Name: - BALU LAKSHMI SUDHA

Week – 4 case study

End-to-end case study: SQL schemas → ADF → data Bricks

This document shows clear, step-by-step instructions from creating SQL schemas to running Databricks notebooks from Azure Data Factory.

overview

This guide walks through:

1. create database + tables in SQL server (lowercase) and insert sample rows
2. push raw files to azure data lake storage under raw/bronze/
3. build an ADF pipeline that reads a list of table names, loops and copies each table to blob
4. run data Bricks notebooks from ADF to validate, mask and write silver (delta)
5. transform silver into gold (business table)
6. monitoring, artifacts and demo checklist

prerequisites

- azure subscription with storage account and data Bricks workspace
- AAD user with permissions to create resources.
- azure data factory instance
- SQL server (or azure SQL) accessible from ADF or via linked service
- data Bricks token (for linking to ADF)

1. create schemas in SQL server

Run these create-table statements in your SQL server.

```
create table holders( holder_id bigint primary key, holder_code varchar(50) not null unique,
first_name varchar(50), last_name varchar(50), dob date, gender varchar(50), email
varchar(100), phone varchar(50), created_at datetime default sysdatetime() );

create table policies( policy_id bigint primary key, policy_code varchar(50) not null unique,
holder_id bigint not null, policy_type varchar(50), start_date date, end_date date,
premium_amount decimal(10,2), created_at datetime default sysdatetime() );

create table claims( claim_id bigint primary key, claim_code varchar(50) not null unique,
policy_id bigint not null, claim_date date, claim_amount decimal(10,2), status varchar(50),
created_at datetime default sysdatetime() );

create table payments( payment_id bigint primary key, payment_code varchar(50) not null
unique, policy_id bigint not null, payment_date date, amount decimal(10,2), method varchar(50),
created_at datetime default sysdatetime() );

create table agents( agent_id bigint primary key, agent_code varchar(50) not null unique,
first_name varchar(50), last_name varchar(50), phone varchar(50), email varchar(100),
created_at datetime default sysdatetime() );

create table branches( branch_id bigint primary key, branch_code varchar(50) not null unique,
branch_name varchar(100), city varchar(50), state varchar(50), phone varchar(50), created_at
datetime default sysdatetime() );

create table beneficiaries( beneficiary_id bigint primary key, beneficiary_code varchar(50) not
null unique, holder_id bigint not null, first_name varchar(50), last_name varchar(50), relation
varchar(50), phone varchar(50), created_at datetime default sysdatetime() );

create table renewals( renewal_id bigint primary key, renewal_code varchar(50) not null unique,
policy_id bigint not null, renewal_date date, new_end_date date, premium_amount
decimal(10,2), created_at datetime default sysdatetime() );

create table policy_types( type_id bigint primary key, type_code varchar(50) not null unique,
type_name varchar(100), description varchar(255), created_at datetime default sysdatetime() );

create table risk_assessments( risk_id bigint primary key, risk_code varchar(50) not null unique,
policy_id bigint not null, assessment_date date, risk_level varchar(50), notes varchar(255),
created_at datetime default sysdatetime() );
```

2. insert sample rows

insert into holders (holder_id, holder_code, first_name, last_name, dob, gender, email, phone) values (1, 'hld1001', 'john', 'smith', '1985-03-15', 'male', 'john.smith@email.com', '9876543210'), (2, 'hld1002', 'mary', 'johnson', '1990-07-22', 'female', 'mary.johnson@email.com', '9123456780'), (3, 'hld1003', 'robert', 'brown', '1978-11-05', 'male', 'robert.brown@email.com', '9988776655'), (4, 'hld1004', 'linda', 'davis', '1982-01-19', 'female', 'linda.davis@email.com', '9011223344'), (5, 'hld1005', 'david', 'miller', '1995-09-30', 'male', 'david.miller@email.com', '9090909090');

insert into policies (policy_id, policy_code, holder_id, policy_type, start_date, end_date, premium_amount) values (1, 'pol2001', 1, 'car', '2025-01-01', '2025-12-31', 15000), (2, 'pol2002', 1, 'health', '2025-02-01', '2026-01-31', 12000), (3, 'pol2003', 2, 'life', '2024-06-15', '2034-06-14', 20000), (4, 'pol2004', 3, 'health', '2025-03-01', '2026-02-28', 11000), (5, 'pol2005', 4, 'car', '2025-04-01', '2026-03-31', 14000);

