

Incremental Data Load with UPSERT using Azure Data Factory

1. Objective

The objective of this implementation is to **load data incrementally into Azure SQL Database** by performing **UPSERT operations** using **Azure Data Factory**. The solution ensures that **new records are inserted** and **existing records are updated** based on key columns, avoiding full data reloads and maintaining data consistency.

2. What is UPSERT?

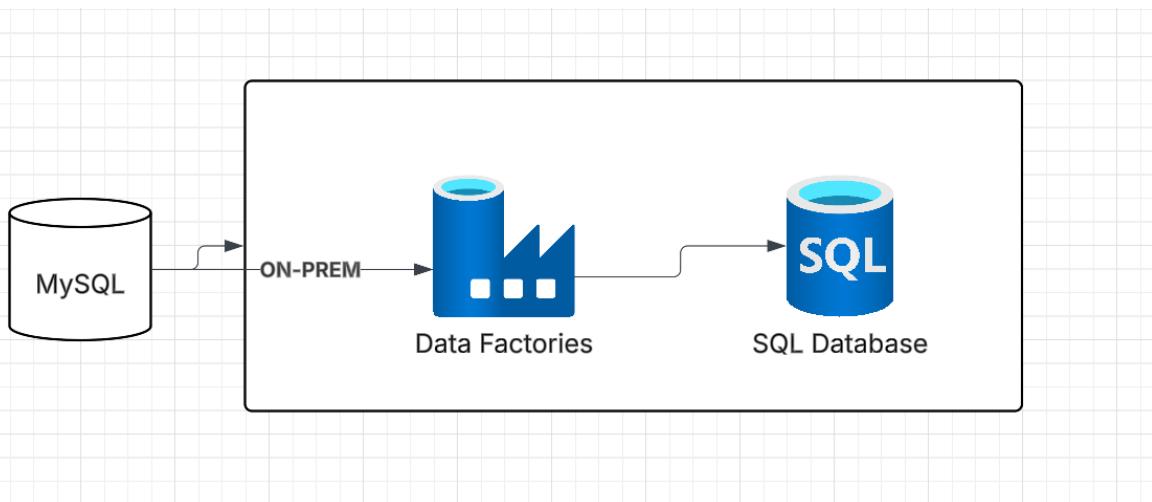
UPSERT is a combination of:

INSERT → when a record does not exist in the target

UPDATE → when a record already exists in the target

In Azure Data Factory, UPSERT is used in **Copy Activity** to efficiently handle **incremental data loads**.

3. PROJECT ARCHITECTURE



4. Resources Required

- Azure Data Factory
- Azure SQL Database
- Source system (MySQL / ADLS)

- Azure Integration Runtime / Self-hosted IR
- Primary / Business key columns

5. Creating Link Service

- Mysql to Azure link service
- Azure SQL Database to Azure Link Service

The screenshot shows the 'Linked services' configuration page in the Azure Data Factory interface. On the left, the navigation menu is visible with 'Connections' selected under 'Connections'. The main area is titled 'Linked services' with the sub-instruction 'Linked service defines the connection information to a data store'. A 'New' button is present. The configuration fields include:

- Port:** 3306
- Database name ***: auto_mig
- User name ***: root
- Password** (radio button selected): Azure Key Vault (button)
- Password ***: (password input field)
- SSL mode**: Preferred
- Use system trust store** (radio button): Not use system trust store (selected)

A green checkmark indicates 'Connection successful'. Buttons at the bottom include 'Create', 'Back', 'Test connection', and 'Cancel'.

Creating Mysql2adf link service

The screenshot shows the 'Linked services' configuration page in the Azure Data Factory interface. On the left, the navigation menu is visible with 'Connections' selected under 'Connections'. The main area is titled 'Linked services' with the sub-instruction 'Linked service defines the connection information to a data store'. A 'New' button is present. The configuration fields include:

- Account selection method**: From Azure subscription (radio button selected)
- Azure subscription**: Azure subscription 1 (c839e411-b4b3-41d8-a904-3f8cfefbd75)
- Server name ***: migrserver
- Database name ***: migdb
- Authentication type ***: SQL authentication
- User name ***: mahi_123

A green checkmark indicates 'Connection successful'. Buttons at the bottom include 'Create', 'Back', 'Test connection', and 'Cancel'.

