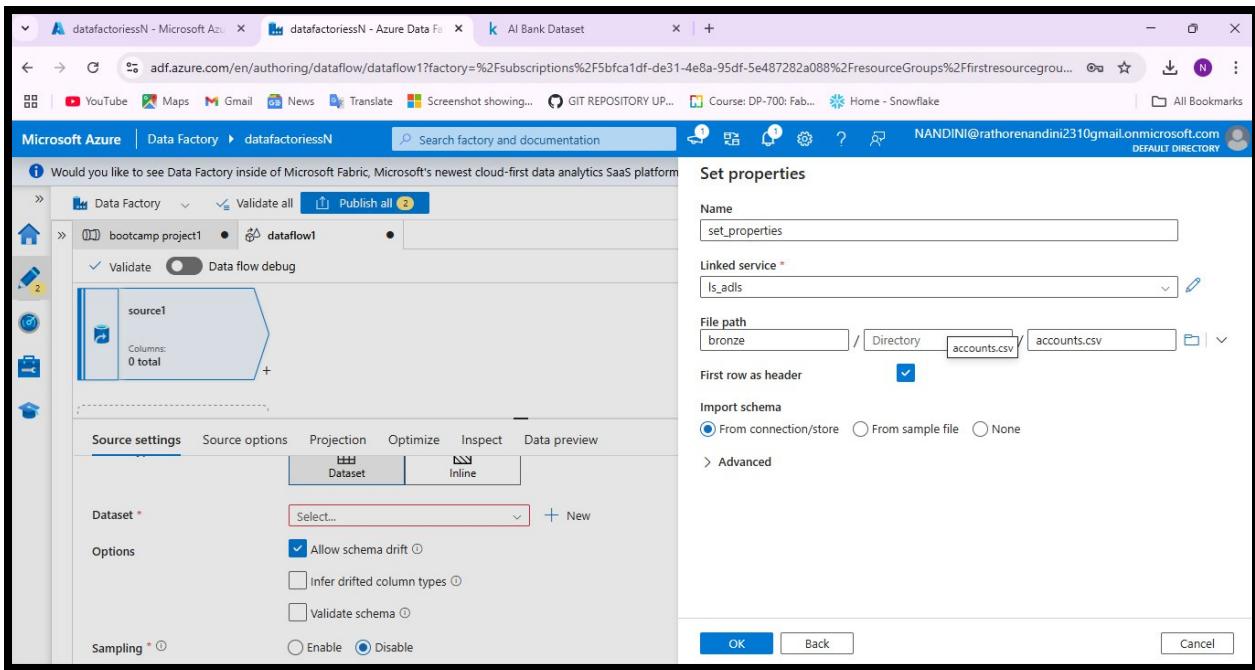
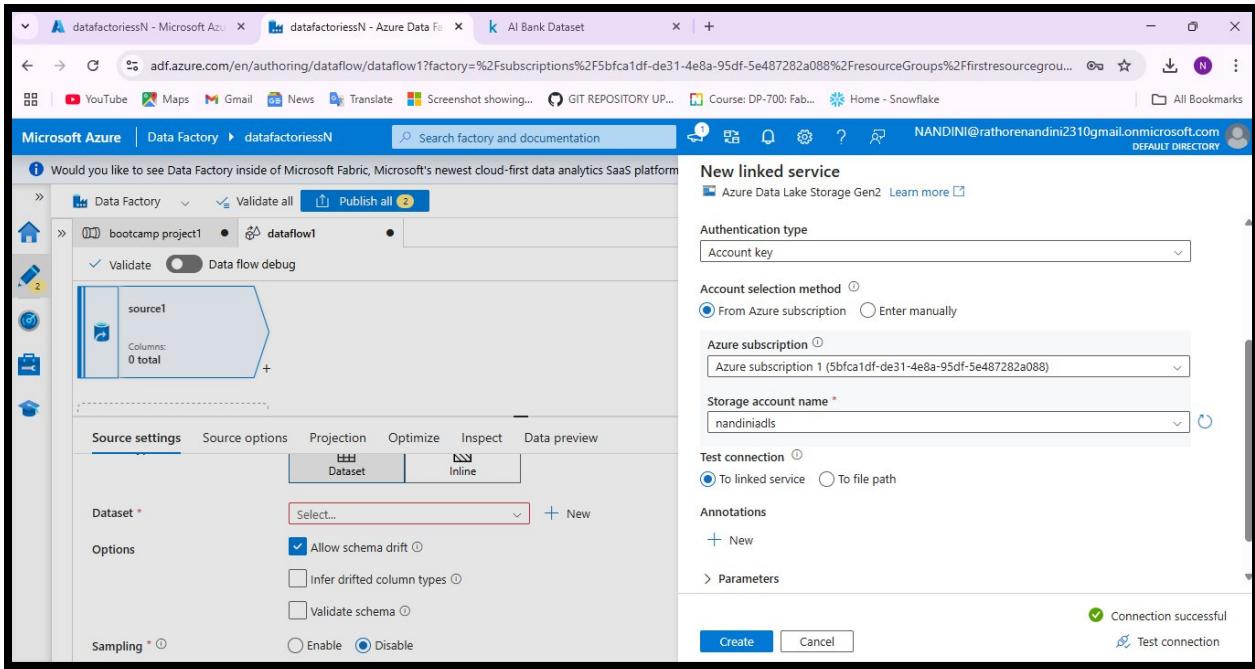
Name	Last modified	Access tier	Blob type	Size	Lease state
accounts.csv	07/08/2025, 22:35:19	Hot (Inferred)	Block blob	2.28 KiB	Available
customers.csv	07/08/2025, 22:35:19	Hot (Inferred)	Block blob	4.5 KiB	Available
loan_payments.csv	07/08/2025, 22:35:19	Hot (Inferred)	Block blob	2.55 KiB	Available
loans.csv	07/08/2025, 22:35:19	Hot (Inferred)	Block blob	2.29 KiB	Available
transactions.csv	07/08/2025, 22:35:19	Hot (Inferred)	Block blob	3.43 KiB	Available

Ingesting raw data in bronze layer.

The screenshot shows the Microsoft Azure Data Factory 'New dataset' configuration dialog. On the left, there's a preview of a data flow with a 'source1' component. On the right, the 'Select a data store' dialog is open, showing various options:

- All
- Azure
- Database
- File
- Generic protocol
- Amazon S3
- Azure Blob Storage
- Azure Data Explorer (Kusto)
- Azure Data Lake Storage Gen2
- Azure Database for MySQL
- Azure Cosmos DB IoT NoSQL

Select source dataset.



NANDINI RATHORE

The screenshot shows the Microsoft Azure Data Factory interface. A data flow named 'dataflow1' is selected. The 'Projection' tab is active, showing four columns: account_id, customer_id, account_type, and balance. Each column has its type defined (e.g., short for account_id, string for account_type) and a 'Format' dropdown set to 'Specify format'.

The screenshot shows the Microsoft Azure Data Factory interface with the 'Data preview' tab selected for the 'dataflow1' data flow. The preview area displays 10 rows of sample data:

account_id	customer_id	account_type	balance
1	45	Savinas	1000.50
2	12	Checking	2500.75
3	78	Savinas	1500.00
4	34	Checking	3000.25
5	56	Savinas	500.00
6	23	Checking	1200.50
7	89	Savinas	800.75
8	67	Checking	2200.00
9	14	Savinas	900.25
10	92	Checking	1800.50

Add conditional splitting, split into two streams clean data and null values.

NANDINI RATHORE

The screenshot shows the Microsoft Azure Data Factory Dataflow expression builder. The top navigation bar includes tabs for 'datafactoryessN - Microsoft Azure', 'datafactoryessN - Azure Data Fl...', 'AI Bank Dataset', 'Schema changes explanation', and a plus sign for creating new components. Below the navigation is a search bar and a user profile for 'NANDINI@rathorenandini2310@gmail.onmicrosoft.com'. The main area is titled 'Dataflow expression builder' and contains sections for 'Expression elements' (All, Functions, Input schema), 'Expression values' (with a filter input), and 'Data preview'. The 'Data preview' section shows an output table with columns 'customer_id' and 'balance'. The table data is as follows:

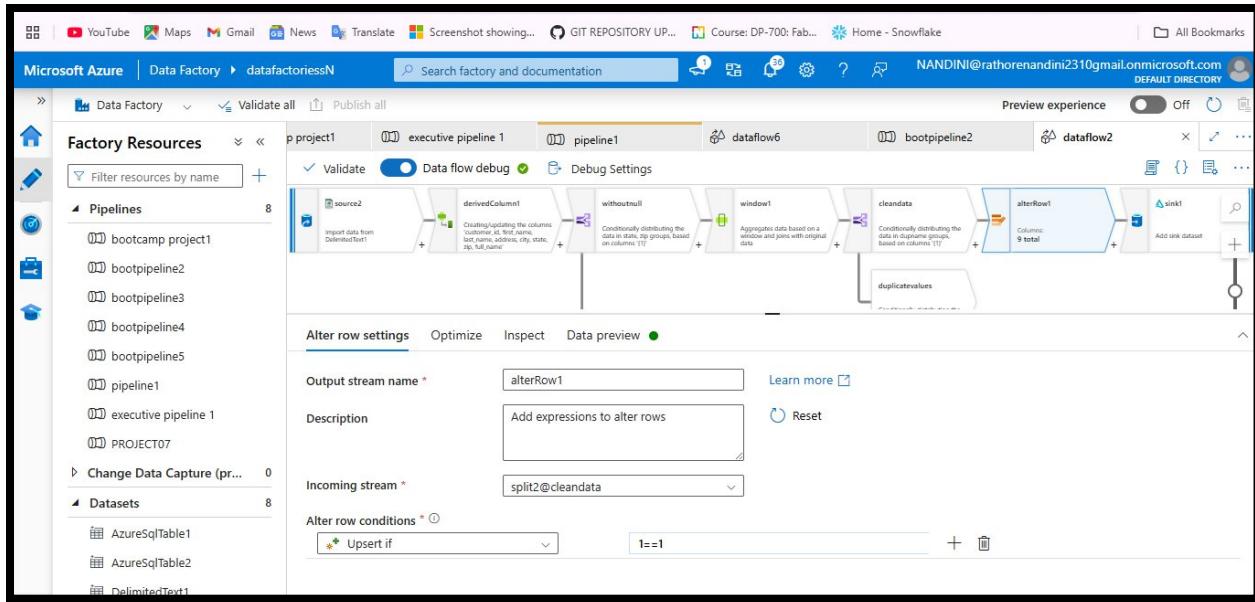
	customer_id	balance
✓	45	1000.5
✓	12	2500.75
✓	78	1500.0

At the bottom are buttons for 'Save and finish', 'Cancel', and 'Clear contents'.

The screenshot shows the Microsoft Azure Data Factory Data preview interface for a dataflow component named 'dataflow1'. The top navigation bar is identical to the previous screenshot. The main area displays a preview of the data being processed by the dataflow. The preview table has columns 'account_id', 'customer_id', 'account_type', and 'balance'. The data rows are:

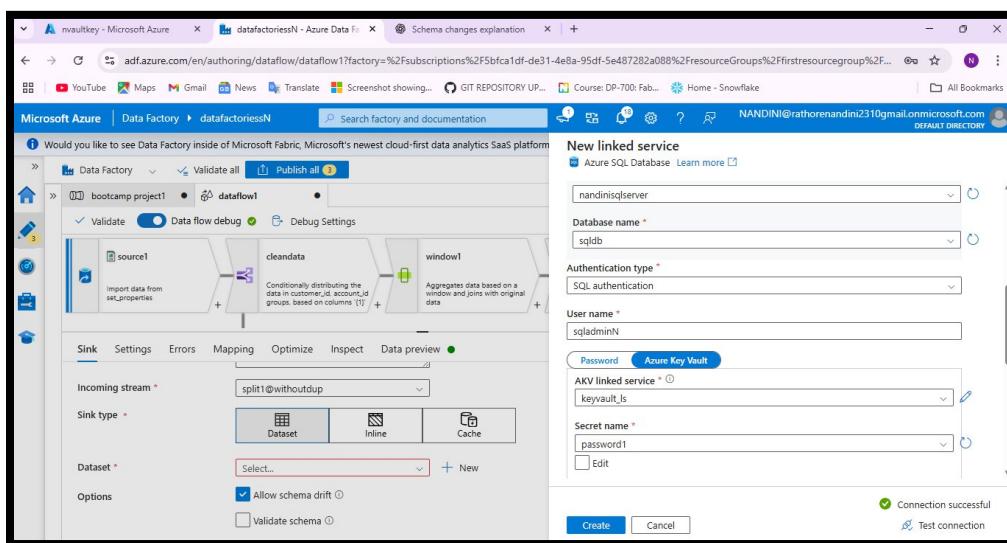
account_id	customer_id	account_type	balance
1	45	Savings	1000.5
2	12	Checking	2500.75
3	78	Savings	1500.0
4	34	Checking	3000.25
5	56	Savings	500.0
6	23	Checking	1200.5
7	89	Savings	800.75
8	67	Checking	2200.0

On the right side, there is a 'Properties' panel with tabs for 'General' and 'Related (1)'. The 'General' tab shows the 'Name' field set to 'dataflow1' and an empty 'Description' field.



alter row.

Using key vault to save secrets.



The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists various pipelines and datasets. The main area displays a data flow named 'pipeline1'. The data flow consists of several stages: 'source2' (Import data from DelimitedText), followed by 'derivedColumn1' (Creating/updating the columns 'customer_id', 'first_name', 'last_name', 'address', 'city', 'state', 'zip', 'full_name'), 'withoutnull' (Conditionally distributing the data in state, zip groups, based on columns '1-7'), 'window1' (Aggregating data based on a window and joins with original data), 'cleandata' (Conditionally distributing the data in duplicate groups, based on columns '1-7'), and finally 'sink1' (Add expressions to alter rows). Below the data flow, the 'Sink' tab is selected, showing configuration for the 'sink1' output stream. The 'Description' field is set to 'Add sink dataset'. The 'Incoming stream' is 'alterRow1'. The 'Sink type' is set to 'Dataset'. The 'Inline dataset type' is 'Delta'. The 'Linked service' is 'ls adls'. A note at the top of the sink configuration panel states: 'This sink currently has Single partition set in Optimize. This will make your data flow execution longer. The recommended setting is Use current partitioning.'

Sink csv file into delta format in silver layer.

SOURCE 2—CUSTOMER.

This screenshot shows the same Microsoft Azure Data Factory interface as the previous one, but the focus is on the 'Source settings' tab for the 'source2' component in the data flow 'pipeline1'. The 'Source settings' tab is active, showing the following configuration: 'Output stream name' is 'source2', 'Description' is 'Import data from DelimitedText1', 'Source type' is 'Dataset', 'Dataset' is 'DelimitedText1', and 'Options' include 'Allow schema drift'. The rest of the data flow and pipeline structure are visible in the background.

The screenshot shows the Microsoft Azure Data Factory Data Preview interface. On the left, there's a sidebar with icons for Home, Data Factory, Data Flow, Data Pipeline, Data Integration, and Data Governance. The main area shows a project named "bootcamp project1" with a data flow named "dataflow1". The "set_properties" step is selected. The "Data preview" tab is active, displaying a table of data with columns: customer_id, first_name, last_name, address, city, state, and zip. The data consists of 10 rows of customer information. To the right, there's a "Properties" panel with tabs for General and Related (1), showing the name "dataflow1" and a description field.

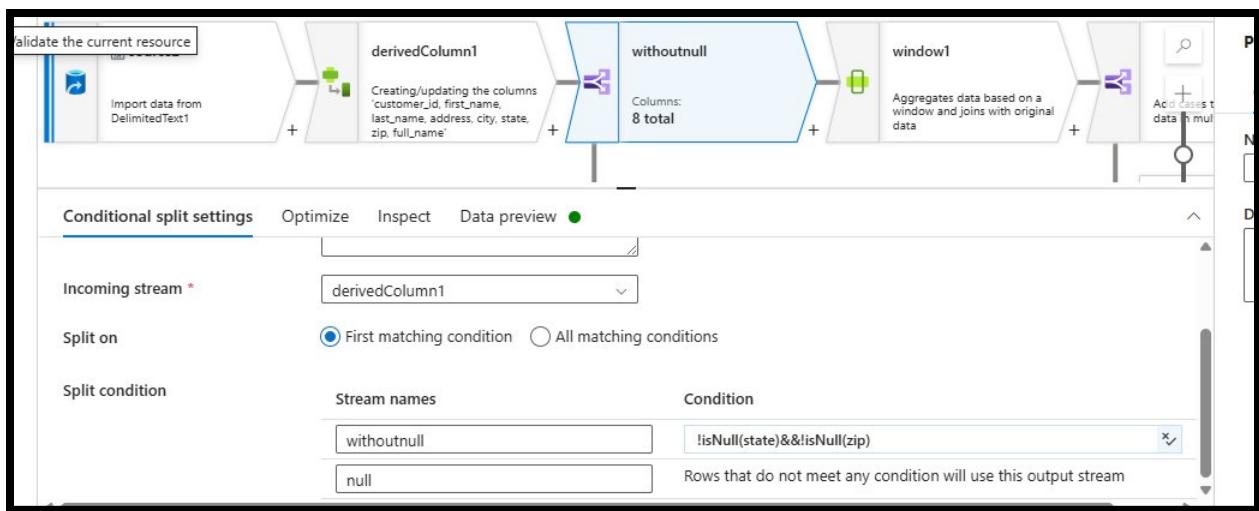
customer_id	first_name	last_name	address	city	state	zip
1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1

The screenshot shows the Microsoft Azure Data Factory Dataflow expression builder interface. It displays a "Dataflow expression builder" window with a "derivedColumn3" node. Under "Derived Columns", there's a "Create new" button and a "full_name" entry. The "Expression" section shows the expression "concat(first_name,' ',last_name)". Below it, there are tabs for "Expression elements" and "Expression values". The "Data preview" section shows the output of the derived column: "full_name abc" with rows "JohnDoe" and "JaneSmith", and corresponding "first_name" and "last_name" columns. At the bottom, there are "Save and finish", "Cancel", and "Clear contents" buttons.