insert into claims (claim_id, claim_code, policy_id, claim_date, claim_amount, status) values (1, 'clm3001', 1, '2025-05-10', 50000, 'approved'), (2, 'clm3002', 2, '2025-07-20', 20000, 'pending'), (3, 'clm3003', 3, '2025-03-15', 150000, 'rejected'), (4, 'clm3004', 4, '2025-08-01', 10000, 'approved'), (5, 'clm3005', 5, '2025-06-25', 25000, 'pending');

insert into payments (payment_id, payment_code, policy_id, payment_date, amount, method) values (1, 'pay4001', 1, '2025-01-05', 15000, 'bank transfer'), (2, 'pay4002', 2, '2025-02-05', 12000, 'credit card'), (3, 'pay4003', 3, '2024-06-20', 20000, 'upi'), (4, 'pay4004', 4, '2025-03-05', 11000, 'bank transfer'), (5, 'pay4005', 5, '2025-04-05', 14000, 'credit card');

insert into agents (agent_id, agent_code, first_name, last_name, phone, email) values (1, 'agt5001', 'emma', 'williams', '9001112233', 'emma.williams@email.com'), (2, 'agt5002', 'liam', 'jones', '9002223344', 'liam.jones@email.com'), (3, 'agt5003', 'olivia', 'martin', '9003334455', 'olivia.martin@email.com'), (4, 'agt5004', 'noah', 'lee', '9004445566', 'noah.lee@email.com'), (5, 'agt5005', 'ava', 'clark', '9005556677', 'ava.clark@email.com');

insert into branches (branch_id, branch_code, branch_name, city, state, phone) values (1, 'brn6001', 'central office', 'mumbai', 'maharashtra', '02212345678'), (2, 'brn6002', 'north branch', 'delhi', 'delhi', '01198765432'), (3, 'brn6003', 'south branch', 'chennai', 'tamil nadu', '04487654321'), (4, 'brn6004', 'east branch', 'kolkata', 'west bengal', '03376543210'), (5, 'brn6005', 'west branch', 'pune', 'maharashtra', '02011223344');

insert into beneficiaries (beneficiary_id, beneficiary_code, holder_id, relation, first_name, last_name) values (1, 'ben7001', 1, 'spouse', 'sarah', 'smith'), (2, 'ben7002', 1, 'child', 'michael', 'smith'), (3, 'ben7003', 2, 'parent', 'william', 'johnson'), (4, 'ben7004', 3, 'spouse', 'patricia', 'brown'), (5, 'ben7005', 4, 'child', 'emily', 'davis');

insert into renewals (renewal_id, renewal_code, policy_id, renewal_date, new_end_date, premium_amount) values (1, 'ren8001', 1, '2025-12-15', '2026-12-31', 15000), (2, 'ren8002', 2, '2026-01-20', '2027-01-31', 12000), (3, 'ren8003', 4, '2026-02-15', '2027-02-28', 11000), (4, 'ren8004', 5, '2026-03-10', '2027-03-31', 14000), (5, 'ren8005', 3, '2034-05-30', '2044-06-14', 20000);

insert into policy_types (type_id, type_code, type_name, description) values (1, 'typ9001', 'car', 'vehicle insurance for cars'), (2, 'typ9002', 'health', 'medical expense coverage'), (3, 'typ9003', 'life', 'life coverage and death benefits'), (4, 'typ9004', 'home', 'home property insurance'), (5, 'typ9005', 'travel', 'travel-related insurance');