Creating azure sql to adf link service

Link services

The screenshot shows the Azure Data Factory interface. On the left, there's a navigation sidebar with icons for Home, General, Connections, Source control, and Author. Under 'Connections', 'Linked services' is selected. The main area is titled 'Linked services' and contains a sub-instruction: 'Linked service defines the connection information to a data store or compute. Learn more'. Below this is a '+ New' button and a search/filter bar with 'Annotations: Any'. A table lists two items:

Name	Type	Related	Annotations
adf2sql	Azure SQL Database	0	
on_prem_mysql	MySQL	0	

6. INTEGRATION RUNTIME SETUP

- In the process of creating the link service for Mysql to azure
- We need to create the self-host integration runtime

Integration runtime setup

Network environment:

Choose the network environment of the data source / destination or external compute to which the integration runtime will connect to for data flows, data movement or dispatch activities:



Azure

Use this for running data flows, data movement, external and pipeline activities in a fully managed, serverless compute in Azure.



Self-Hosted

Use this for running activities in an on-premises / private network

[View more ▾](#)

External Resources:

You can use an existing self-hosted integration runtime that exists in another resource. This way you can reuse your existing infrastructure where self-hosted integration runtime is setup.



Linked Self-Hosted

[Learn more ↗](#)

[Continue](#)

[Back](#)

[Cancel](#)

SELF HOST INTEGRATION RUNTIME

- We need to download the integration runtime

Check migration readiness Run an assessment to identify which resources could be upgraded to Microsoft Fabric.

Data Factory Validate all Publish all

Linked services

Linked service defines the connection information to a data store.

+ New

Filter by name Annotations : Any

If you expect to see

Integration runtime setup

integrationRuntime1

Self-contained interactive authoring ⓘ

Disable Enable

Option 1: Express setup

[Click here to launch the express setup for this computer](#)

Option 2: Manual setup

Step 1: [Download and install integration runtime](#)

Step 2: Use this key to register your integration runtime

Name	Authentication key
Key1	IR@69a37397-c40e-45de-9419-58876390aae3@migratioadf@Serv
Key2	IR@69a37397-c40e-45de-9419-58876390aae3@migratioadf@Serv

Close

- After installation of IR copy the key from azure and paste in checkbox of IR

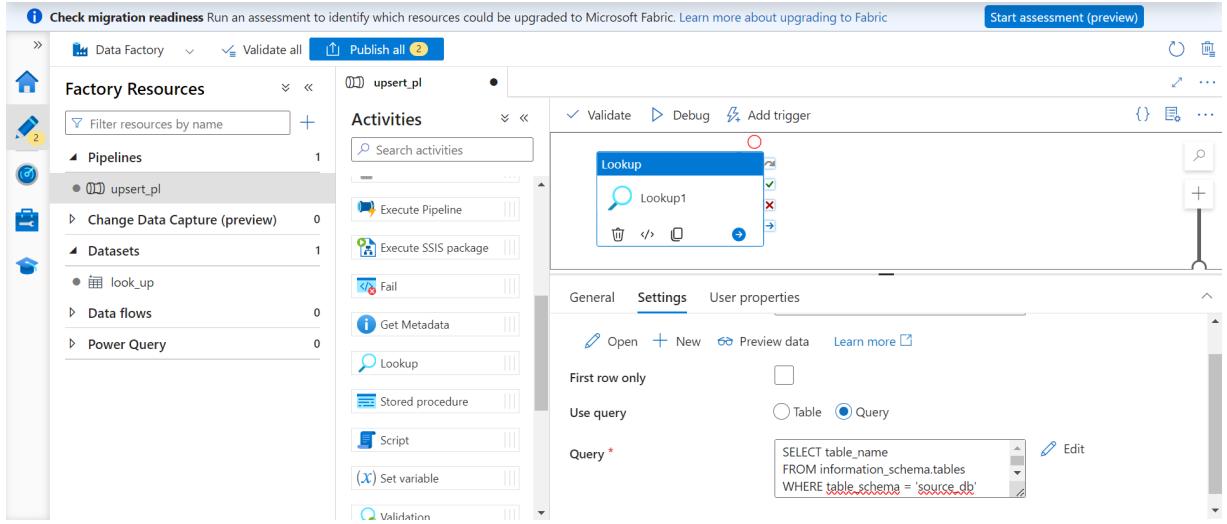
The screenshot shows the Microsoft Integration Runtime Configuration Manager interface. The title bar reads "Microsoft Integration Runtime Configuration Manager". The main header says "Register Integration Runtime (Self-hosted)". A welcome message states: "Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key." Below this is a large input field for the authentication key, preceded by a "Show Authentication Key" checkbox and a link to "Learn how to find the Authentication Key". Under the heading "HTTP Proxy", there is a "Current Proxy: No proxy" section with a "Change" link. A green checkmark icon indicates that the "Integration Runtime (Self-hosted) node has been registered successfully". A note below says: "Note: You can associate up to 4 physical nodes with a Self-hosted Integration Runtime. This enables high availability and scalability for the Self-hosted Integration Runtime. We recommend you setup at least 2 nodes for higher availability. [See Integration Runtime \(Self-hosted\) article for details.](#)" At the bottom are "Launch Configuration Manager" and "Close" buttons.