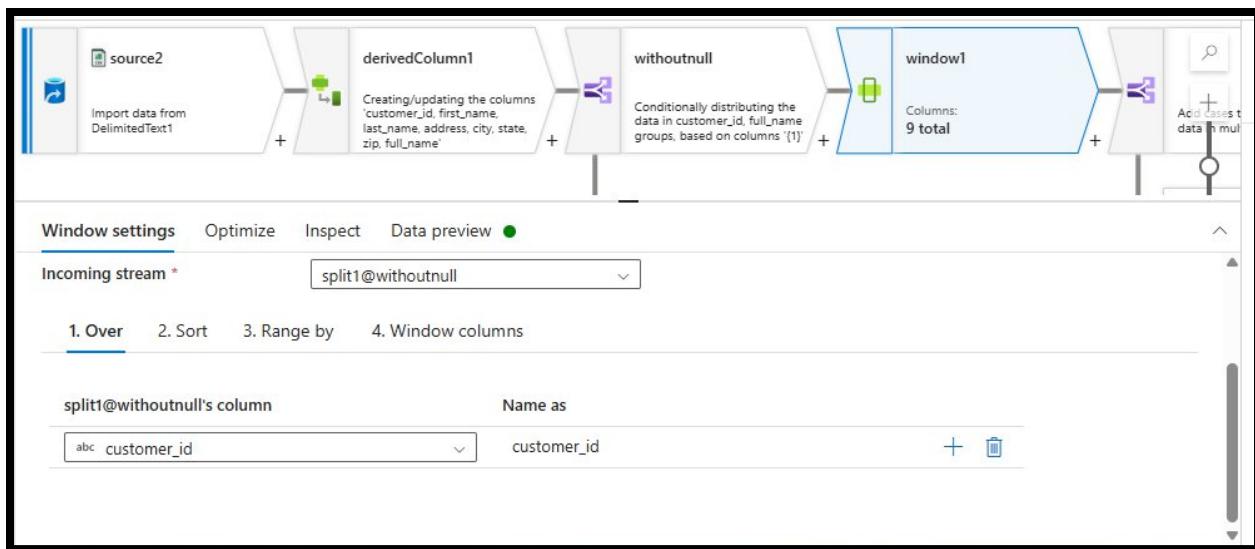
Using derived column to generate a new column-full name.

The screenshot shows the Data Flow preview interface in Azure Data Factory. The top navigation bar includes 'Data Factory', 'Validate all', 'Publish all', 'Preview experience' (Off), and other options. The main area displays a table of data with the following columns and rows:

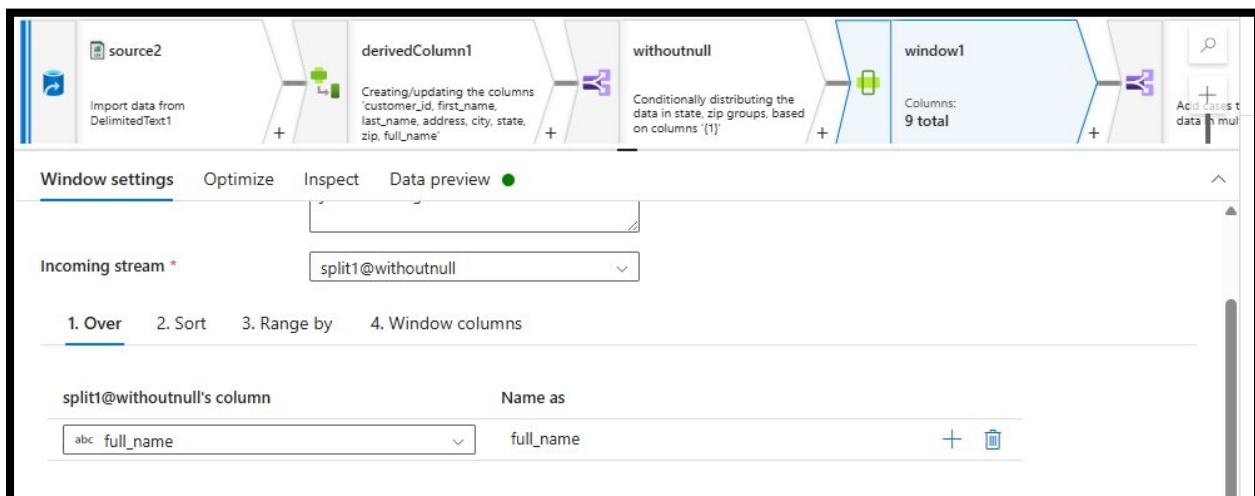
	customer_id	first_name	last_name	address	city	state	zip	full_name
+	78	Abiqail	Cole	7777 Fir St	Sundridge	ON	P0A0A1	AbiqailCole
+	52	Abiqail	Henderson	5151 Cypress Ave	Mount Albert	ON	L0G0A1	AbiqailHenderson
+	26	Abiqail	Parker	2525 Poplar St	Barrie	ON	L4M0A1	AbiqailParker
+	37	Alexander	Bell	3636 Redwood Dr	Stratford	ON	N5A0A1	AlexanderBell
+	63	Alexander	Foster	6262 Spruce Ln	Stayner	ON	L0M0A1	AlexanderFoster
+	11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1	AlexanderThomas
+	18	Amelia	Adams	1717 Oak Dr	Saskatoon	SK	S7K0A1	AmeliaAdams
+	70	Amelia	Hayes	6969 Beech Dr	Waubashene	ON	L0K0A1	AmeliaHayes

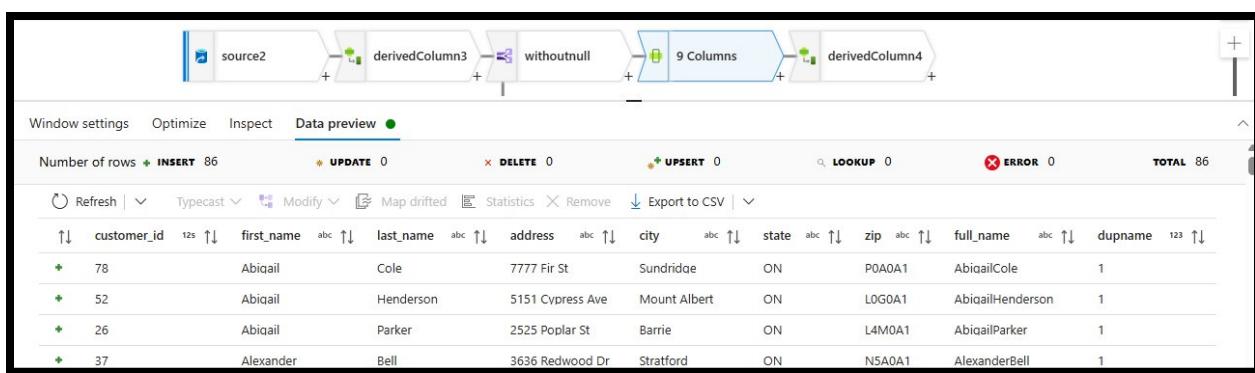
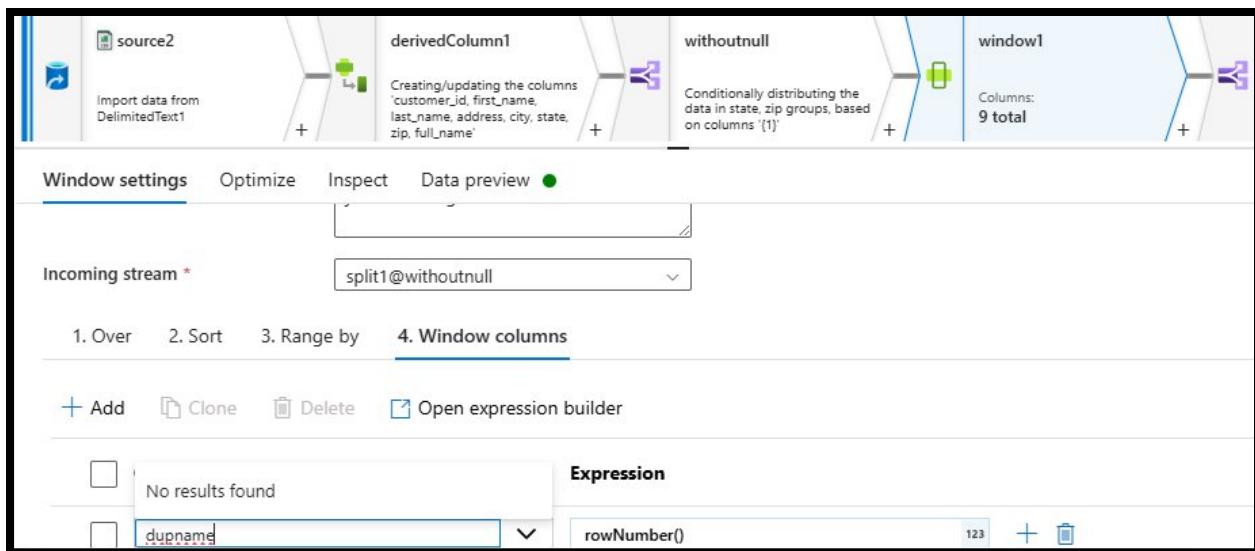


Removing null values.



1 row have null values.





The screenshot shows the configuration for a **Conditional split**. The **Output stream name** is set to **split3**. The **Incoming stream** is **window2**. The **Split on** condition is set to **First matching condition**. There are two split conditions defined:

- Stream names:** cleandataa, **Condition:** dupname==1
- Stream names:** duplicatevalues, **Condition:** Rows that do not meet any condition will use this output stream

Removing duplicate values.

Conditional split settings Optimize Inspect **Data preview**

Number of rows + **INSERT** 86 * **UPDATE** 0 x **DELETE** 0 + **UPsert** 0 q **LOOKUP** 0 x **ERROR** 0 TOTAL 86

Output stream cleandataa

	customer_id	12s ↑	first_name	abc ↑↓	last_name	abc ↑↓	address	abc ↑↓	city	abc ↑↓	state	abc ↑↓	zip	abc ↑↓	full_name	abc ↑↓	dupname	123 ↑↓
+	78		Abiqail		Cole		7777 Fir St		Sundridge		ON		P0A0A1		AbiqailCole		1	
+	52		Abiqail		Henderson		5151 Cypress Ave		Mount Albert		ON		L0G0A1		AbiqailHenderson		1	
+	26		Abiqail		Parker		2525 Poplar St		Barrie		ON		L4M0A1		AbiqailParker		1	
+	37		Alexander		Bell		3636 Redwood Dr		Stratford		ON		N5A0A1		AlexanderBell		1	
+	63		Alexander		Foster		6262 Spruce Ln		Stayner		ON		L0M0A1		AlexanderFoster		1	

Conditional split settings Optimize Inspect **Data preview**

Number of rows + **INSERT** N/A * **UPDATE** N/A x **DELETE** N/A + **UPsert** N/A q **LOOKUP** N/A x **ERROR** N/A

Output stream duplicatevalues

	customer_id	abc ↑↓	first_name	abc ↑↓	last_name	abc ↑↓	address	abc ↑↓	city	abc ↑↓	state	abc ↑↓	zip	abc ↑↓	full_name	abc ↑↓	dupname
No output data.																	

No duplicate values are there.

bootcamp project ● dataflow1 ● set_properties ●

Validate Data flow debug Debug Settings

Source: source2

Derived Column: derivedColumn3

Processor: withoutnull

Processor: window2

Processor: cleandataa

Processor: duplicatevalues

Sink: 9 Columns

Conditional split settings Optimize Inspect **Data preview**

Number of rows + **INSERT** N/A * **UPDATE** N/A x **DELETE** N/A + **UPsert** N/A q **LOOKUP** N/A x **ERROR** N/A TOTAL N/A

Output stream

	customer_id	12s ↑↓	first_name	abc ↑↓	last_name	abc ↑↓	address	abc ↑↓	city	abc ↑↓	state	abc ↑↓	zip	abc ↑↓	full_name	abc ↑↓	dupname	123 ↑↓
	78		Abiqail		Cole		7777 Fir St		Sundridge		ON		P0A0A1		AbiqailCole		1	
	52		Abiqail		Henderson		5151 Cypress Ave		Mount Albert		ON		L0G0A1		AbiqailHenderson		1	

3 source

3 SOURCE—LOAN_PAYMENTS.

The screenshot shows the 'dataflow1' blade in the Azure Data Factory interface. The 'Source settings' tab is selected. The 'Output stream name' is set to 'source3loanpayments'. The 'Description' field contains the text 'Import data from DelimitedText1'. The 'Source type' dropdown is set to 'Dataset'. The 'Dataset' dropdown is set to 'DelimitedText1'. The 'Options' section includes a checked checkbox for 'Allow schema drift'.

The screenshot shows the 'Inspect' tab in the Data Flow blade. It displays the schema for the 'source3loanpayments' stream. There are four columns: 'payment_id' (short), 'loan_id' (short), 'payment_date' (date), and 'payment_amount' (double).

Order	Column	Type
1	payment_id	short
2	loan_id	short
3	payment_date	date
4	payment_amount	double

The screenshot shows the 'Data preview' tab in the Data Flow blade. It displays the first six rows of the 'source3loanpayments' stream. The columns are 'payment_id', 'loan_id', 'payment_date', and 'payment_amount'.

payment_id	loan_id	payment_date	payment_amount
1	45	2024-01-01	100.0
2	23	2024-01-02	150.0
3	67	2024-01-03	200.0
4	89	2024-01-04	250.0
5	12	2024-01-05	300.0
6	34	2024-01-06	350.0

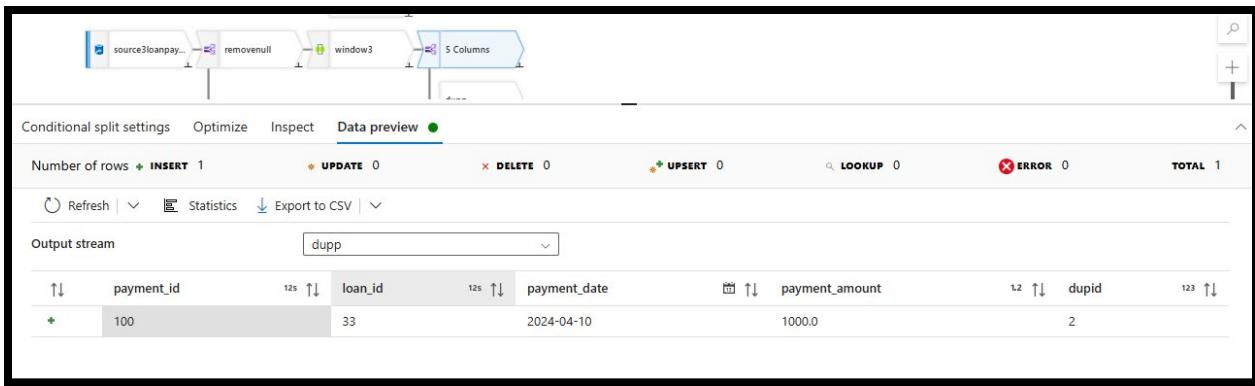
The screenshot shows the 'Conditional split settings' section of the Azure Data Flow interface. At the top, there are tabs for 'Conditional split settings', 'Optimize', 'Inspect', and 'Data preview'. The 'Conditional split settings' tab is selected. Below it, there is a 'Description' field containing the text: 'Conditionally distributing the data in loan_id, payment_date groups, based on columns '{1}''. Under 'Incoming stream *', the value is 'source3loanpayments'. Under 'Split on', the radio button is set to 'First matching condition'. In the 'Split condition' section, there are two entries: 'removennull' with the condition '!isNull(loan_id)&&!isNull(payment_date)' and 'withnull' with the condition 'Rows that do not meet any condition will use this output stream'.

The screenshot shows the 'Data preview' section of the Azure Data Flow interface. It displays a preview of the data from the 'removennull' output stream. The data consists of four columns: payment_id, loan_id, payment_date, and payment_amount. There are three rows shown:

	payment_id	loan_id	payment_date	payment_amount
+	1	45	2024-01-01	100.0
+	2	23	2024-01-02	150.0
+	3	67	2024-01-03	200.0

At the top of the preview area, there are counters for various operations: INSERT 100, UPDATE 0, DELETE 0, UPSERT 0, LOOKUP 0, and ERROR 0. Below the preview, there are buttons for Refresh, Statistics, Export to CSV, and a search bar.

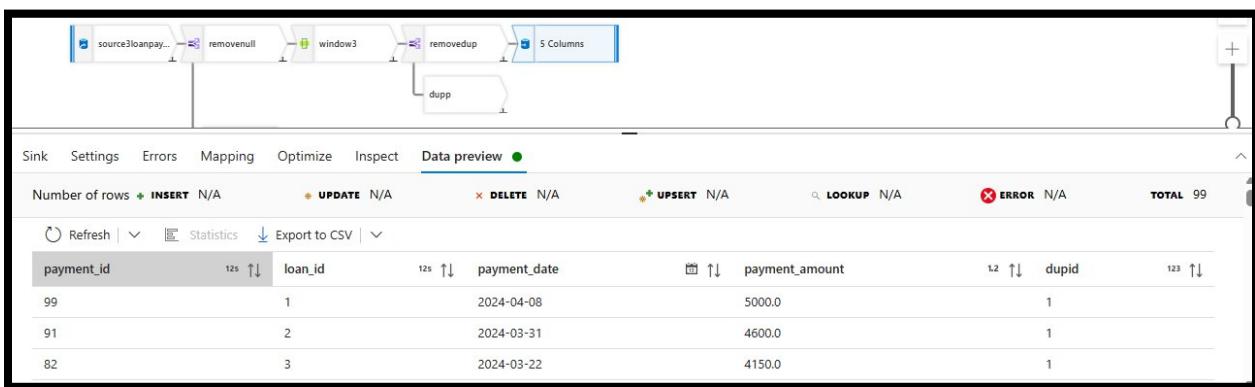
The screenshot shows the 'Expression' builder page. It has two main sections: 'Column name *' and 'Expression'. In the 'Column name *' section, the value 'dupid' is entered. In the 'Expression' section, the expression 'rowNumber()' is typed. Below the expression input, there is a toolbar with various operators: +, -, *, /, ||, &&, !, ^, ==, ===, <=>, !=, >.