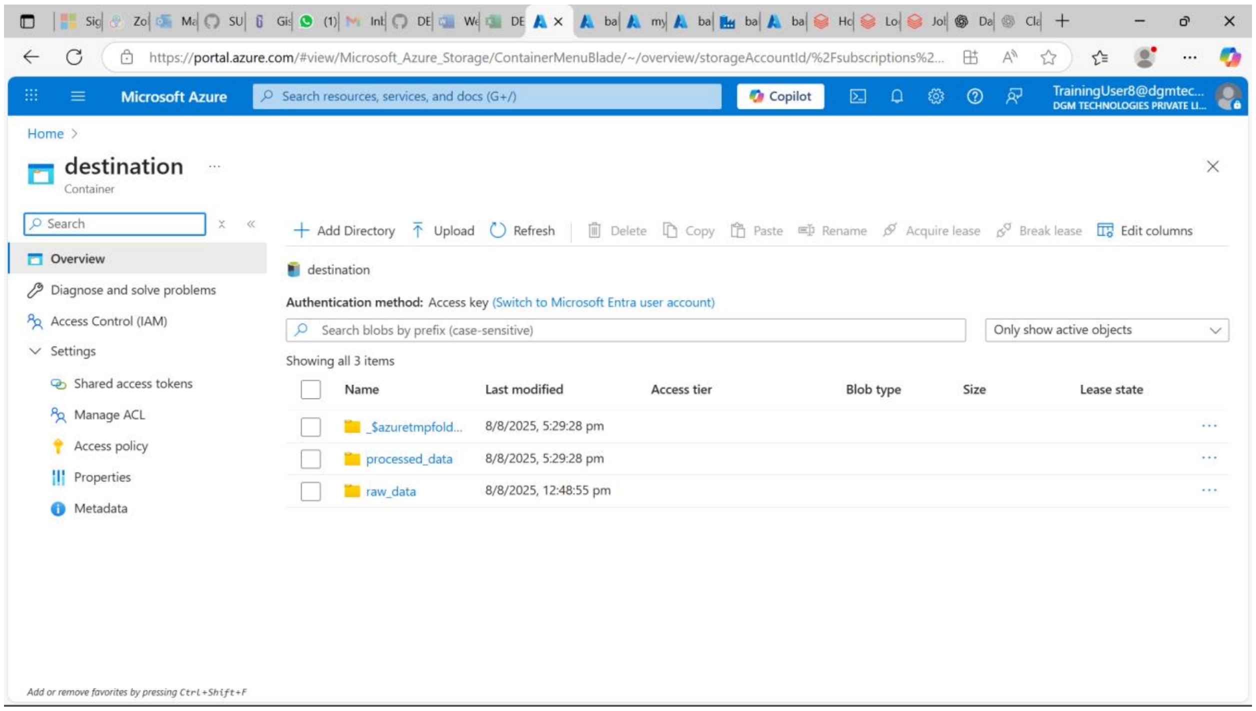
insert into risk_assessments (risk_id, risk_code, policy_id, risk_level, notes) values (1, 'rsk10001', 1, 'medium', 'minor accidents in past 3 years'), (2, 'rsk10002', 2, 'low', 'healthy medical history'), (3, 'rsk10003', 3, 'low', 'non-smoker, no health issues'), (4, 'rsk10004', 4, 'high', 'previous major surgery'), (5, 'rsk10005', 5, 'medium', 'drives in high traffic areas daily');

3. upload raw csv files to azure data lake (raw_data/bronze)

In Azure Storage Explorer or Azure Portal, create a container (example: destination).

Create folder path: raw_data/bronze/.

Upload CSV files named exactly like your tables: e.g. holders.csv, policies.csv, risk_assessments.csv, etc.



destination
Container

destination

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive) | Only show active objects

Showing all 3 items

Name	Last modified	Access tier	Blob type	Size	Lease state
_$azuretmpfold...	8/8/2025, 5:29:28 pm				
processed_data	8/8/2025, 5:29:28 pm				
raw_data	8/8/2025, 12:48:55 pm				

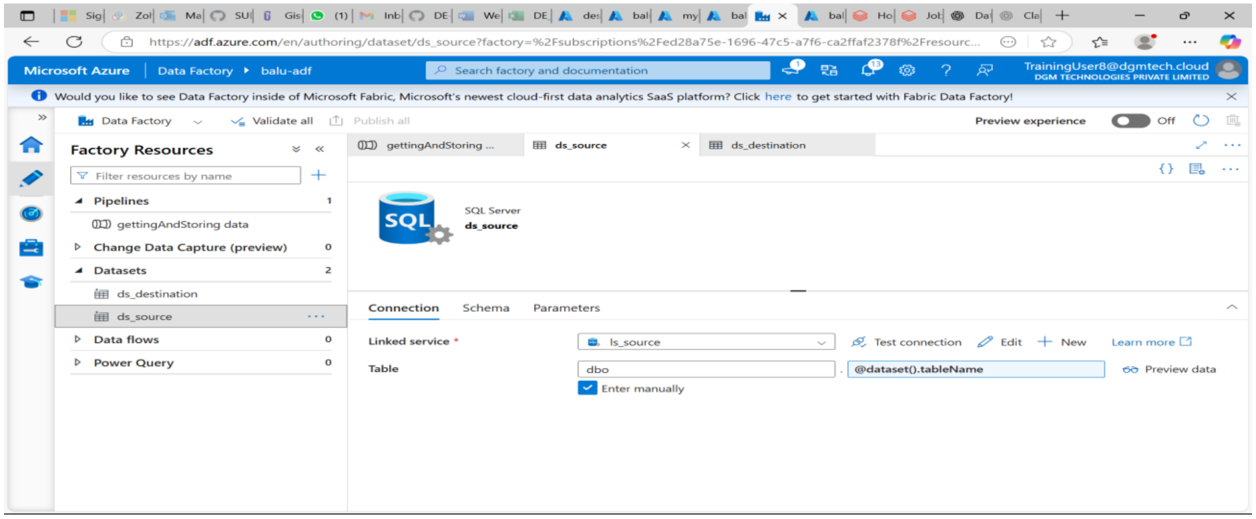
4. build ADF pipeline (lookup → foreach → copy)