COPY ANY OF THE KEY AND PASTE

7. Implementation Steps

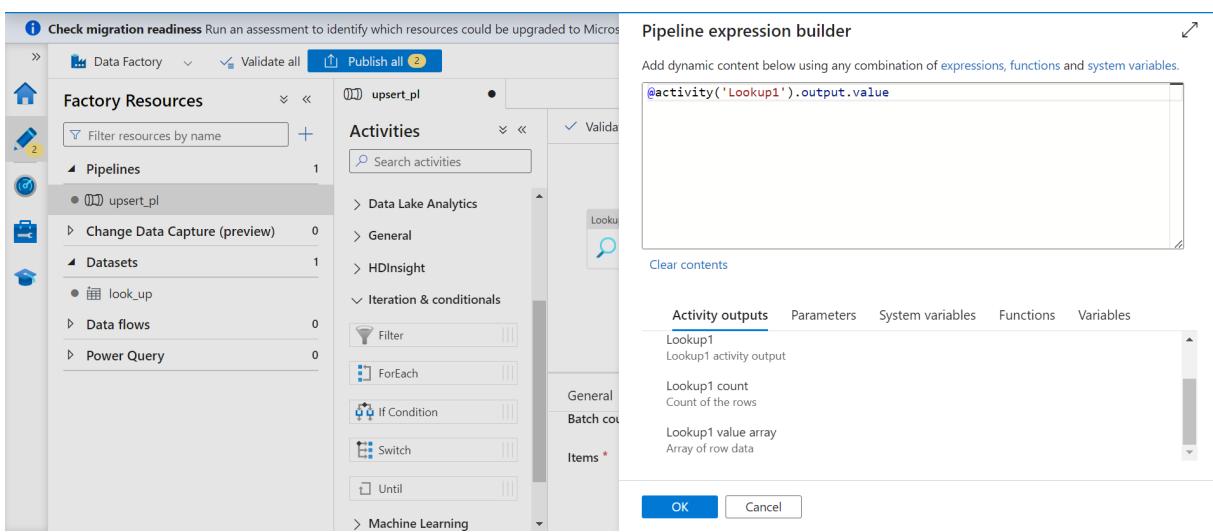
➤ COPY ACTIVITY

- Drag the copy activity and fill the expression with the query
- The query should display the table_name in the database



➤ FOREACH ACTIVITY

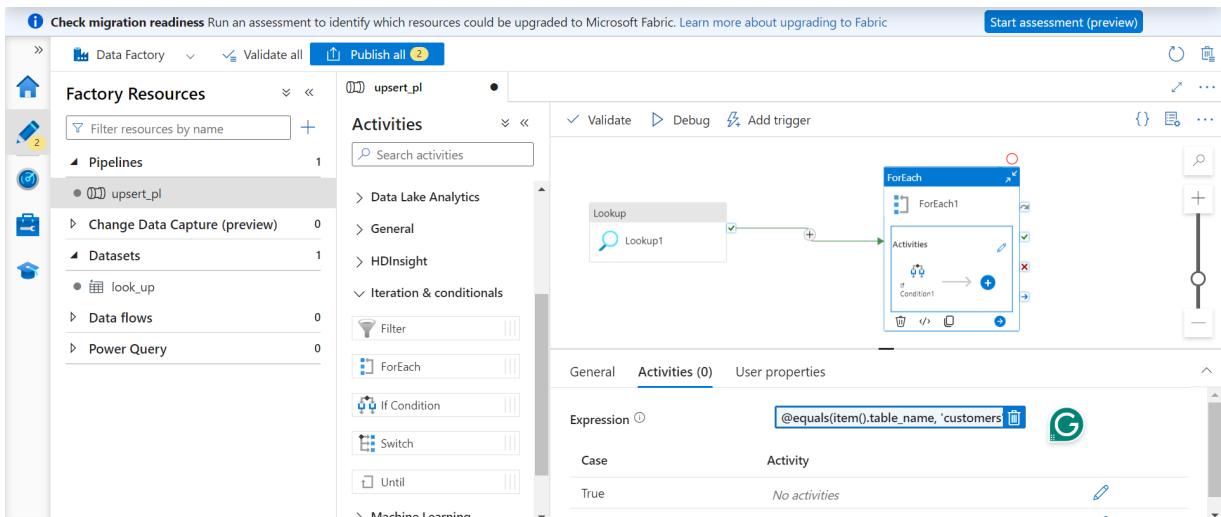
- Write a query to extract the values from the json file created by the lookup