DUPLICATE VALUE.

This screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' and 'Datasets'. The main area shows a pipeline named 'dataflow6' with several stages: 'source1' (Import data from DelimitedText1), 'removennull' (Conditionally distributing the data in loan_id, payment_date groups, based on columns [1]), 'window1' (Aggregates data based on a window and joins with original data), 'dedup' (Conditionally distributing the data in dupid groups, based on columns [1]), 'alterRow1' (Add expressions to alter rows), and finally 'sink1' (Sink dataset). The 'sink1' stage is highlighted. The 'Sink' tab is selected, showing details like 'Output stream name' (sink1), 'Description' (Add sink dataset), 'Incoming stream' (alterRow1), 'Sink type' (Dataset, selected), 'Inline dataset type' (Delta), and 'Linked service' (ls_adls).

SINK TO SQLDB.



4 SOURCEàLOANS.

Source settings

Output stream name *: source4loans

Description: Import data from DelimitedText1

Source type *: Dataset

Dataset *: DelimitedText1

Options: Allow schema drift

loan_id	customer_id	loan_amount	interest_rate	loan_term
1	45	10000.5	5.5	36
2	12	20000.75	4.5	48
3	78	15000.0	6.0	60
4	34	30000.25	3.5	24
5	56	25000.0	5.0	36

Conditional split settings

Output stream name *: split6

Description: Conditionally distributing the data in loan_id, customer_id groups, based on columns '{1}'

Incoming stream *: source4loans

Split on: First matching condition

Split condition: Stream names: removingnull, Condition: !isNull(loan_id)&&!isNull(customer_id)

Conditional split settings Optimize Inspect **Data preview**

Number of rows: **INSERT 100** **UPDATE 0** **DELETE 0** **UPSERT 0** **LOOKUP 0** **ERROR 0** **TOTAL 100**

Output stream: **removingnull**

↑↓	loan_id	↑↓	customer_id	↑↓	loan_amount	↑↓	interest_rate	↑↓	loan_term	↑↓
+	1		45		10000.5		5.5		36	
+	2		12		20000.75		4.5		48	
+	3		78		15000.0		6.0		60	
+	4		34		30000.25		3.5		24	
+	5		55		25000.0		5.0		36	

Window settings Optimize Inspect **Data preview**

Description: Aggregates data based on a window and joins with original data

Incoming stream: **split6@removingnull**

4. Window columns

Add Clone Delete Open expression builder

Column Expression

dupid **rowNumber()**

Conditional split settings Optimize Inspect **Data preview**

Description: Conditionally distributing the data in dupid groups, based on columns '{1}'

Incoming stream: **window4**

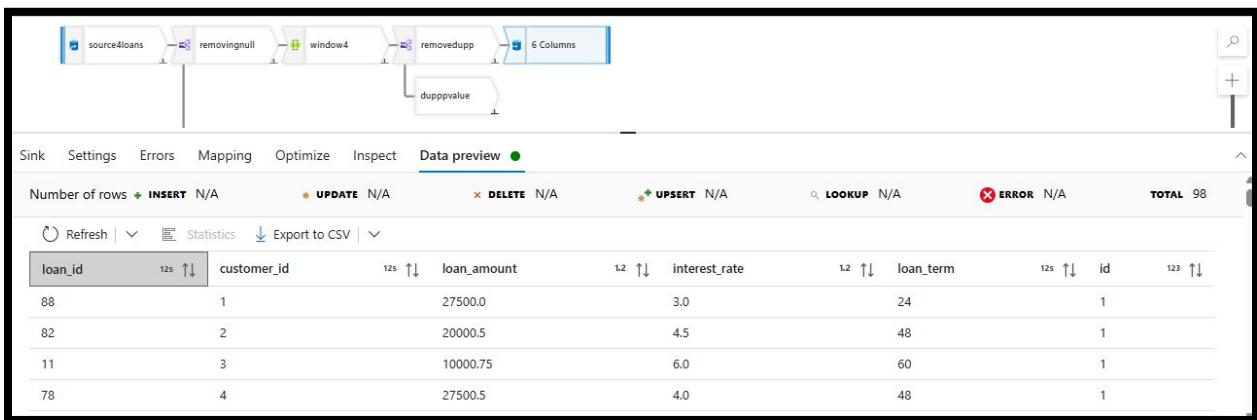
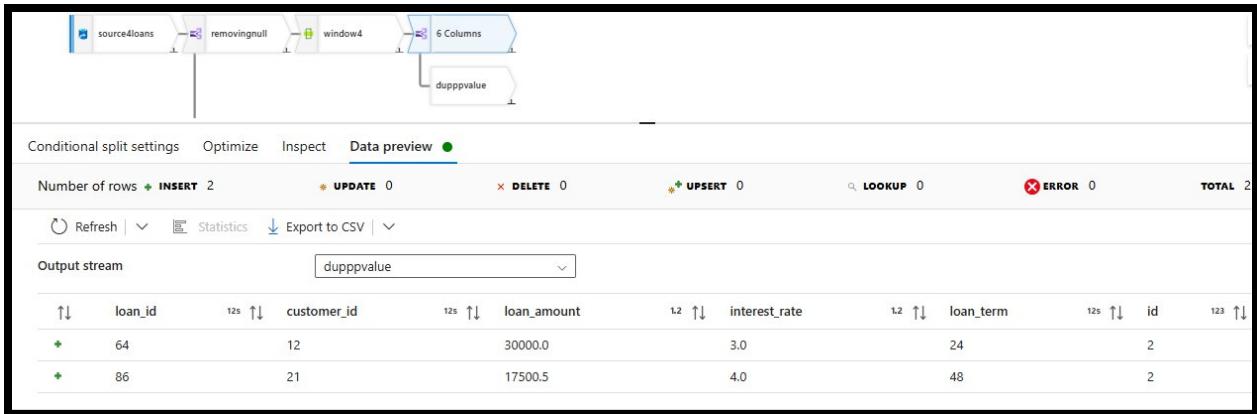
Split on: First matching condition All matching conditions

Split condition:

Stream names	Condition
removedupp	dupid==1
dupppvalue	

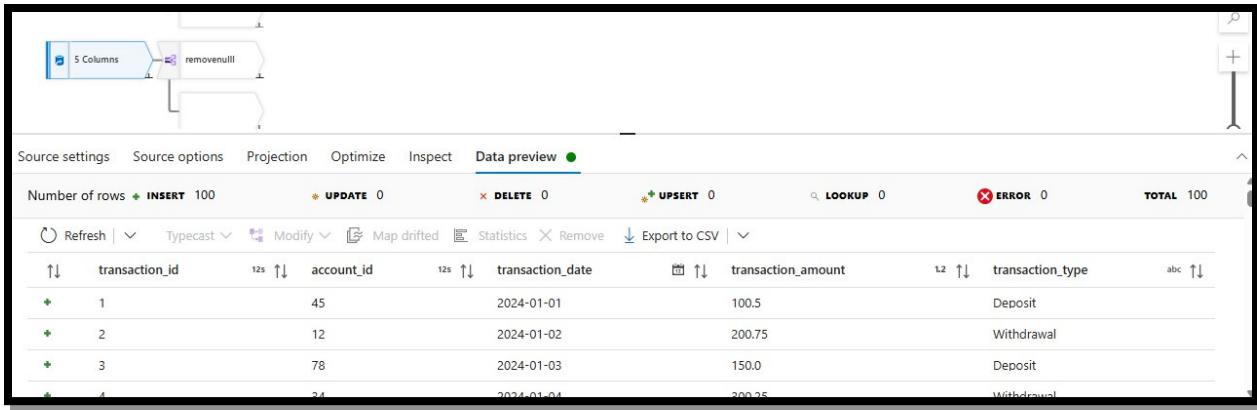
Rows that do not meet any condition will use this output stream

NANDINI RATHORE

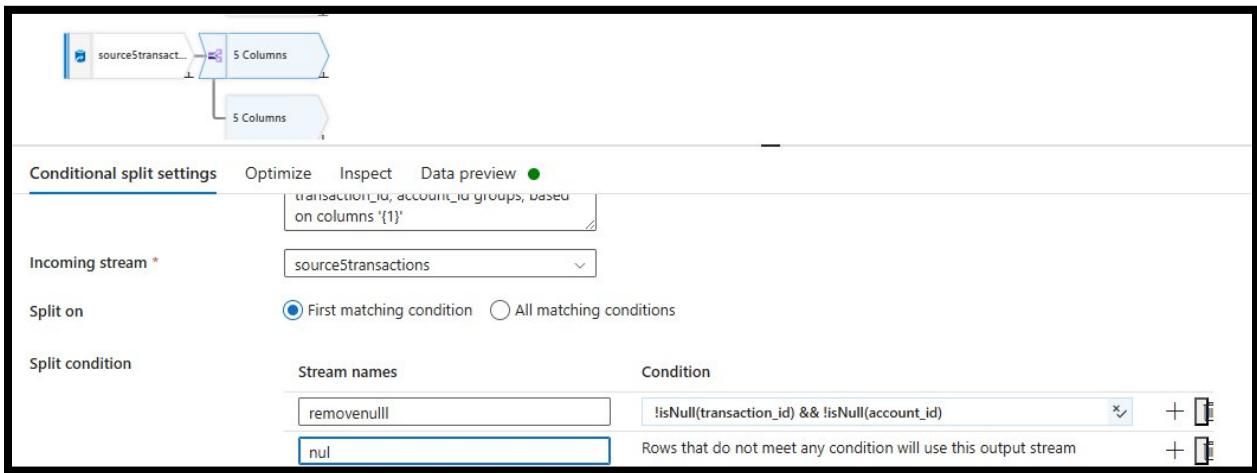


5. SOURCE à TRANSACTIONS

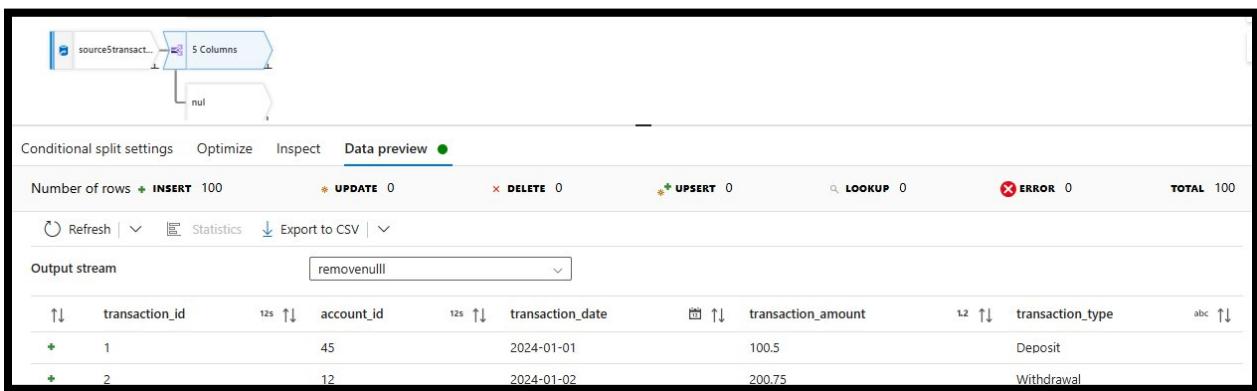
The screenshot shows the 'Source settings' page for a new source. The 'Source settings' tab is selected. The 'Output stream name' is set to 'source5transactions'. The 'Description' field contains the text 'Import data from DelimitedText1'. The 'Source type' section shows two options: 'Dataset' (selected) and 'Inline'. There are also 'Source options', 'Projection', 'Optimize', 'Inspect', and 'Data preview' tabs at the top.



DATA PREVIEW.



REMOVING NULL VALUES.



Window settings Optimize Inspect Data preview ●

Output stream name * window5

Description Aggregates data based on a window and joins with original data

Incoming stream * split8@removennull

1. Over 2. Sort 3. Range by 4. Window columns

split8@removennull's column Name as

12s account_id account_id + -

Window settings Optimize Inspect Data preview ●

Incoming stream * split8@removennull

1. Over 2. Sort 3. Range by 4. Window columns

+ Add Clone Delete Open expression builder

<input type="checkbox"/> Column	Expression
<input type="checkbox"/> dupid	rowNumber()

CREATING WINDOW COLUMN. TO REMOVE DUPLICATE VALUES.

sourceTransact... → removennull → 6 Columns → dupid

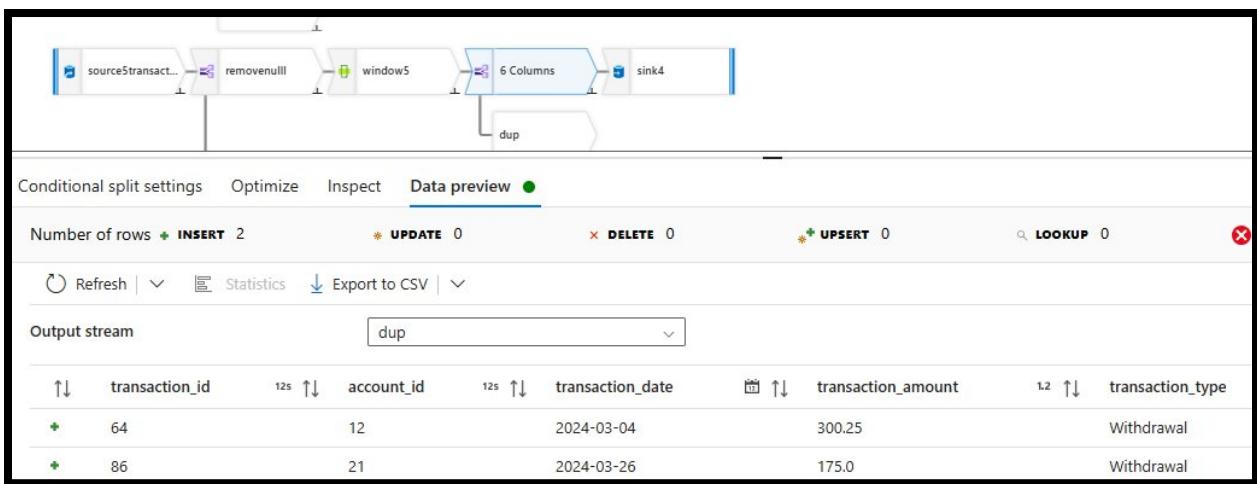
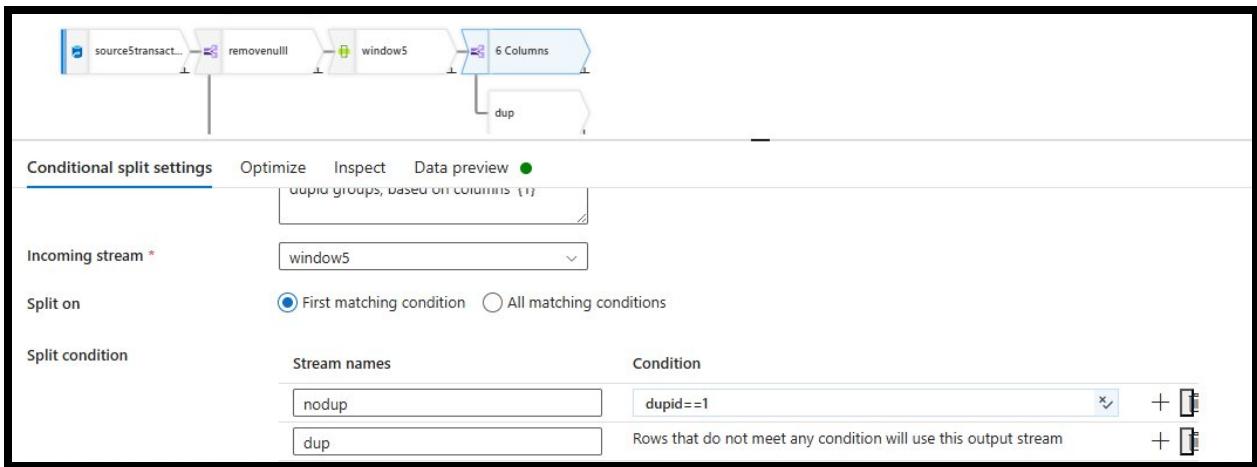
Window settings Optimize Inspect Data preview ●