4.1 linked services

create linked service to azure SQL.
create linked service to azure data lake storage (use account key or sas if you cannot use key vault).

4.2 datasets

source dataset: add parameter tableName (type string). In table name use @dataset().tableName.

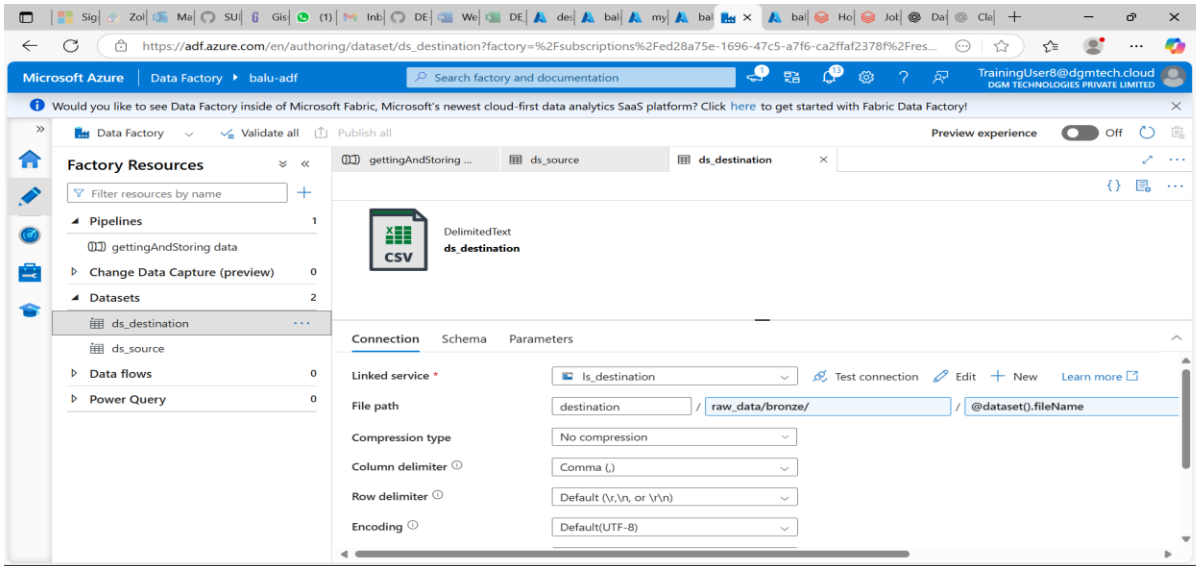


Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!

SQL Server
ds_source

Linked service *	ls_source
Table	dbo . @dataset().tableName
	✓ Enter manually

blob sink dataset (csv): add parameter fileName (type string). In file name use @dataset().fileName.



Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!

Linked service *	ls_destination
File path	destination / raw_data/bronze/ / @dataset().fileName
Compression type	No compression
Column delimiter	Comma (,)
Row delimiter	Default (\r,\n, or \r\n)
Encoding	Default(UTF-8)

4.3 Creating the Lookup Activity in ADF

Added a Lookup activity to the pipeline.

Connected it to my SQL Server Linked Service.

I Wrote a SQL query inside it to return the table names.

SELECT table_name FROM information_schema.tables WHERE table_schema = 'dbo'

The Lookup output is an array of table names.

This output is used later in the pipeline inside a ForEach activity to loop through each table dynamically.