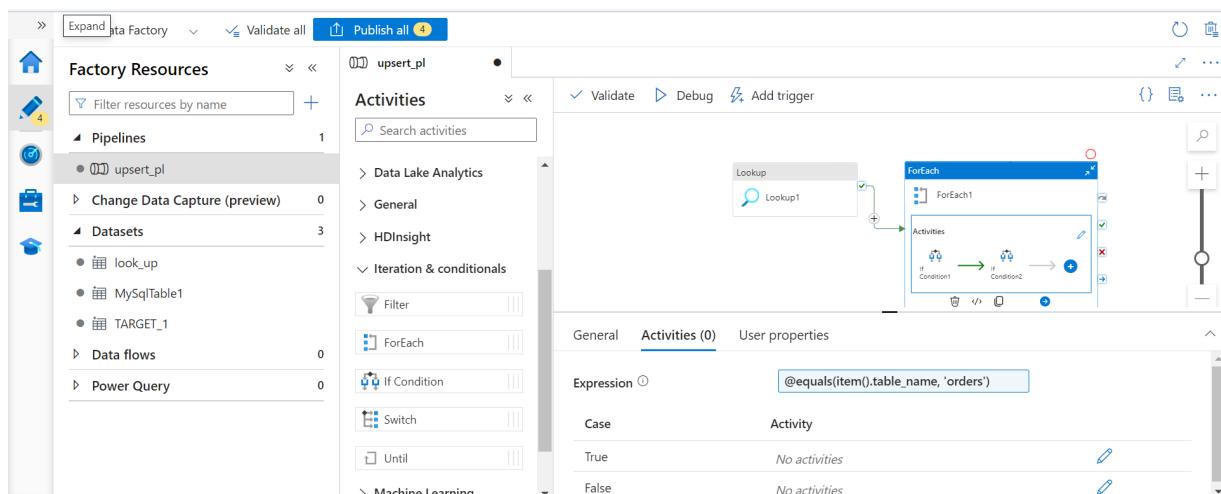
- Inside each for each activity we are going to add if condition activity
- Here we need to have copy activity inside every if condition
- Here the problem is if there are more tables then we need more if conditions and copy for each copy activity it creates two datasets

➤ IF CONDITION

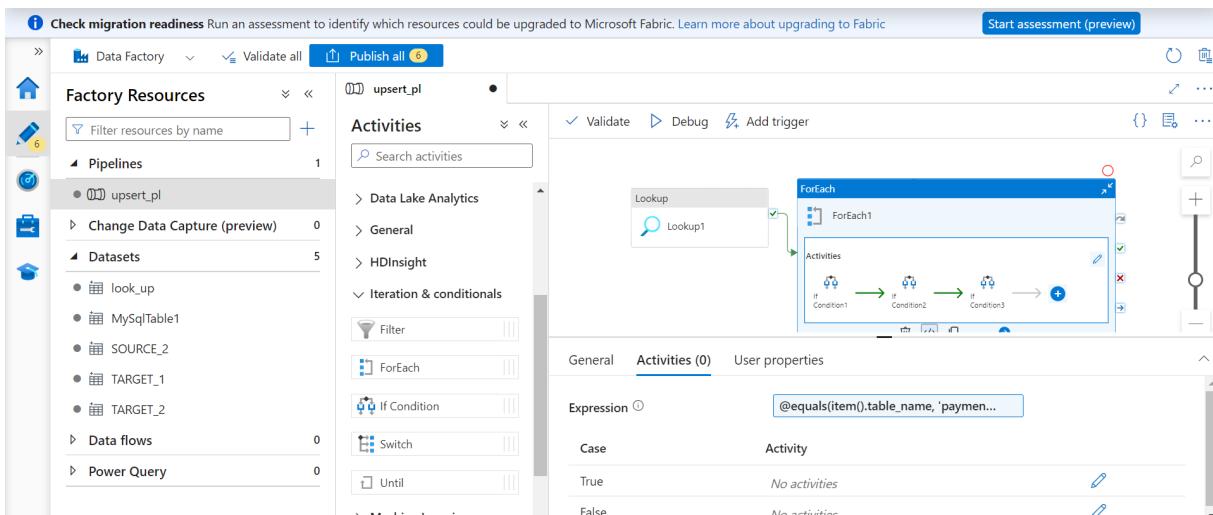
- In your project there are three tables so we need have three if conditions
- For each if condition we are going to use equal method if true then the copy activity is going execute
- The three tables are customers,orders,payments.