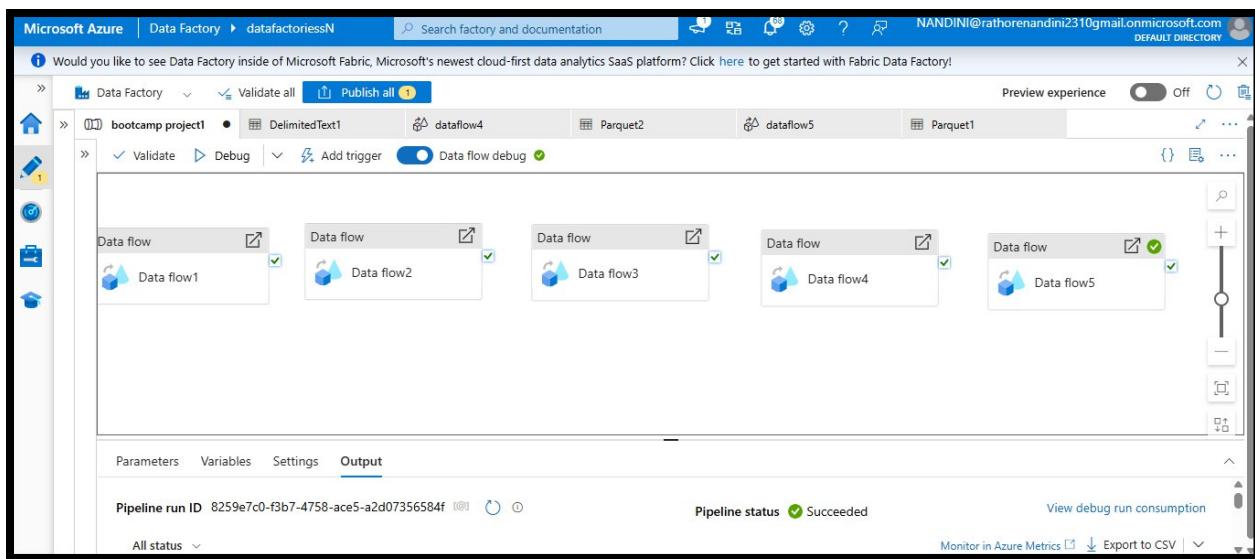
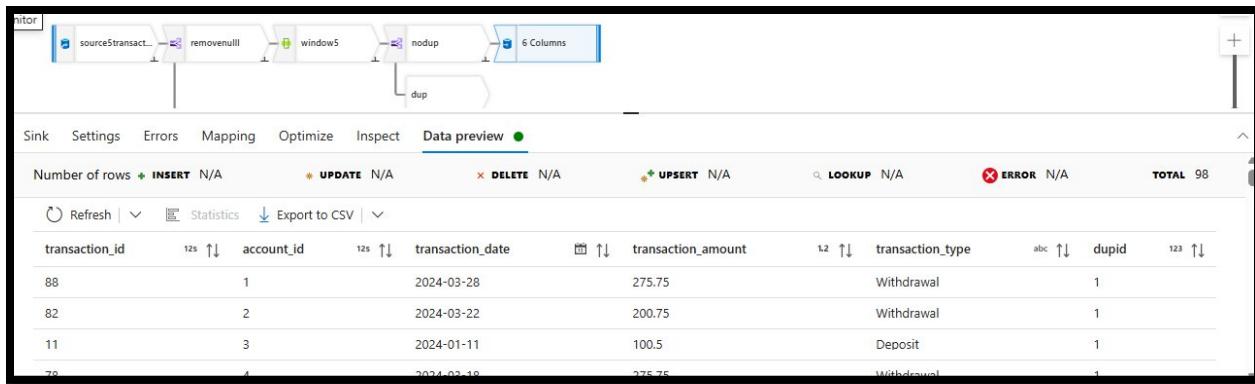
Number of rows INSERT 100 UPDATE 0 DELETE 0 UPSERT 0 LOOKUP 0 ERROR 0 TOTAL 1

Refresh | Typecast | Modify | Map drifted | Statistics | Remove | Export to CSV |

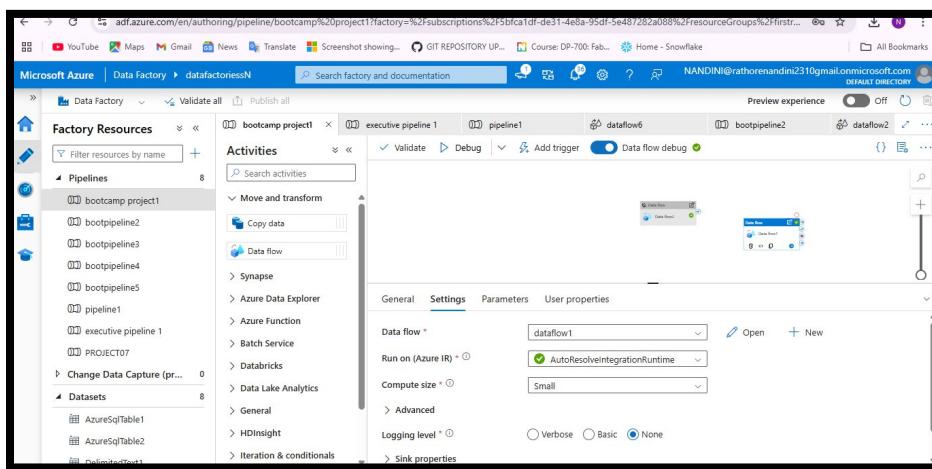
transaction_id	account_id	transaction_date	transaction_amount	transaction_type	dupid
1	45	2024-01-01	100.5	Deposit	1
2	12	2024-01-02	200.75	Withdrawal	1
3	78	2024-01-03	150.0	Deposit	1
4	34	2024-01-04	300.25	Withdrawal	1

NANDINI RATHORE

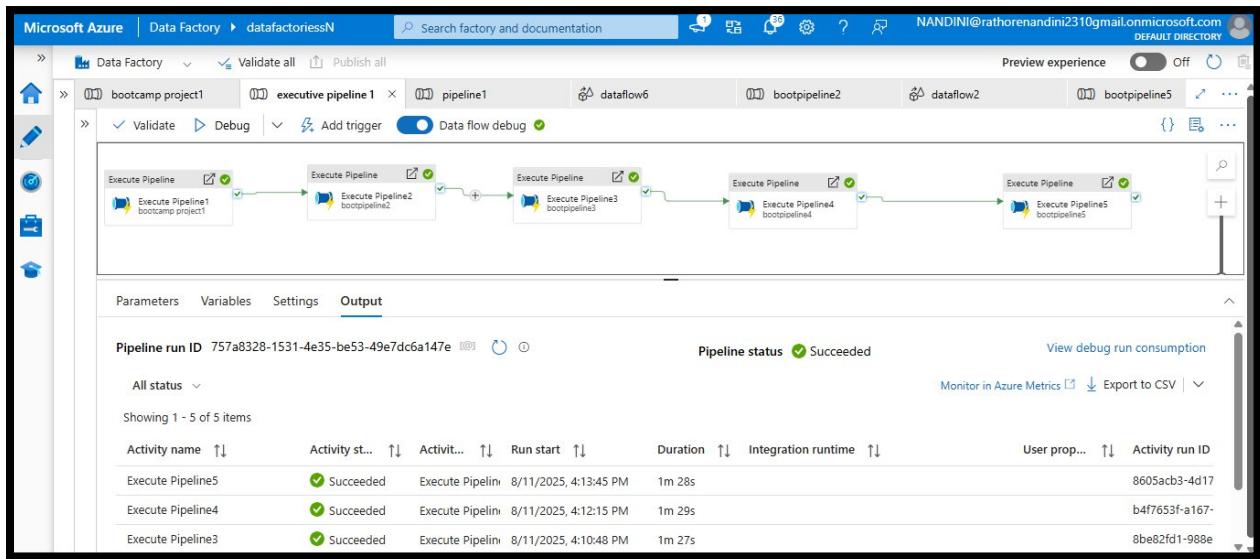




First I used this method for dataflow of 5 different sources.



I used 5 different pipelines having different dataflows and run them in a master pipeline to reduce the execution time and connect each pipeline with other and set behavior true . to trigger pipeline means other pipeline executes when previously executed successfully.



Target table creating in sql db.

```
1  create table logging_table
2  (id int identity(1,1),
3  tablename varchar(100),
4  schemaname varchar(100),
5  error_message varchar(500),
6  pipeline_status varchar(20),
7  pipeline_name varchar(100),
8  pipeline_runid varchar(100),
9  execution_details varchar(100))
10 drop table logging_table
11
12
13
```

Logging table.

```
1  create procedure log_proc
2  @tablename varchar(100),
3  @schemaname varchar(100),
4  @error_message varchar(1000),
5  @pipeline_status varchar(20),
6  @pipeline_name varchar(100),
7  @pipeline_runid varchar(100),
8  @execution_details varchar(100)
9  AS
10 BEGIN
11 | SET NOCOUNT ON
12 insert into logging_table VALUES
13 (@tablename,
14 @schemaname,
15 @error_message,
16 @pipeline_status,
17 @pipeline_name,
18 @pipeline_runid,
19 @execution_details
20 )
21 END
22
23 select * from logging_table
24
25 drop procedure log_proc
```

Results	Messages

Creating stored procedure.

Scdtype1 :-ACCOUNTS

The screenshot shows the Microsoft Azure Data Factory interface. In the top navigation bar, 'Data Factory' is selected. Below it, the 'Activities' section is open, showing 'Move and transform' options like 'Copy data' and 'Data flow'. A specific 'Data flow' activity named 'Data flow scd typ1 on accounts' is selected. The main panel displays the 'Settings' tab for this data flow, which is named 'dataflow6'. It is set to run on 'AutoResolveIntegrationRuntime' with a 'Small' compute size. The 'Logging level' is set to 'Verbose'. The 'Preview experience' toggle is off.

This screenshot shows the 'Data preview' tab of the Data Flow preview pane. It displays a complex data pipeline consisting of several stages: 'source1' (a dataset from 'ls_adls'), 'select1' (renaming columns), 'derivedColumn1' (creating/updating columns), 'join1' (left outer join on 'derivedColumn1' and 'source2'), 'insert' (conditionally distributing data), 'derivedColumn2' (creating/updating columns), and finally an 'sink1' stage (represented by a blob icon). Below the preview, the 'Source settings' tab is active, showing 'Output stream name' as 'source1', 'Description' as 'Import data from ls_adls', 'Source type' as 'Dataset', 'Inline dataset type' as 'Delta', and 'Linked service' as 'ls_adls'.

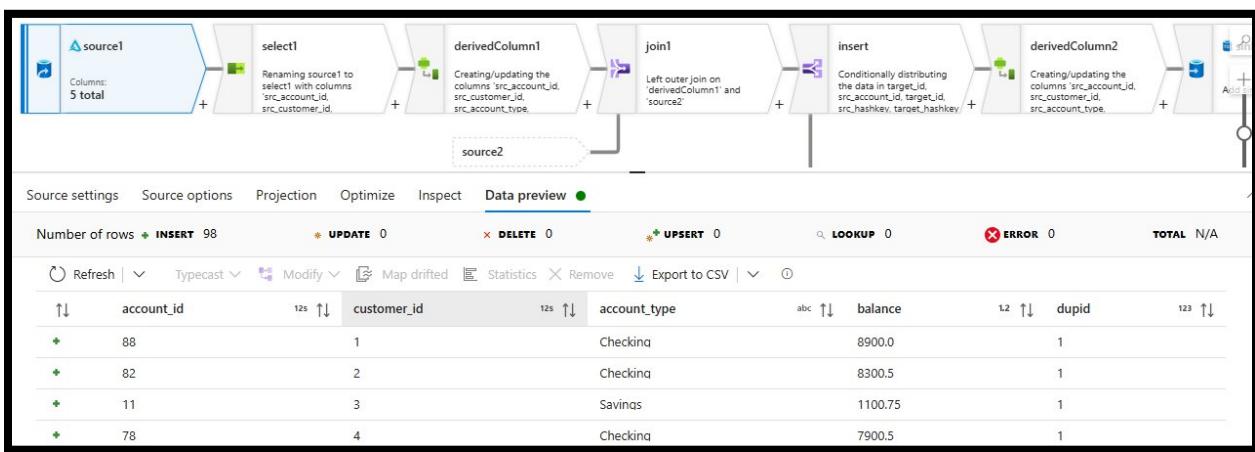
Source:> ACCOUNTS.

This screenshot shows the 'Source settings' tab for the 'ACCOUNTS' source. The 'Folder path' is set to 'silver / delta/accounts'. Other settings include 'Allow no files found' (unchecked), 'Compression type' as 'No compression', and 'Time travel' set to 'Disable'. The preview pane shows the same complex data pipeline as the previous screenshot, starting from 'source1' and ending at 'sink1'.

Browsing the accounts file in delta format in silver container in adlsgen2.



Import schema.



Target table creating in sqldb.

Output stream name *: source2

Description: Import data from AzureSqlDatabase1

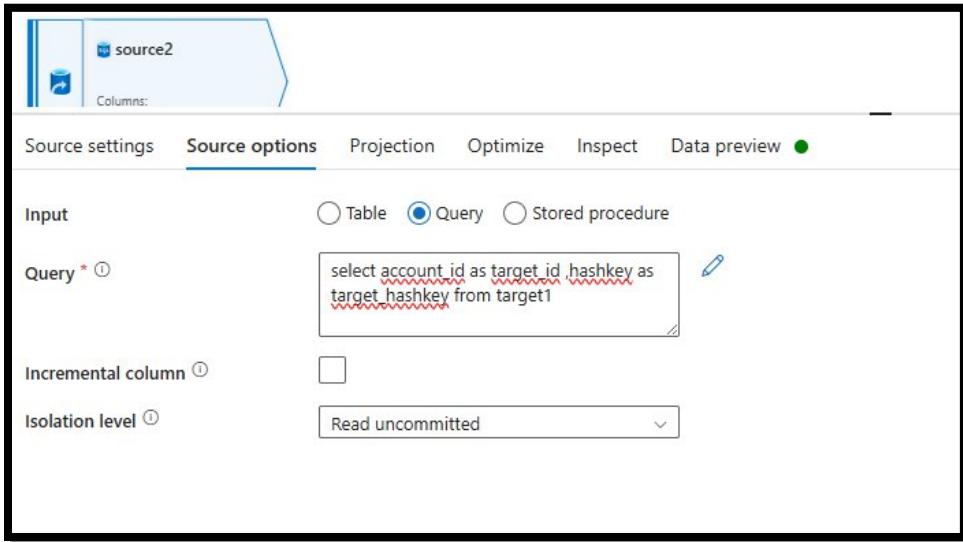
Source type *: Dataset

Inline dataset type *: Azure SQL Database

Linked service *: AzureSqlDatabase1

Sampling *: Enable Disable

Test connection: Connection successful



Using query

"select account_id as target_id,hashkey as target_hashkey from target1"

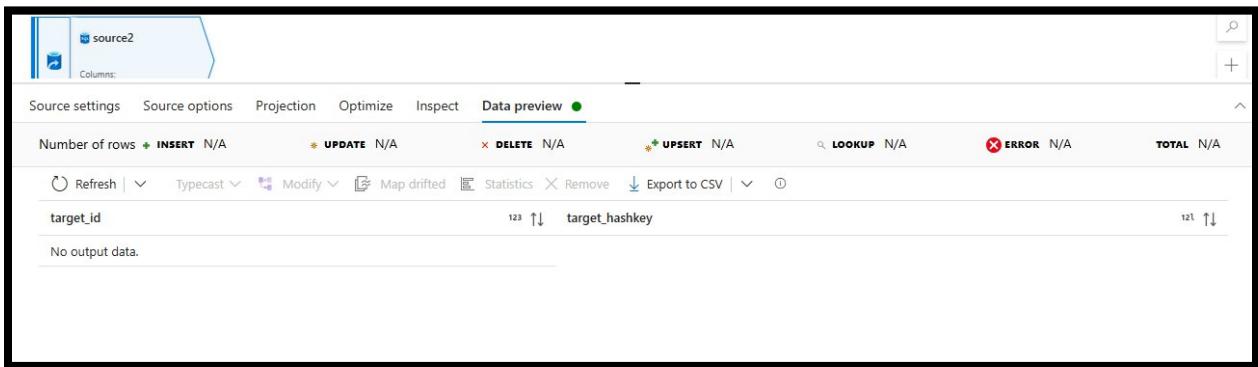
SELECT-->Specifies which columns you want to retrieve from the table.

Account_id AS target_id-->The column id will be renamed (aliased) as target_id in the output.

hashkey AS target_hashkey-->The column hashkey will be renamed as target_hashkey in the output.

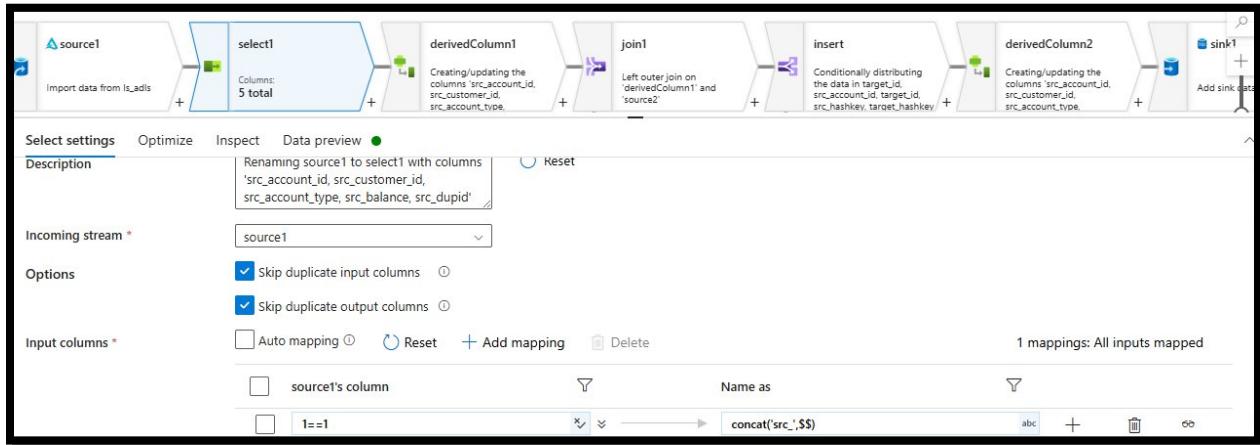
FROM target1-->The data is being fetched from the table named target1 .

THEN IMPORT SCHEMA,OPTIMIZE-USE CURRENT PARTITIONING,INSPECT AND DATA PREVIEW.



To clearly differentiate source vs target fields — especially useful in ETL pipelines, SCD Data Flows, or comparison queries.

Makes output easier to understand — you know these are "target" values.



USE OF 1==1:> Always True Condition.

Used in joins/lookups to match all rows when no specific key is needed.

Helps create Cross Joins or dummy conditions.

Example:

1==1 ensures every row matches every other row. Means every row of source matches to target row.

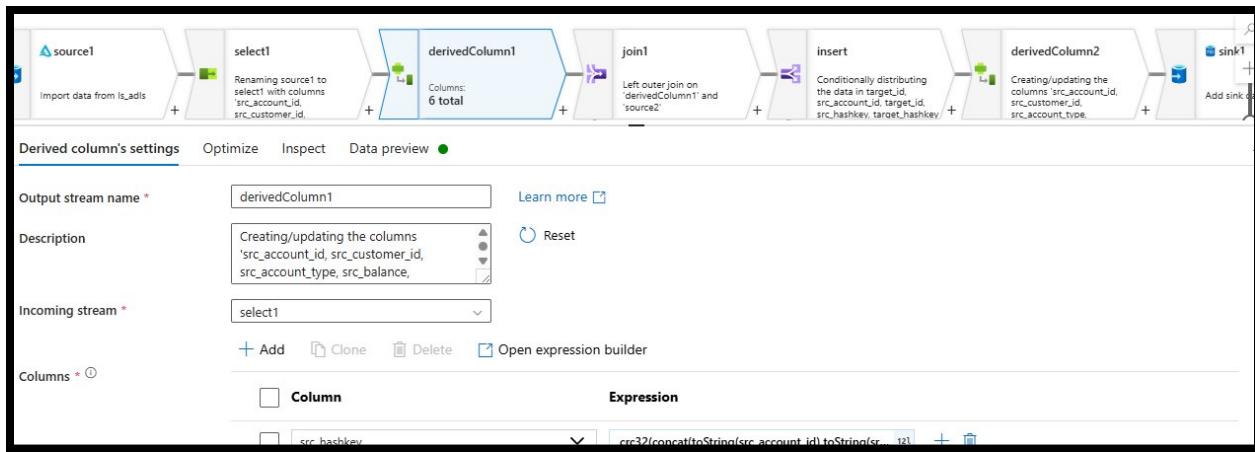
Use of concat('src_',\$\$):> To dynamically build strings in expressions.