4.4 foreach settings

Set Items: @activity('Lookup'). output.value

Inside ForEach add a Copy Data activity

4.5 copy activity dataset parameter binding

In Copy activity Source dataset parameters: set tableName = @item ().table_name

In Copy activity Sink dataset parameters: set fileName = @concat (item ().table_name, '.csv')

Step 5: Connecting ADF to Databricks Notebook

Purpose:
Some transformations (like Great Expectations validations, masking, aggregation) were done in Databricks notebooks.

What I did:

In ADF, created a Databricks Linked Service:

Choose "Azure Databricks".

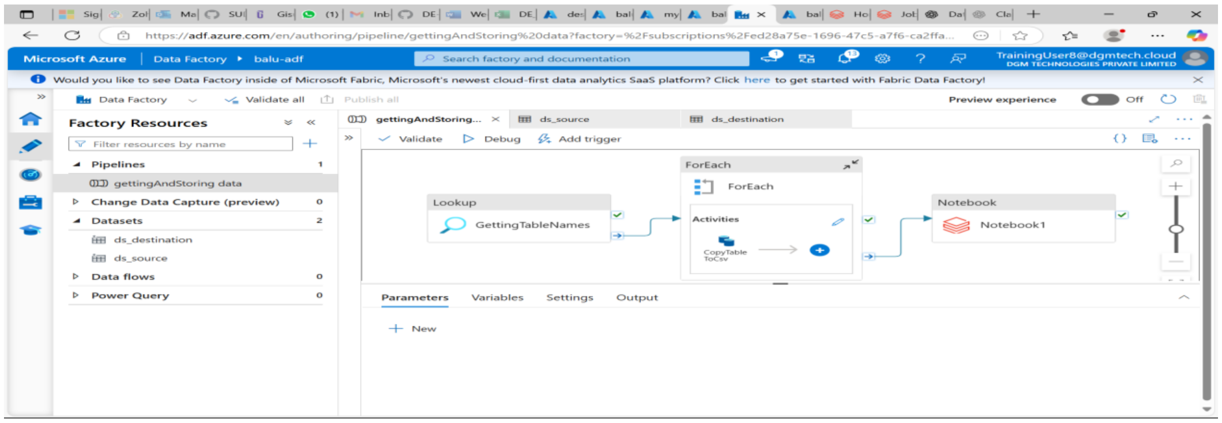
Authentication: Access token (generated in my Databricks workspace → User Settings → Access Tokens).

Selected my Databricks workspace and cluster.

Added a Notebook Activity to the pipeline.

Linked it to my Databricks Linked Service.
Selected my existing notebook file that has my transformation logic.
Added a parameter to the Notebook Activity called container.

Set its value to come from a pipeline parameter container_name.


https://adf.azure.com/en/authoring/pipeline/gettingAndStoring%20data?factory=%2Fsubscriptions%2Fed28a75e-1696-47c5-a7f6-ca2ffa...

Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!

Step 6: Passing the Container Parameter

Purpose:
I needed the notebook to know which storage container it should work on.

How the flow works:

Pipeline parameter container_name → passed into Notebook Activity parameter container → used inside the Databricks notebook.

In ADF Notebook Activity:

Parameter name: container

Value: @pipeline().parameters.container_name

In Databricks notebook:

```
dbutils.widgets.text("container", "")
container = dbutils.widgets.get("container")
print(f"Processing container: {container}")
```

Step 7: Pipeline Flow Summary

Overall execution order:

Lookup Activity → gets table names from SQL Server.

ForEach Activity → loops through each table name.

Inside the loop:

Copy activity: Copies table data from SQL Server to Blob Storage (Bronze layer).

Databricks Notebook Activity: Runs data validation and masking on Bronze data → saves Silver data.

Final Databricks Notebook Activity: Aggregates Silver data into Gold table (holder_id, total_premium_amount).

Gold table is saved in Delta format in azure data lake storage.

Conclusion:

In this project, I successfully built an end-to-end data pipeline starting from extracting data from SQL Server, processing it in Databricks, validating and masking sensitive data using Great Expectations, and then loading the results into bronze, silver, and gold layers in Azure data lake Storage. We connected Azure Data Factory with Databricks using parameters for dynamic data movement, created linked services and datasets for secure and reusable connections, and automated the flow for efficiency. This setup makes the data pipeline flexible, secure, and easy to maintain for future needs.