FIRST IF CONDITION FOR CUSTOMERS



SECOND IF CONDITION FOR ORDERS



THIRD IF CONDITION FOR PAYMENTS

➤ COPY ACTIVITY

- In each copy activity I am going to create two datasets one for source and other for sink
- ❖ **SOURCE-1**
 - In the source we are going to write code by using the method concat to write a query
 - For every source in copy activity we are going to perform same

SOURCE DATASET

QUERY TO CALL THE TABLE

❖ COPY-1

- In the copy we need to follow some rules and this same for other copy activity also
- First you need to create tables in the azure sql database and need assign primary key
- During creation of sink dataset we need to select the table
- In the sink part we need to select the upsert and use existing table
- And we need enter the primary key in the key column area

SELECTING THE TABLE

SELECTING THE UPSERT

➤ SOURCE-2

Set properties

Name: SOURCE_2
Linked service: upsert_ls
Connect via integration runtime: integrationRuntime2
Table name: Loading...
OK Back Cancel

Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat('select * from ',item().TABLE_NAME)
```

Activity outputs

Search: Copy data1
Copy data1 activity output
Lookup1

Parameters System variables Functions Variables

OK Cancel

SOURCE FOR ORDERS TABLE

➤ COPY-2

The screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Factory Resources' pane lists Pipelines, Datasets, and Power Query. The 'Activities' pane shows an 'upsert_pl' pipeline with an 'upsert' activity selected. A 'Set properties' dialog is open on the right, titled 'Set properties'. It contains the following fields:

- Name: TARGET_2
- Linked service *: AzureSqlDatabase1
- Table name: dbo.orders
- Import schema: From connection/store (radio button selected)

At the bottom of the dialog are 'OK', 'Back', and 'Cancel' buttons.

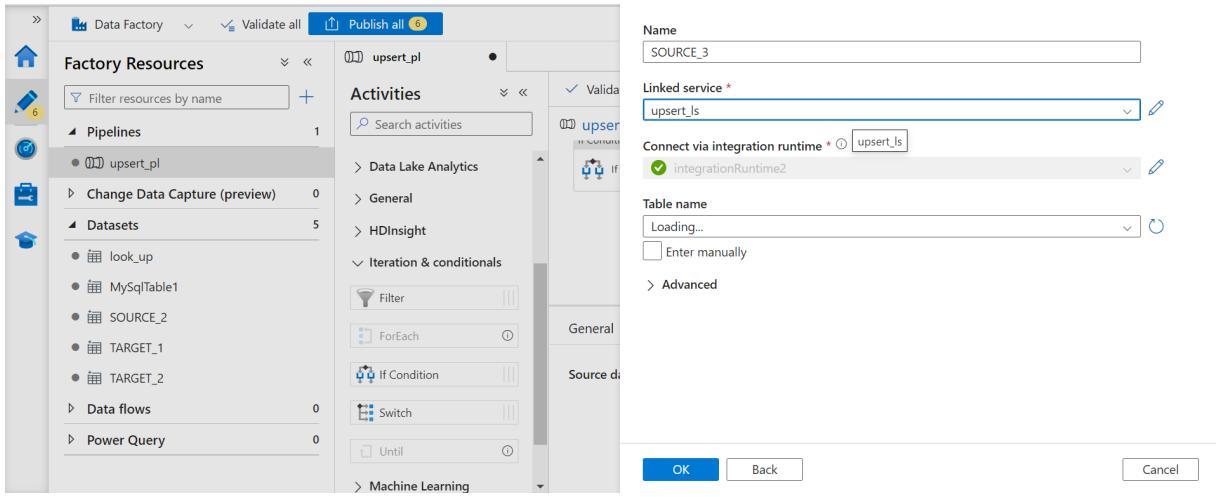
SELECTING THE ORDERS TABLE

The screenshot shows the Azure Data Factory pipeline editor with a more complex pipeline structure. The 'Activities' pane shows an 'upsert_pl' pipeline with an 'upsert' activity, which has an 'ForEach' loop and an 'If Condition2 > True' branch. The 'Sink' tab is selected in the configuration pane. The configuration includes:

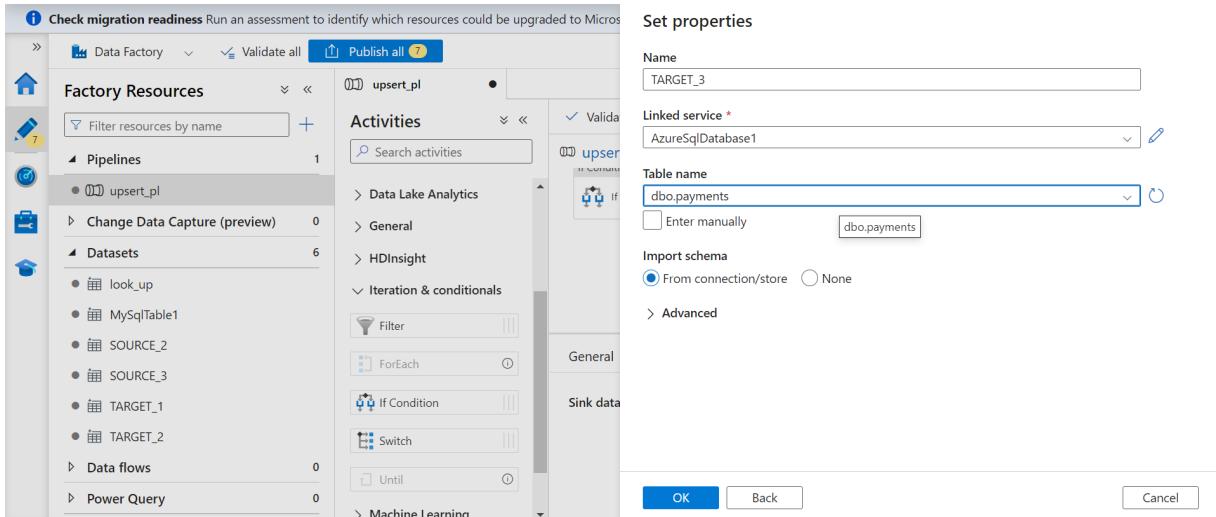
- Write behavior: Upsert (radio button selected)
- Use tempdb: checked
- Key columns: order_id
- Bulk insert table lock: No (radio button selected)

SELECTING UPSETR, USE EXISTING

➤ SOURCE-3



➤ COPY-3



SELECTING THE LAST TABLE PAYMENTS

Factory Resources

- Pipelines
 - upsert_pl
- Datasets
 - look_up
 - MySQLTable1