\$\$ is a parameter placeholder — its value comes at runtime. Useful for making column names or strings dynamic. Example:

If \$\$ = 'id' then: concat('src_', \$\$) → 'src_id'.

src_account_id	src_customer_id	src_account_type	src_balance	src_dupid
88	1	Checking	8900.0	1
82	2	Checking	8300.5	1
11	3	Savings	1100.75	1
78	4	Checking	7900.5	1
18	5	Checking	1600.5	1

Generating hashkey:-



Hashkey is the unique number that represents the combines value of columns.

It detects if any value in those number is change if changes, data changed.

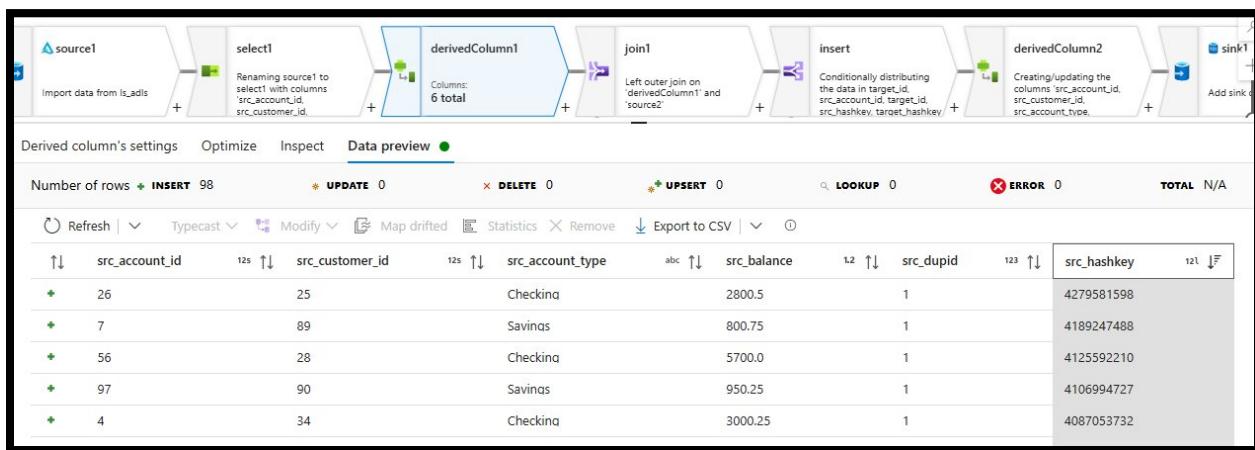
Concat is used to join all the columns, toString is used to convert numbers into string.

Crc32 hash function which helps to create a unique numeric value (hashkey).

The screenshot shows the 'Dataflow expression builder' interface. A new derived column 'src_hashkey' is being created with the following expression:

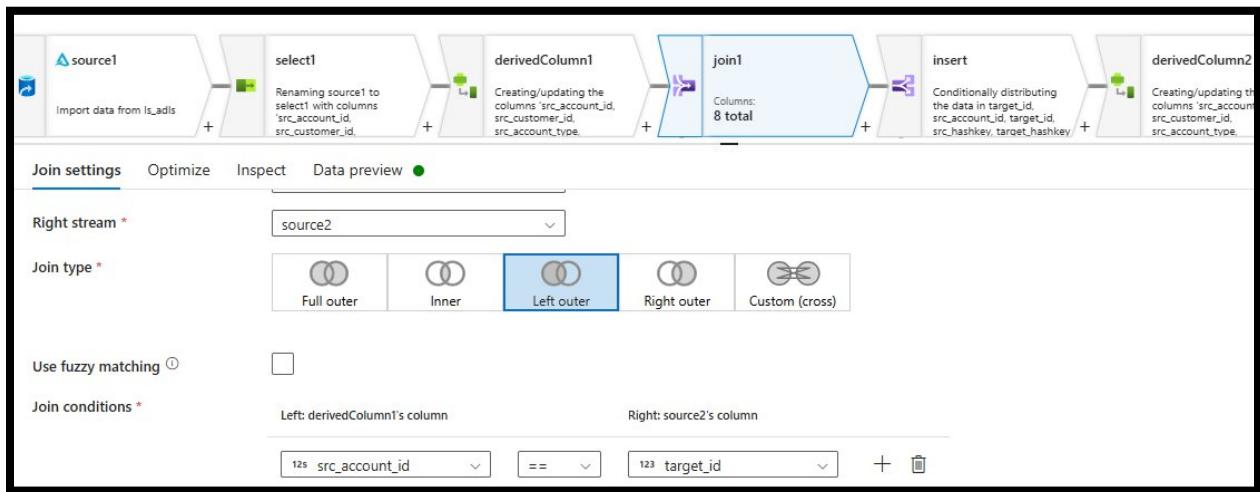
```
crc32(concat(toString(src_account_id),toString(src_customer_id),src_account_type,toString(src_balance)))
```

Use current partitioning to optimize value.



Data preview.

JOIN:- left outer join of delta files with target table.



Src_customer_id==target_id mean source equals to target on the bases of id.

The data preview table shows the following columns and data:

	src_account_id	src_customer_id	src_account_type	src_balance	src_dupid	src_hashkey	target_id	target_hashkey
*	88	1	Checking	8900.0	1	3995045802	NULL	NULL
*	82	2	Checking	8300.5	1	689386274	NULL	NULL
*	11	3	Savings	1100.75	1	877522712	NULL	NULL
*	78	4	Checking	7900.5	1	3909513848	NULL	NULL
*	18	5	Checking	1600.5	1	3683965059	NULL	NULL

The Conditional split settings for the **join1** stream are as follows:

- Description**: Conditionally distributing the data in target_id, src_account_id, target_id, src_hashkey, target_hashkey groups.
- Incoming stream**: join1
- Split on**: First matching condition (radio button selected).
- Split condition**:

Stream names	Condition
insert	isNull(target_id)
update	src_account_id==target_id&&src_hashkey==target_hashkey

	src_account_id	src_customer_id	src_account_type	src_balance	src_dupid	src_hashkey	target_id	target_hashkey
*	88	1	Checking	8900.0	1	3995045802	NULL	NULL
*	82	2	Checking	8300.5	1	689386274	NULL	NULL
*	11	3	Savings	1100.75	1	877522712	NULL	NULL
*	70	4	Checking	7000.5	1	3090512848	NULL	NULL

Insert:> `isNull(target_id)`

Update:> `src_account_id==target_id&&src_hashkey==target_hashkey`

For the first run: Insert Condition is TRUE — because `target_id` is null (record does not exist in target).

Update Condition is FALSE — because nothing matches (no old record yet).

First time run = Only Insert works (because target is empty).

- ✓ Update is empty because no matching records yet.
- ✓ Next runs = Update will work when changes are detected.

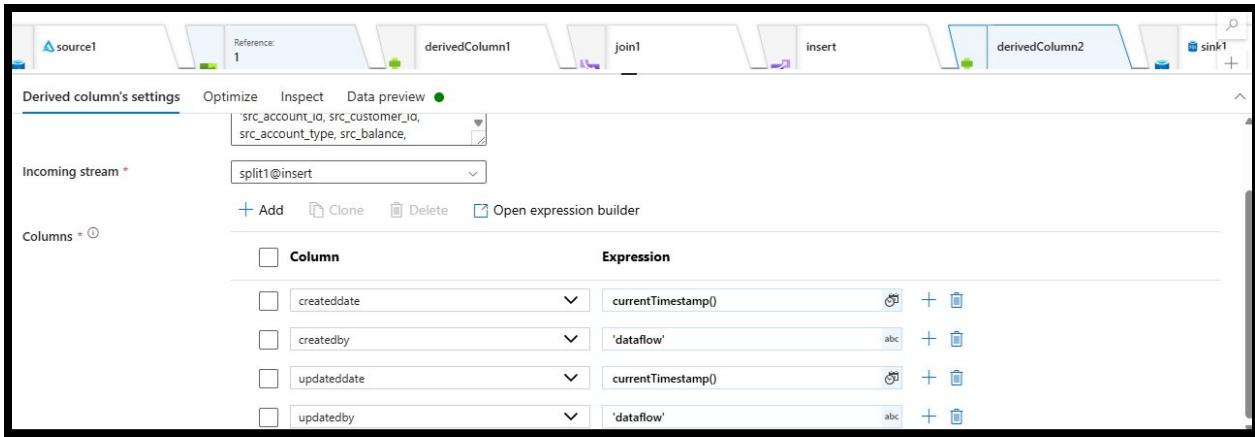
`isNull(target_id)` checks if the target record does not exist (means no matching id found in the target table).

used to identify new records that should be inserted into the target.

`src_account_id == target_id && src_hashkey == target_hashkey` checks if the record exists in the target and the data

has not changed.

this condition usually means "**No Update Needed**" — because both ID and data (hashkey) are the same.



Adding new columns.

The screenshot shows the 'Data preview' section of the Azure Data Factory interface. It displays two sets of data rows. The first set (src_account) has 98 rows and includes columns: src_account_id, src_customer_id, src_account_type, src_balance. The second set (src_dupid) has 10 rows and includes columns: src_dupid, src_hashkey, target_id, target_hashkey, createddate, createdby. The data is as follows:

src_account	src_dupid	src_hashkey	target_id	target_hashkey	createddate	createdby
2	1	1425950702	NULL	NULL	2025-08-11 21:03:12.806	dataflow
4	1	4087053732	NULL	NULL	2025-08-11 21:03:12.806	dataflow
6	1	3231065676	NULL	NULL	2025-08-11 21:03:12.806	dataflow
8	1	840806741	NULL	NULL	2025-08-11 21:03:12.806	dataflow
10	1	732231488	NULL	NULL	2025-08-11 21:03:12.806	dataflow

Sink:

The screenshot shows the 'Sink' configuration section of the Azure Data Factory interface. The configuration includes:

- Output stream name: sink1
- Description: Add sink dataset
- Incoming stream: derivedColumn2
- Sink type: Dataset
- Inline dataset type: Azure SQL Database
- Linked service: AzureSqlDatabase1

NANDINI RATHORE

The screenshot shows the 'Settings' tab for a sink configuration. The schema name is set to 'dbo' and the table name is 'accounts'. The 'Table action' is set to 'None'. Under 'Update method', 'Allow insert' is checked, while 'Allow delete', 'Allow upsert', and 'Allow update' are unchecked. The 'Use tempdb' checkbox is also checked.

The screenshot shows the 'Mapping' tab where columns are mapped from the source to the output. The source columns are: src_account_id, src_customer_id, src_account_type, src_balance, src_hashkey, createddate, createdby, updateddate, and updatedby. The corresponding output columns are: account_id, customer_id, account_type, balance, hashkey, createddate, createdby, updateddate, and updatedby. Each mapping is represented by a double-headed arrow between the source and output fields.

The screenshot shows the 'Data preview' tab with a table of sample data. The columns are account_id, customer_id, account_type, balance, hashkey, createddate, createdby, updateddate, and updatedby. The data consists of six rows:

	account_id	customer_id	account_type	balance	hashkey	createddate	createdby	updateddate	updatedby
1	45	Savings	1000.5	2331568292	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow
2	12	Checking	2500.75	1425950702	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow
3	78	Savings	1500.0	2797132651	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow
4	34	Checking	3000.25	4087053732	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow
5	56	Savings	500.0	3722820882	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow
6	23	Checking	1200.5	3231065676	2025-08-11 21:08:46.652	dataflow		2025-08-11 21:08:46.652	dataflow

```

1 create table target1(account_id int, customer_id int, account_type varchar(50), balance int, hashkey bigint, createdby varchar(100), createddate datetime, updatedby varchar(100), updateddate datetime)
2
3 select * from dbo.target1

```

	account_id	customer_id	account_type	balance	hashkey	createdby	createddate	updatedby	updateddate
1	1	45	Savings	1000	2331568292	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
2	2	12	Checking	2500	1425950702	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
3	3	78	Savings	1500	2797132651	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
4	4	34	Checking	3000	4087053732	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
5	5	56	Savings	500	3722820882	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
6	6	23	Checking	1200	3231065676	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
7	7	89	Savings	800	4189247488	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
8	8	67	Checking	2200	840806741	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
9	9	14	Savings	900	962565403	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
10	10	92	Checking	1800	732231488	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
11	11	3	Savings	1100	877522712	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
12	12	81	Checking	2700	1037549439	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
13	13	29	Savings	1300	1490255253	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
14	14	64	Checking	3200	2199603256	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
15	15	47	Savings	700	2748620714	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
16	16	18	Checking	1400	1036945993	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
17	17	99	Savings	600	862821406	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
18	18	5	Checking	1600	3683965059	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
19	19	76	Savings	400	2534900348	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130
20	20	21	Checking	2000	2070077009	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130

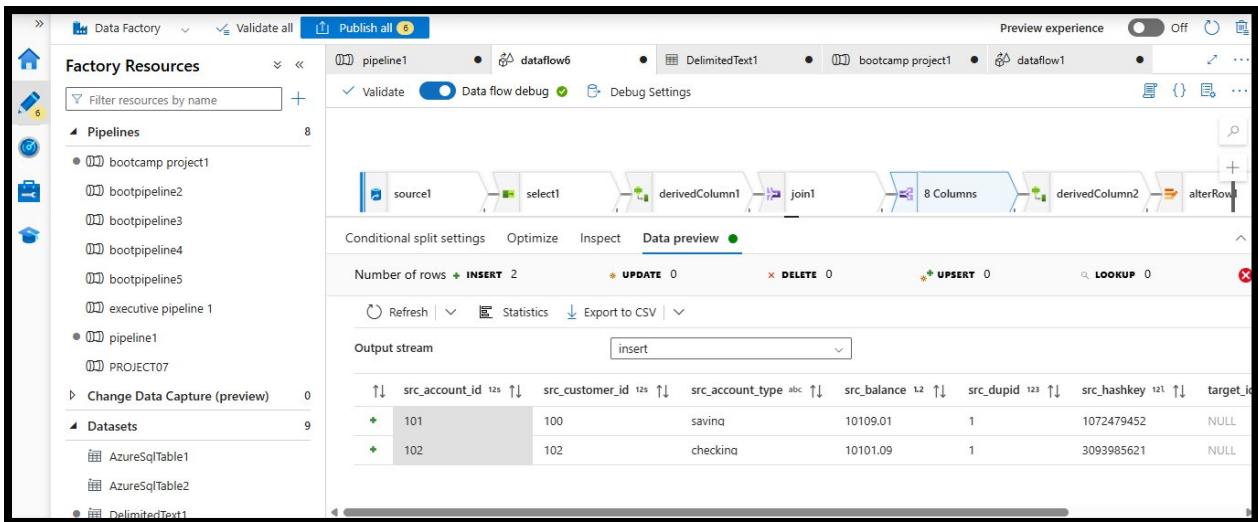
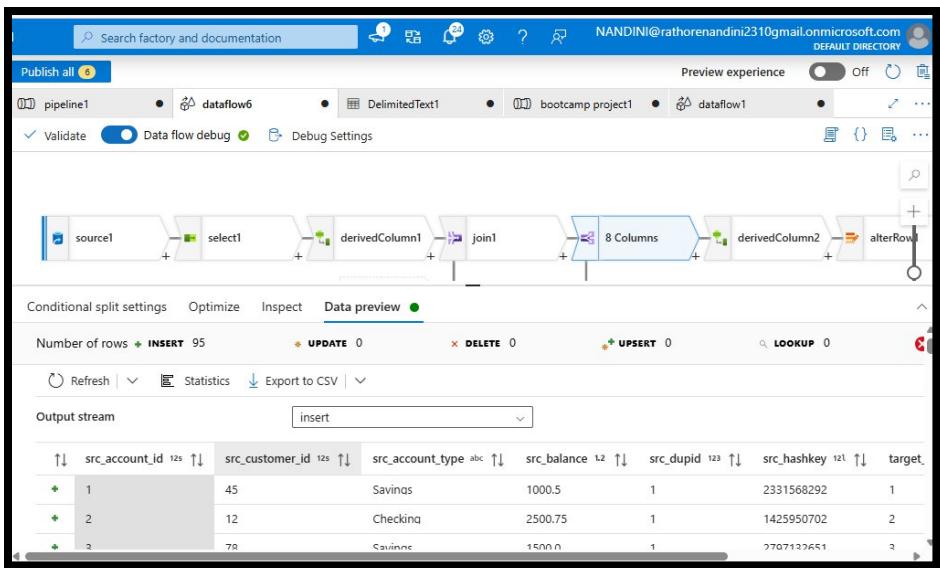
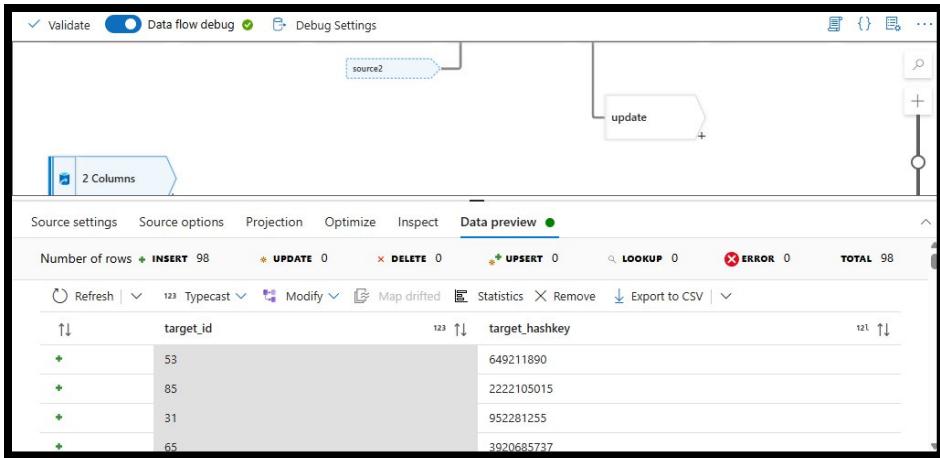
```
6 select account_id as target_id,hashkey as target_hashkey from target1
```

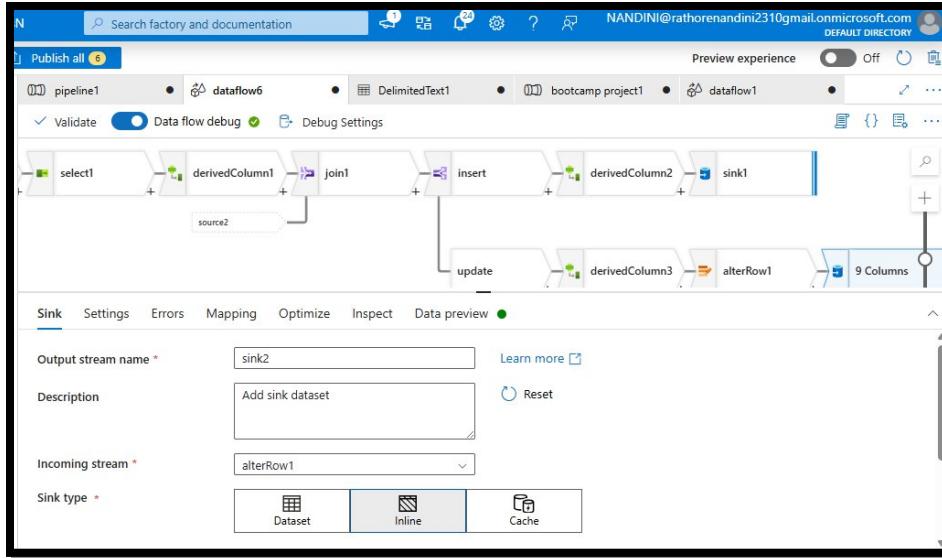
	target_id	target_hashkey
1	1	2331568292
2	2	1425950702
3	3	2797132651
4	4	4087053732
5	5	3722820882
6	6	3231065676
7	7	4189247488
8	8	840806741
9	9	962565403
10	10	732231488
11	11	877522712
12	12	1037549439
13	13	1490255253
14	14	2199603256
15	15	2748620714
16	16	1036945993
17	17	862821406

Logging table:-

Update table:>

NANDINI RATHORE



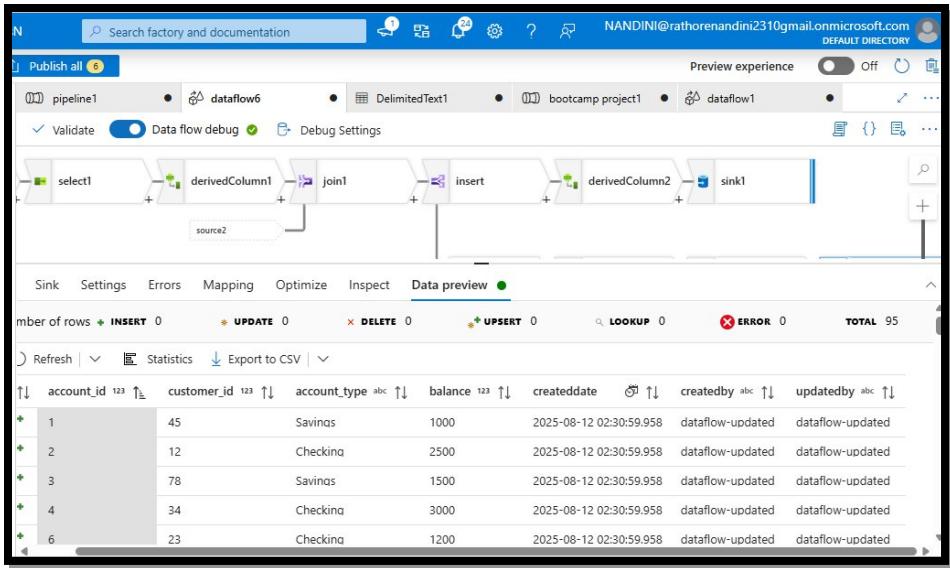


Data preview->

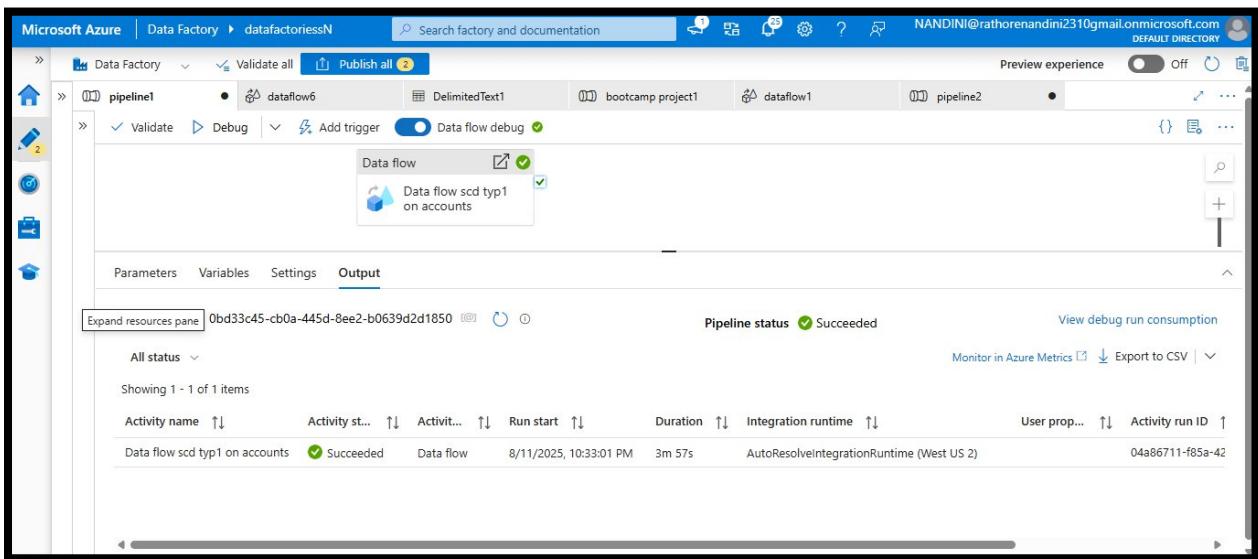
Input columns	Output columns
target_id	account_id
src_customer_id	customer_id
src_account_type	account_type
src_balance	balance
target_hashkey	src_hashkey
createddate	createddate
createdby	createdby
updatedate	updatedate
updatedby	updatedby

Mapping of update columns.

NANDINI RATHORE



Now run the pipeline and check table in sqldb.



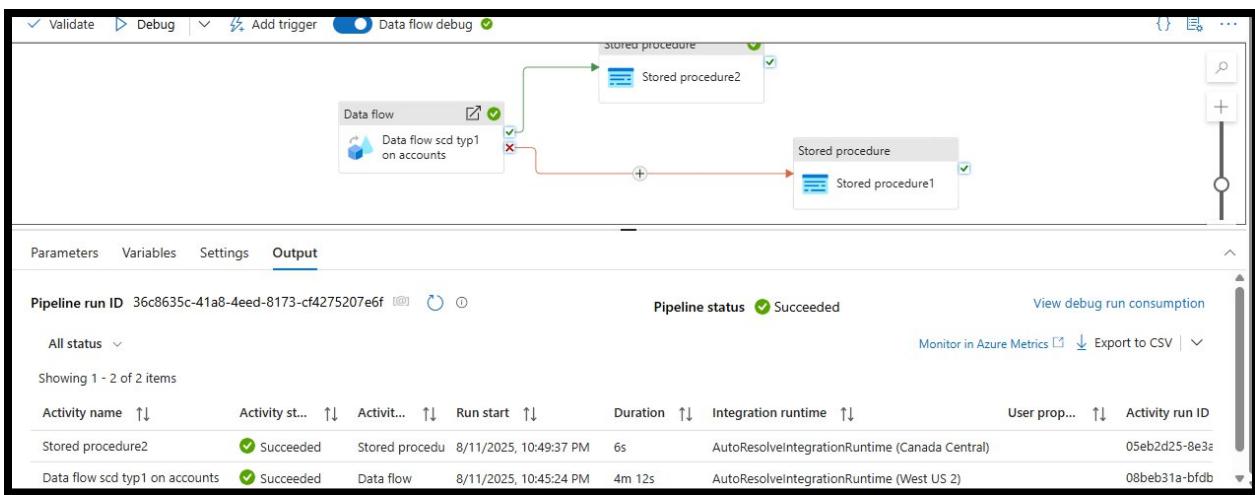
NANDINI RATHORE

```

1 create table target1(account_id int, customer_id int, account_type varchar(50), balance int, hashkey bigint, createdby varchar(100), createddate datetime, updatedby varchar(100), updateddate datetime)
2
3 select * from dbo.target1
4

```

	account_id	customer_id	account_type	balance	hashkey	createdby	createddate	updatedby	updateddate		
1	5	56	Savings	500	3722820882	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130		
2	9	14	Savings	900	962565403	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130		
3	99	88	Savings	975	2556707248	dataflow	2025-08-11 22:42:32.130	dataflow	2025-08-11 22:42:32.130		
	4	101	100	saving	10109	1072479452	dataflow	2025-08-12 02:34:21.767	dataflow	2025-08-12 02:34:21.767	
		5	102	checkning	10101	3093985621	dataflow	2025-08-12 02:34:21.767	dataflow	2025-08-12 02:34:21.767	
		6	53	Savings	400	649211890	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130	
		7	85	65	Savings	800	2222105015	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		8	31	71	Savings	125	952281255	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		9	65	69	Savings	550	3920685737	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		10	78	4	Checkning	7900	3909513848	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		11	81	70	Savings	750	3423619128	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		12	76	22	Checkning	7700	3799983974	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130
		13	12	81	Checkning	2700	1037549439	dataflow-updated	2025-08-12 02:35:17.173	dataflow-updated	2025-08-11 22:42:32.130



```

1 create table logging_table
2 (id int identity(1,1),
3 tablename varchar(100),
4 schemaname varchar(100),
5 error_message varchar(500),
6 pipeline_status varchar(20),
7 pipeline_name varchar(100),
8 pipeline_runid varchar(100),
9 execution_details varchar(100))
10 select * from logging_table
11
12
13

```

	id	tablename	schemaname	error_message	pipeline_status	pipeline_name	pipeline_runid	execution_details
1	1	target1	dbo	NULL	success	pipeline1	36c8635c-41a8-4eed-8173-cf4275207e6f	2025-08-12T02:45:15

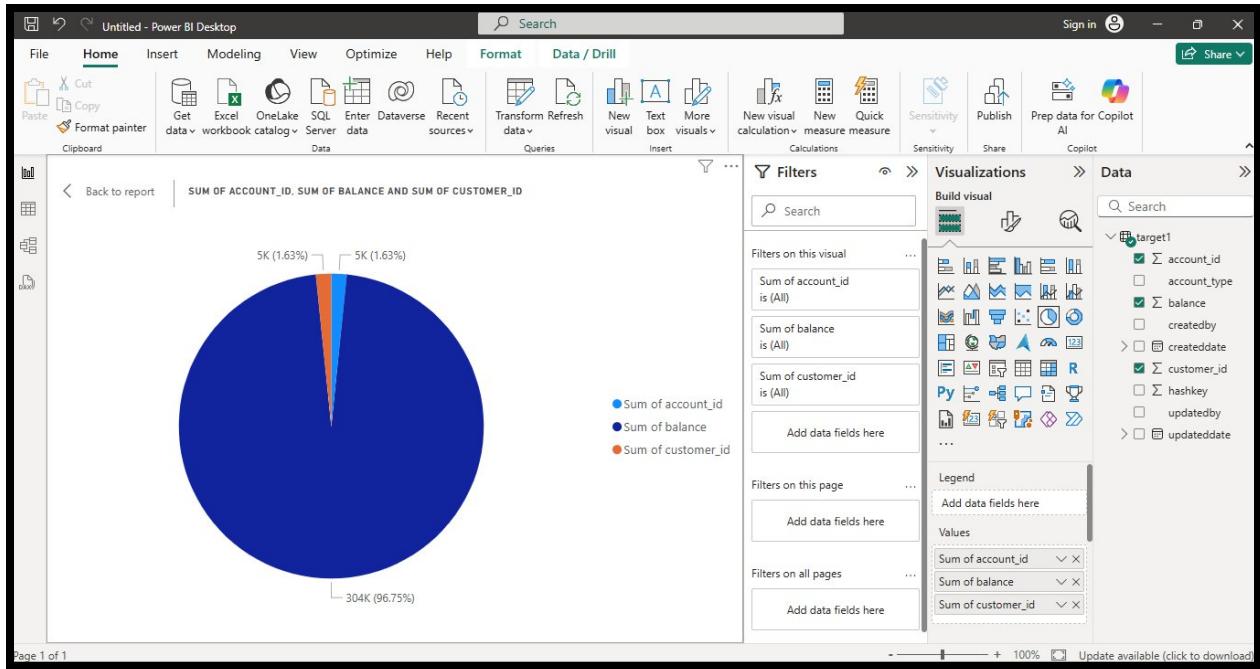
POWER BI:-

The screenshot shows the Power BI Navigator interface. On the left, there is a tree view of available tables and queries. The 'target1' table is selected and highlighted in grey. The main area displays the schema and data of the 'target1' table. The columns are: account_id, customer_id, account_type, balance, hashkey, createdby, createddate, updatedby, and updateddate. The data consists of approximately 30 rows of account information. At the bottom right of the table view, there are buttons for 'Load', 'Transform Data', and 'Cancel'.

SELECTED TARGET1 TABLE.

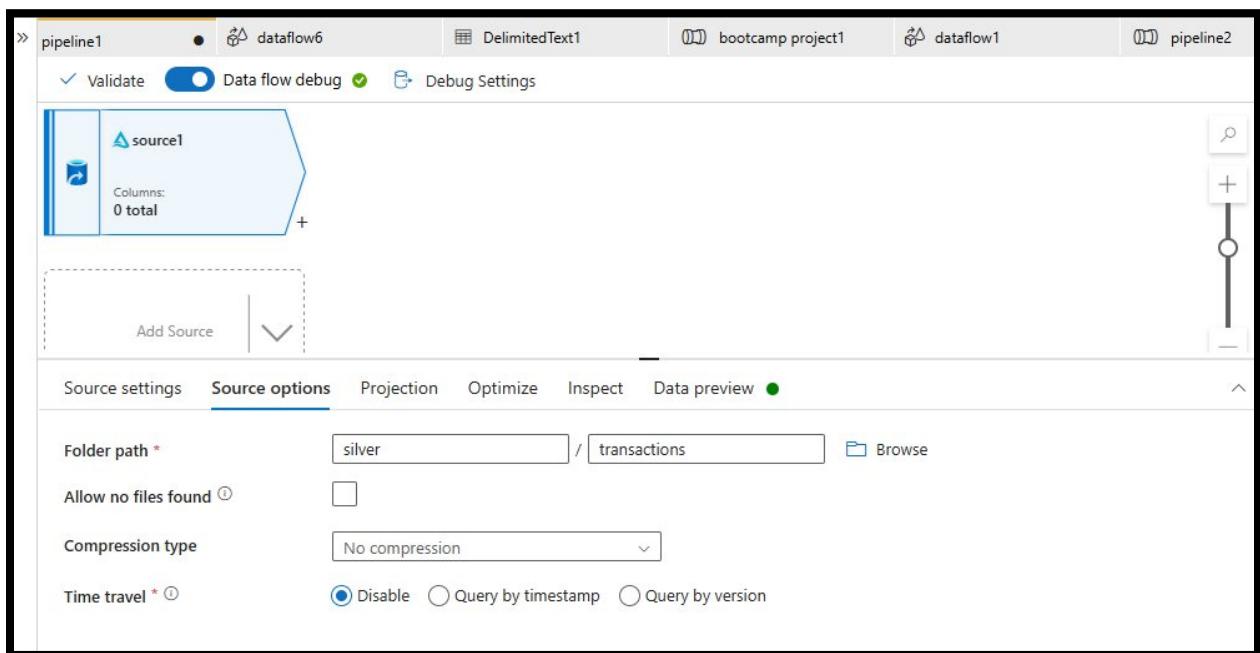
The screenshot shows a Power BI Desktop report titled 'Untitled - Power BI Desktop'. The main visual is a bar chart titled 'SUM OF ACCOUNT_ID BY ACCOUNT_TYPE'. The Y-axis is labeled 'Sum of account_id' and ranges from 0% to 100%. The X-axis is labeled 'account_type' and has three categories: 'Checking', 'Savings', and 'saving'. All three bars are blue and reach the 100% mark. The chart is located in the 'Visualizations' pane. The 'Table tools' ribbon is visible at the top, showing the current table is 'target1'. The 'Data' pane on the right lists the columns of the 'target1' table, with 'account_id', 'account_type', and 'sum of account_id' checked. Other columns like 'balance', 'createdby', etc., are listed but not checked.

COLUMNAR CHART OF TABLE TARGET1



PIE CHART VISUAL REPRESENTATION OF TARGET1.