 - SOURCE_2
 - SOURCE_3
 - TARGET_1
 - TARGET_2
 - TARGET_3
- Data flows

Activities

upsert_pl > ForEach1 > If Condition3 > True activities

General Source Sink Mapping Settings User properties

Write behavior: Insert (radio button), Upsert (radio button selected), Stored procedure

Use tempdb: checked

Key columns: payment_id

Bulk insert table lock: No

Table option: Use existing

SELECTING THE UPSERT, USE EXISTING

➤ PUBLISH AND TRIGGER

Check migration readiness

Factory Resources

- Pipelines
 - upsert_pl
- Datasets
 - look_up
 - MySQLTable1
 - SOURCE_2
 - SOURCE_3
 - TARGET_1
 - TARGET_2
 - TARGET_3
- Data flows

Activities

upsert_pl > ForEach1 > If Condition3 > True activities

Parameters

+ New

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (8)

NAME	CHANGE	EXISTING
upsert_pl	(New)	-
look_up	(New)	-
MySQLTable1	(New)	-
TARGET_1	(New)	-
SOURCE_2	(New)	-
TARGET_2	(New)	-
SOURCE_3	(New)	-
TARGET_3	(New)	-

Publish **Cancel**

Pipeline run

Trigger pipeline now using last published configuration.

Parameters

Name	Type	Value
No records found		

OK **Cancel**

➤ EXCUTION

All pipeline runs > upsert_pl - Activity runs

Rerun Cancel Refresh Update pipeline List Gantt

Activity runs

Pipeline run ID 6c67b782-608a-4a8c-a4b4-14be354876ac

All status List Monitor in Azure Metrics Export to CSV

All pipeline runs > upsert_pl - Activity runs

Rerun Cancel Refresh Update pipeline List Gantt

Activity runs

Pipeline run ID 6c67b782-608a-4a8c-a4b4-14be354876ac

All status List Showing 1 - 14 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime
Lookup1	Succeeded	Lookup	12/21/2025, 11:42:15 PM	29s	integrationRuntime2
ForEach1	Succeeded	ForEach	12/21/2025, 11:42:45 PM	1m 3s	
If Condition1	Succeeded	If Condition	12/21/2025, 11:42:45 PM	Less than 1s	
If Condition1	Succeeded	If Condition	12/21/2025, 11:42:45 PM	54s	
If Condition1	Succeeded	If Condition	12/21/2025, 11:42:45 PM	Less than 1s	
Copy data1	Succeeded	Copy data	12/21/2025, 11:42:46 PM	53s	integrationRuntime2

➤ INSERTION

- In Mysql I am going to insert some of the rows
- And I will rerun the pipeline
- If I look at the already existing table in my there 7 rows in the customers
- And I inserted two rows into customer and rerun the pipeline after the execution the number of columns should be 9

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the schema structure, including the 'upsert' schema which contains tables like 'customers', 'orders', and 'payments'. The main area is a 'Query 1' window titled 'SQL File 3*' containing the following SQL code:

```

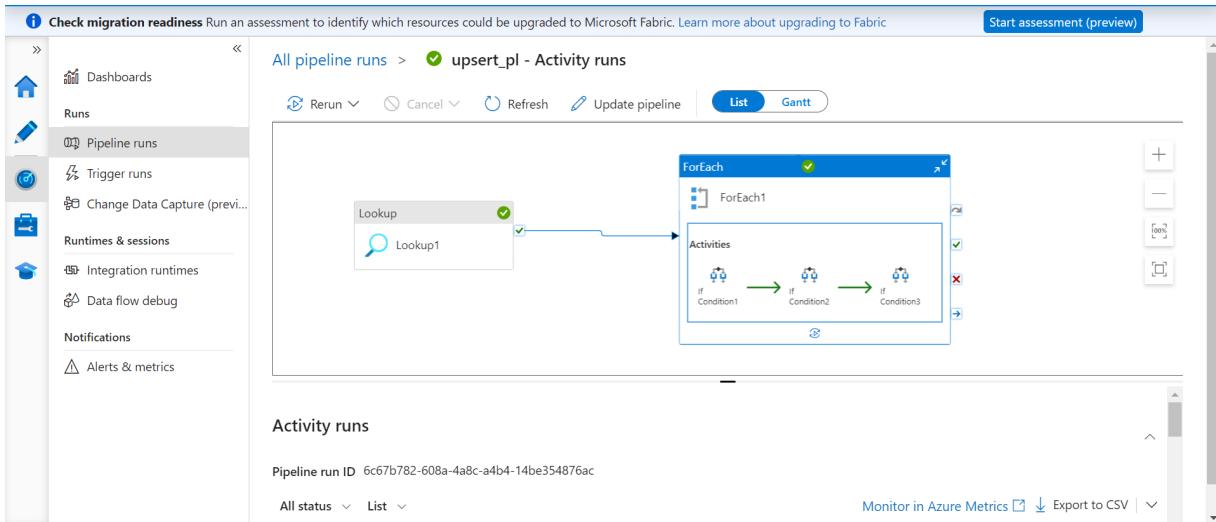
425 insert into payments values(9005,5005,'upi','s','2025-12-20');
426 insert into payments values(9006,5006,'upi','s','2025-12-20');
427 insert into payments values(9007,5007,'upi','s','2025-12-20');