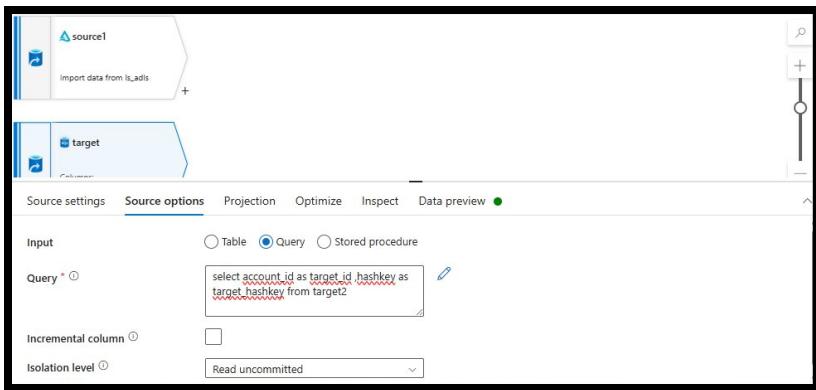
2. SC Dtype1-> TRANSACTIONS.



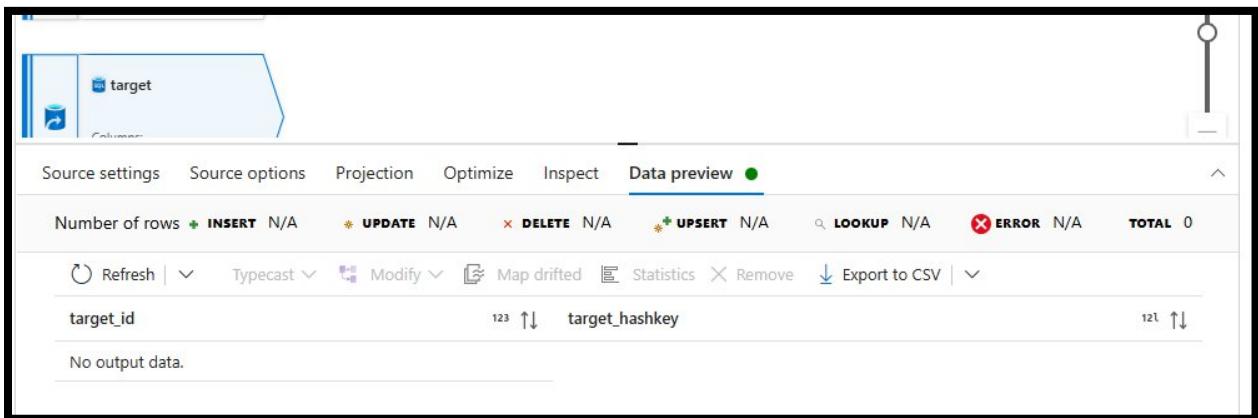
Run Cancel ⌂ Disconnect ⌂ Change Database: sqldb Estimated Plan Enable Actual Plan Parse
1 create table target2(transaction_id INT IDENTITY(1,1),account_id int,transaction_date DATETIME, transaction_amount FLOAT, transaction_type VARCHAR(100),hashkey b
2 SELECT * FROM TARGET2
3

Results Messages

transaction_id	account_id	transaction_date	transaction_amo...	transaction_type	hashkey	createdby	createddate	updatedby



Target table



Generate hashkey using crc32 by selecting derived column->

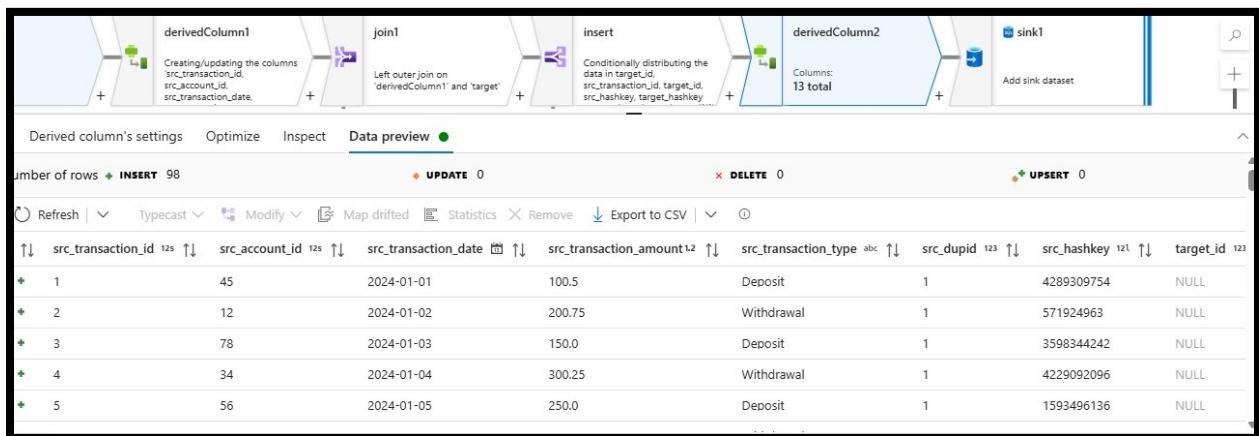
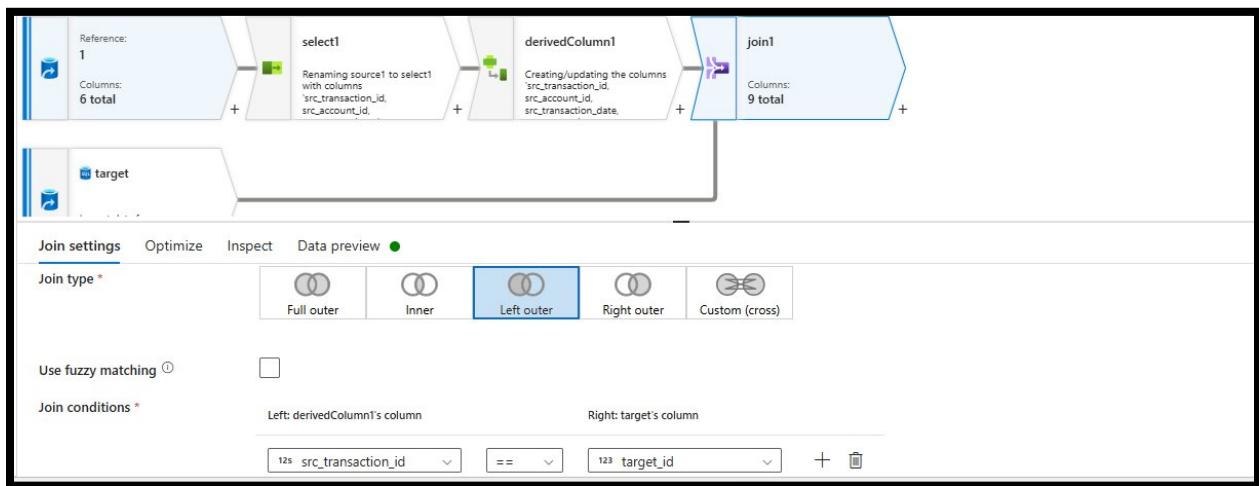
Column name *

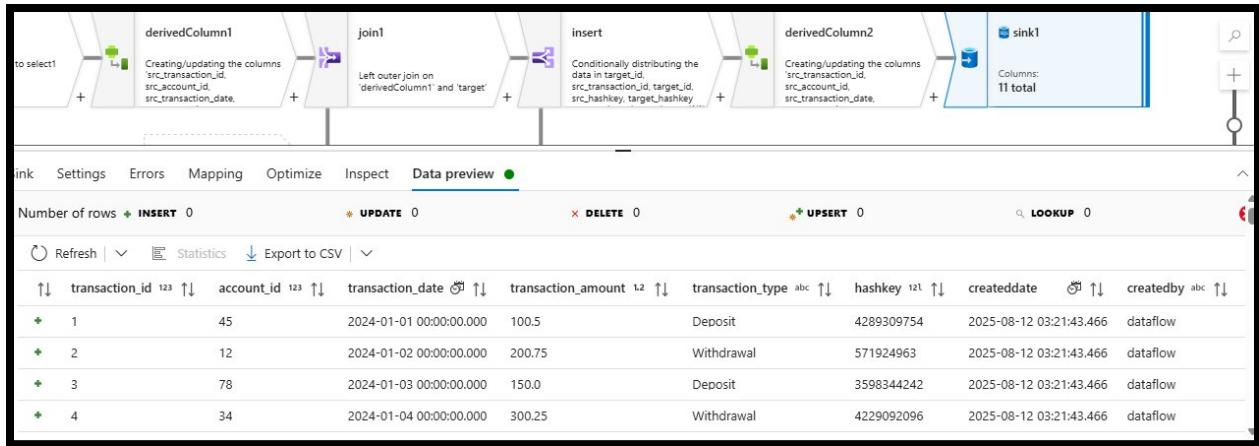
Expression

```
crc32(concat(toString(src_transaction_id),toString(src_account_id),toString(src_transaction_date),toString(
```

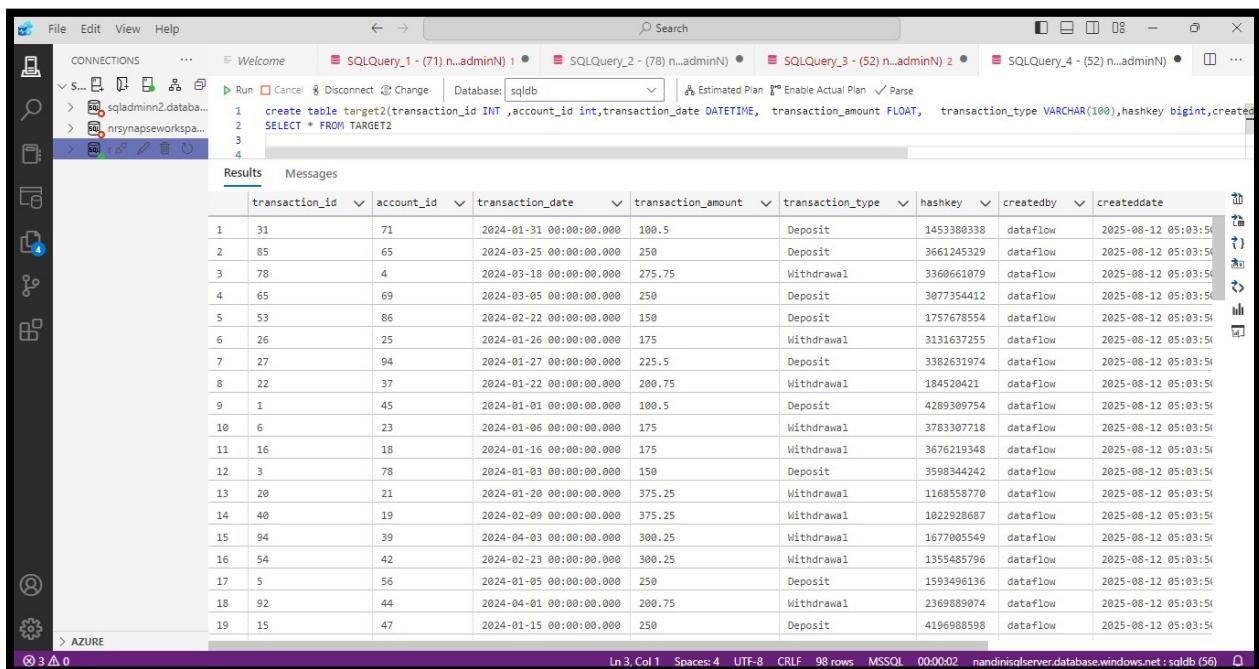
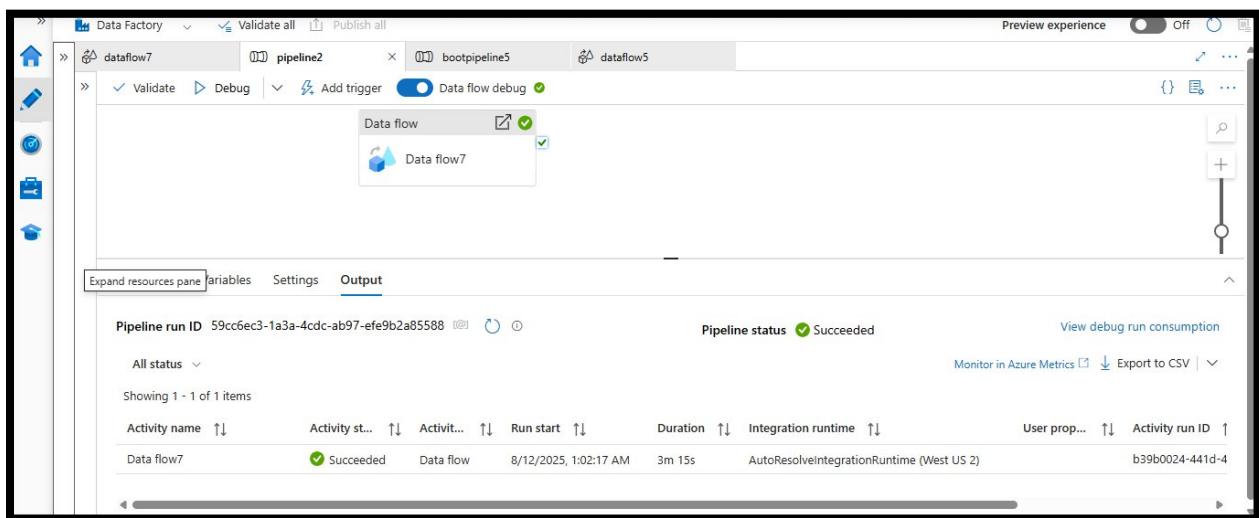
+ - * / || && ! ^ == === <=> != > < >= <= []

Join





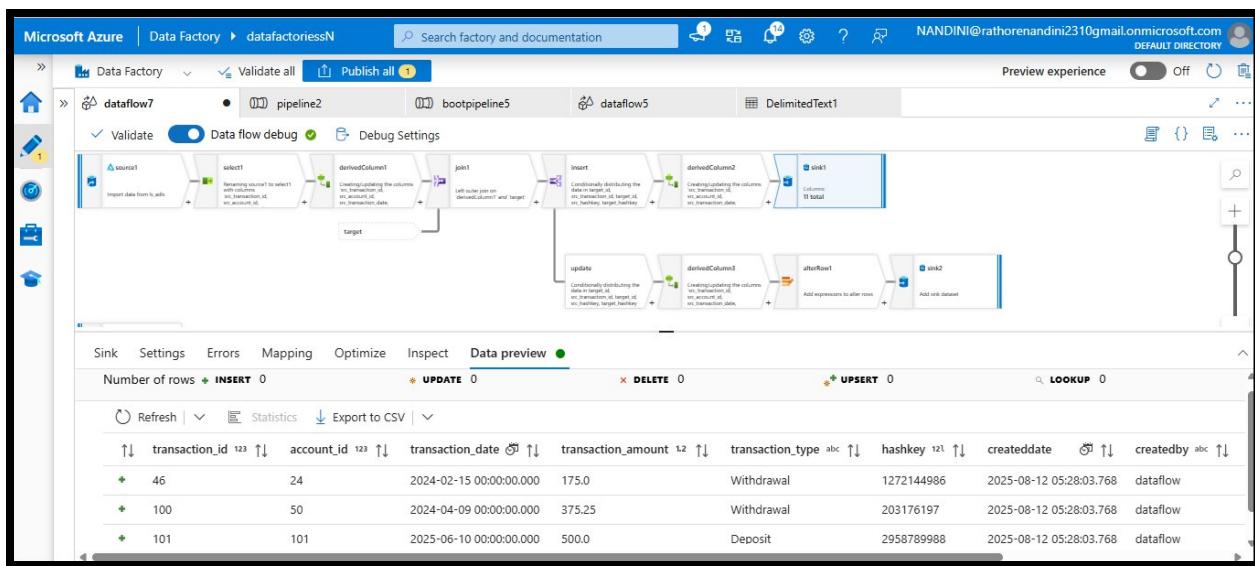
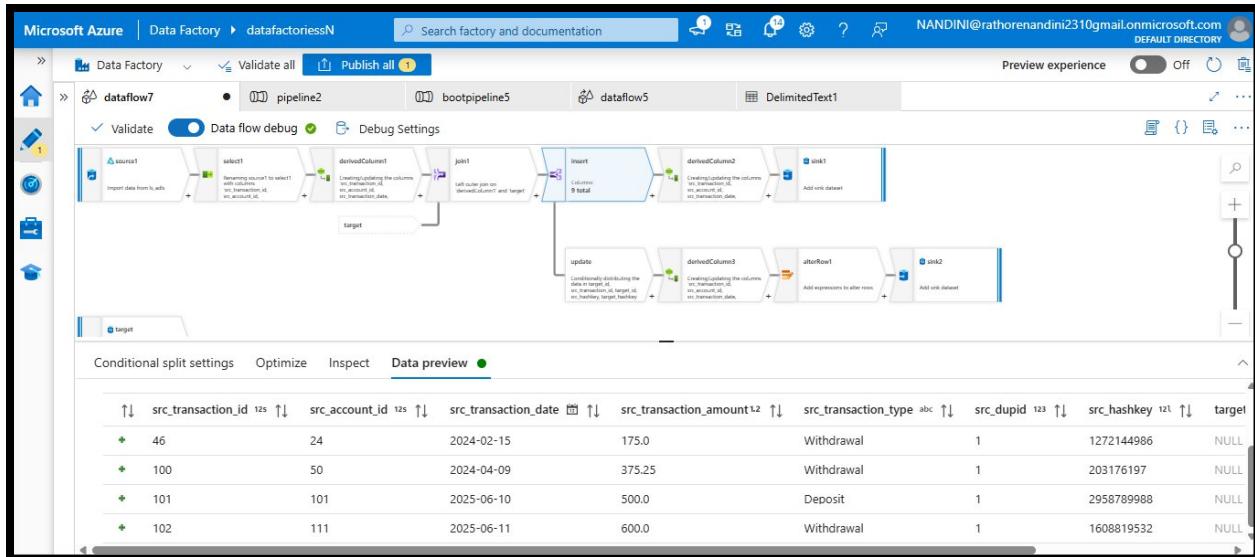
Sink data preview of inserted data.



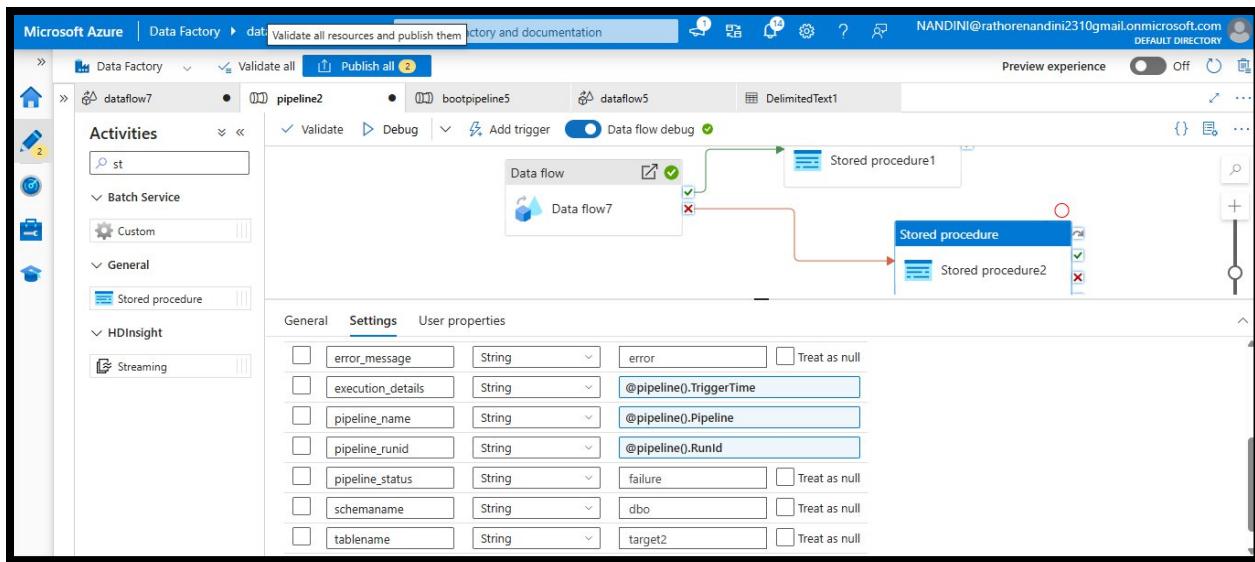
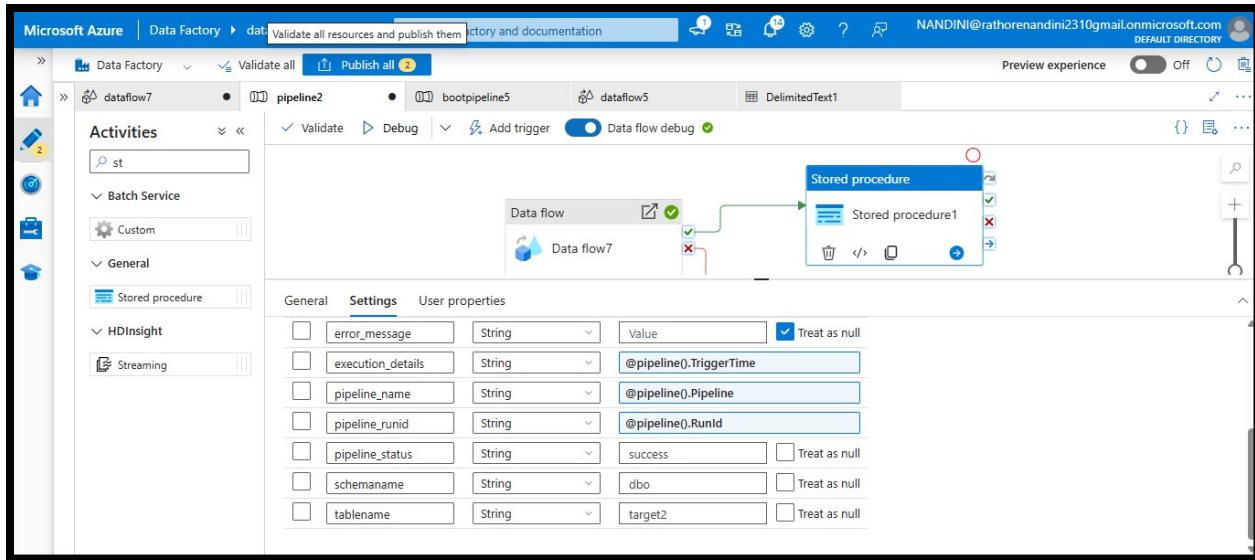
Now load the updated data as a source.

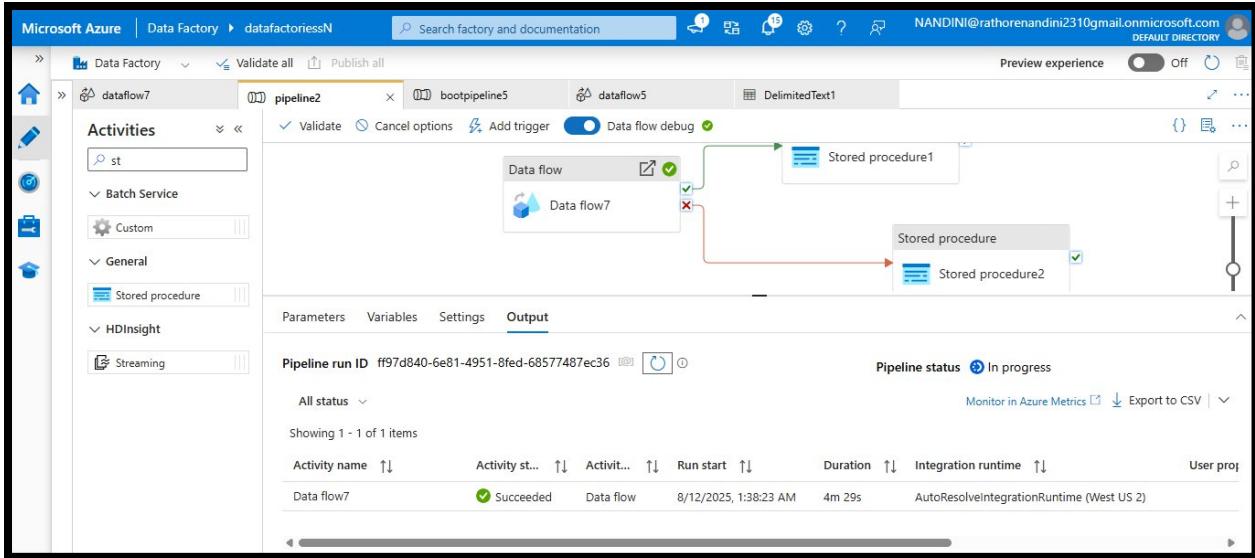
The screenshot shows the Azure Data Factory pipeline run status. The pipeline run ID is 5991c16e-a403-44fe-ad16-4dfa691df435, and the Pipeline status is Succeeded. The run started on 8/12/2025, 1:18:37 AM, and took 1m 39s. The activity Data flow5 is listed as Succeeded. The interface includes a sidebar for activities like Move and transform, Synapse, and Azure Function, and a preview experience toggle.

The screenshot shows the Azure Data Factory pipeline editor. The pipeline is named pipeline2. The data flow activity is selected, showing its detailed steps: source, select1, derivedColumn1, join, target, insert, derivedColumn2, sink1, update, derivedColumn3, alterRow1, and sink2. Below the data flow, a Data preview pane is open, showing the results of the pipeline execution. The preview pane displays 98 rows with columns target_id and target_hashkey, with values 71 and 1453380338, and 65 and 3661245329 respectively.



Sink of inserted columns .





The screenshot shows the SQL Server Management Studio (SSMS) interface. It displays the results of a query that created a table named 'target2' and inserted data into it. The table has columns: transaction_id, account_id, transaction_date, transaction_amount, transaction_type, hashkey, createdby, and createddate. There are 7 rows of data shown in the results grid.

transaction_id	account_id	transaction_date	transaction_amount	transaction_type	hashkey	createdby	createddate
1	31	2024-01-31 00:00:00.000	100.5	Deposit	1453380338	dataflow	2025-08-12 05:03:50
2	85	2024-03-25 00:00:00.000	250	Deposit	3661245329	dataflow	2025-08-12 05:03:50
3	78	2024-03-18 00:00:00.000	275.75	Withdrawal	3360661079	dataflow	2025-08-12 05:03:50
4	65	2024-03-05 00:00:00.000	250	Deposit	3077354412	dataflow	2025-08-12 05:03:50
5	53	2024-02-22 00:00:00.000	150	Deposit	1757678554	dataflow	2025-08-12 05:03:50
6	26	2024-01-26 00:00:00.000	175	Withdrawal	3136137255	dataflow	2025-08-12 05:03:50
7	27	2024-01-27 00:00:00.000	225.5	Deposit	3382631974	dataflow	2025-08-12 05:03:50

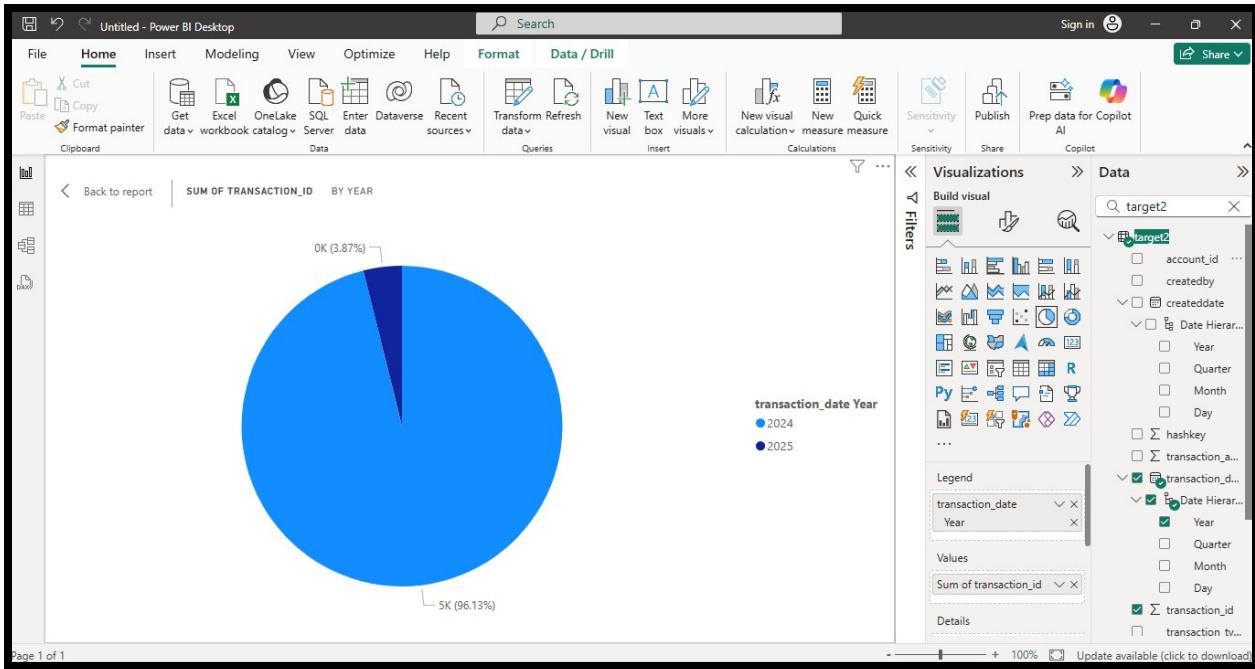
Target2 table in sql db after upsert in dataflow in azure data factory.

The screenshot shows the SSMS interface again. It runs a stored procedure named 'log_proc' which inserts data into a table called 'logging_table'. The results show two rows of data inserted into this table, each corresponding to a pipeline run.

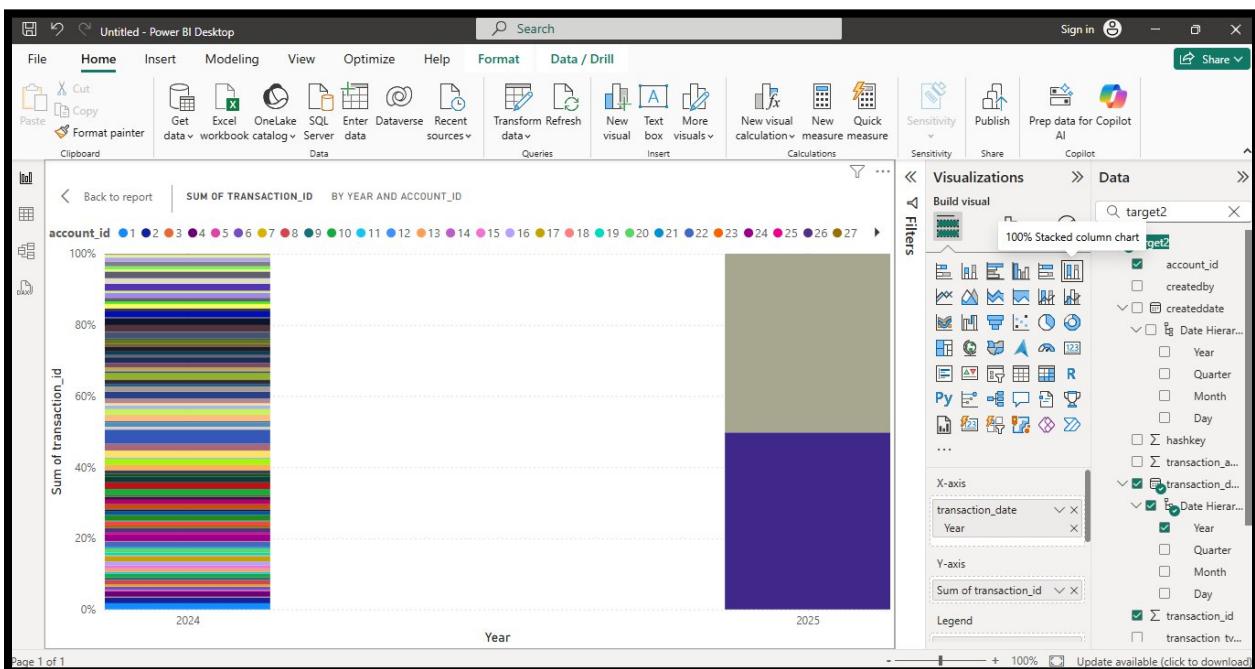
id	tablename	schemaname	error_message	pipeline_status	pipeline_id	pipeline_runid	execution_details
1	target1	dbo	NULL	success	pipeline1	36c8635c-41a8-4eed-8173-cf4275207e6f	2025-08-12T02:45:15.841
2	target2	dbo	NULL	success	pipeline2	ff97d840-6e81-4951-8fed-68577487ec36	2025-08-12T05:38:15.942

Logging table showing the pipeline status.

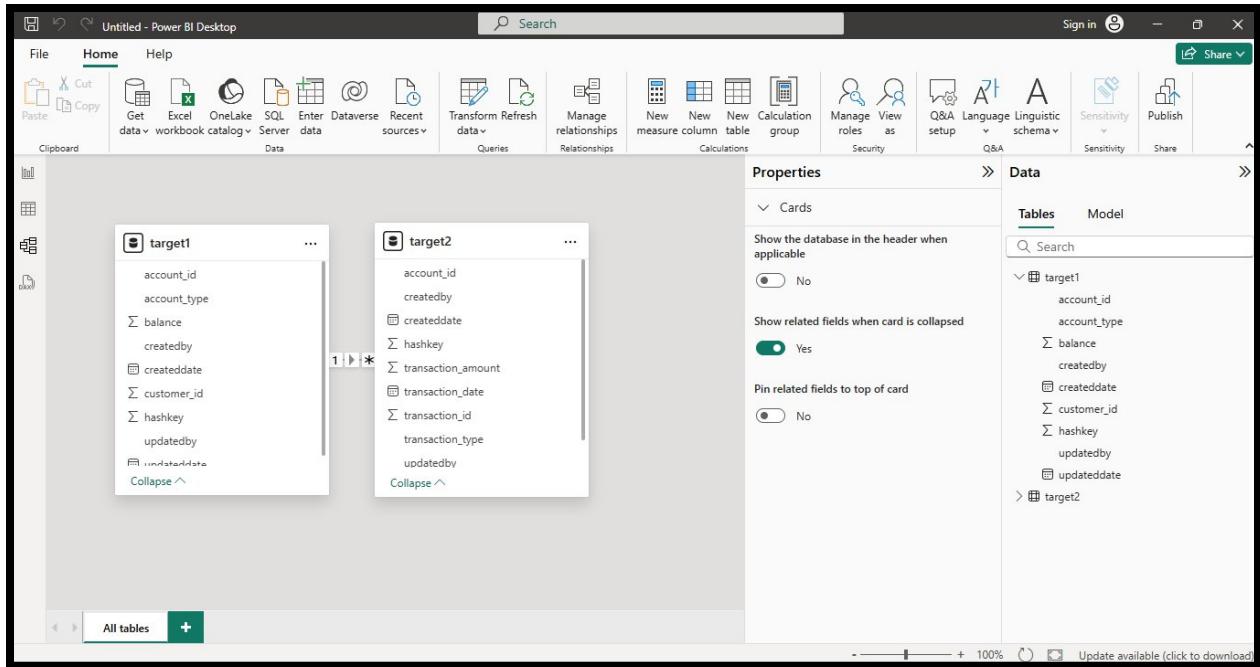
POWER BI REPORT:- PIE CHART



COLUMNAR CHART



MODEL VIEW TWO TABLES ARE CONNECTED ON THE BASIS OF ACCOUNT ID



Conclusion

The "Data Pipeline for Customer Account Analysis" project successfully demonstrates the design and implementation of a secure, efficient, and scalable data processing workflow using Azure Data Factory, Azure Data Lake Storage, and Azure SQL Database. By integrating Azure Key Vault for credential management, the solution ensures data security while adhering to best practices.

Through the staged architecture (Bronze → Silver → Gold), raw customer account data from the backend storage was ingested, cleansed, transformed, and loaded into the target SQL Database using SCD-1 upsert logic. This enabled both new data insertion and existing data updates without duplication.

The project not only streamlined the ETL process but also laid a foundation for advanced analytics by integrating the processed data with Power BI, enabling the generation of meaningful business insights. Overall, the pipeline meets its objectives of reliability, maintainability, and scalability, ensuring that downstream analytics and reporting can be performed on accurate and up-to-date data.

THANKYOU!