429 select count(*) from customers;
430 insert into customers values(108,"manu","i@gmail.com","ap","2025-12-21");
431 insert into customers values(109,"manu","i@gmail.com","ap","2025-12-21");
432

```

The result grid below the query shows the output of the 'select count(*)' statement, which is 9.

INSERTION IN MYSQL



RERUN THE PIPELINE

migdb (kalvapallisenapathireddy@g...)

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Tables

- > dbo.addresses
- > dbo.copy1
- > dbo.customers
- > dbo.departments
- > dbo.employees
- > dbo.order_items
- > dbo.orders
- > dbo.payments

Query 1 × Query 2 × Query 3 × **Query 4 ×**

Run Cancel query Save query Export data as Show only Editor

```

1  SELECT count(*) FROM [dbo].[customers]
2
3

```

Results Messages

Search to filter items...

9

Query succeeded | 1s

TWO ROWS SUCCESFULLY INSERTED

➤ UPDATE

- In the customer table I want to update ap in city to amaravati
- I have performed update operation in mysql

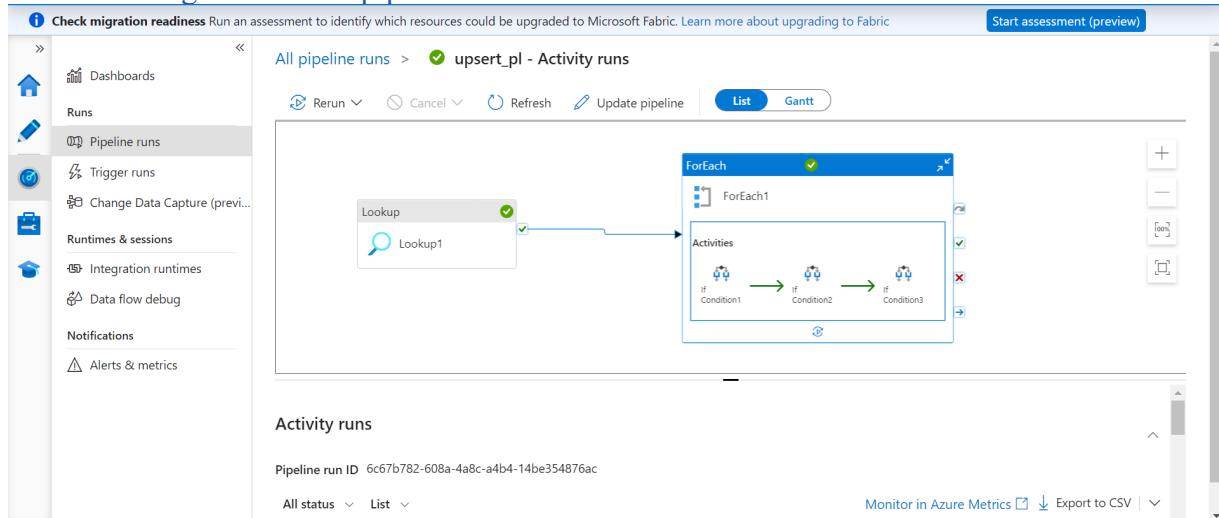
```

select count(*) from customers;
insert into customers values(108,"mani",'i@gmail.com','ap','2025-12-21');
insert into customers values(109,"mani",'i@gmail.com','ap','2025-12-21');

```

```
update customers set city = 'Amaravti' where city = 'ap';
```

- Going to rerun the pipeline



- In the Azure Sql database

The screenshot shows the Azure Data Studio interface. On the left, there's a sidebar with a message about limited object explorer and a list of tables under 'Tables'. The main area has a code editor with three lines of SQL and a results grid below it.

```
1 SELECT count(*) FROM [dbo].[customers]
2 select * from [dbo].[customers]
3
```

Results Messages

107	mani	i@gmail.com	Amaravti	2025-12	↑
108	mani	i@gmail.com	Amaravti	2025-12	↓
109	mani	i@gmail.com	Amaravti	2025-12	↓

Query succeeded | 1s

8. Conclusion

- ◆ UPSERT is suitable when the number of tables to migrate is **less** and **incremental loading** is required.
- ◆ It helps in **updating existing records** and **inserting new records** without full reload.
- ◆ Azure Data Factory does **not support dynamic UPSERT key columns**, which limits scalability.
- ◆ As the number of tables increases, the **number of datasets and If Conditions also increases**.
- ◆ This increases **pipeline complexity and maintenance effort** for large-scale migrations.
- ◆ In this project, UPSERT was used because the **table count was limited** and **incremental load was a business requirement**.