Fraud Detection Using Azure Data Engineering

Project Overview

This project is designed to detect potentially fraudulent transactions based on the distance between a customer's registered city and the city where the transaction occurred. Fraud criticality is categorized as **Low, Medium, or High** based on the calculated distance. The data pipeline is built using **Azure Data Factory (ADF), Azure Data Lake Storage (ADLS), Azure Databricks, and Azure SQL Database**.

Requirement

Based on the city in which the transaction has happened, the fraud criticality needs to be calculated based on the below conditions,

- **Low:** Distance < 500 km
- **Medium:** Distance 5000-1000 km
- **High:** Distance > 1000 km

The following reports need to be developed based on the Fraud criticality,

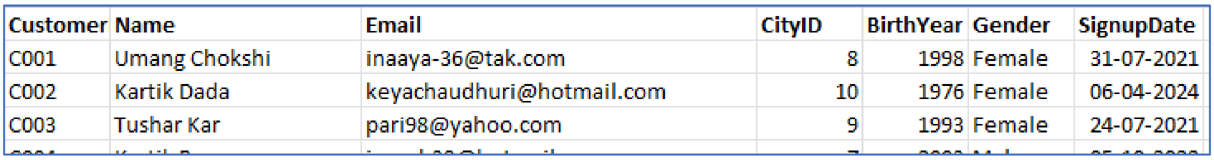
- Bar chart displaying Fraud Alert Counts for past 12 months.
- Pie chart showing Fraud criticality trend
- Top 10 Customers and Cities of High Fraud Criticality
- Pie chart showing Age Group criticality trend

Tech Stack

- **Azure Data Factory (ADF)** – Data ingestion and orchestration.
- **Azure Data Lake Storage (ADLS)** – Raw and landing data storage.
- **Azure Databricks** – Data transformation and fraud analysis.
- **Azure SQL Database** – Storing transformed data for reporting.
- **Power BI** – Visualization and reporting.

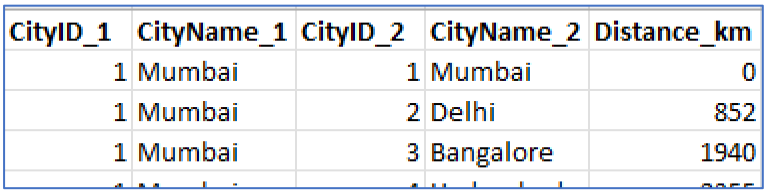
Source Datasets Used

1. **Customer Metadata** – Contains customer details, including city.



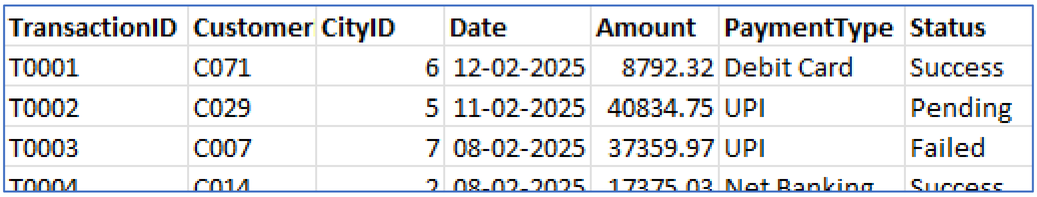
Customer	Name	Email	CityID	BirthYear	Gender	SignupDate
C001	Umang Chokshi	inaaya-36@tak.com	8	1998	Female	31-07-2021
C002	Kartik Dada	keyachaudhuri@hotmail.com	10	1976	Female	06-04-2024
C003	Tushar Kar	pari98@yahoo.com	9	1993	Female	24-07-2021

2. **City Distance Metadata** – Contains distance between every two cities in India.



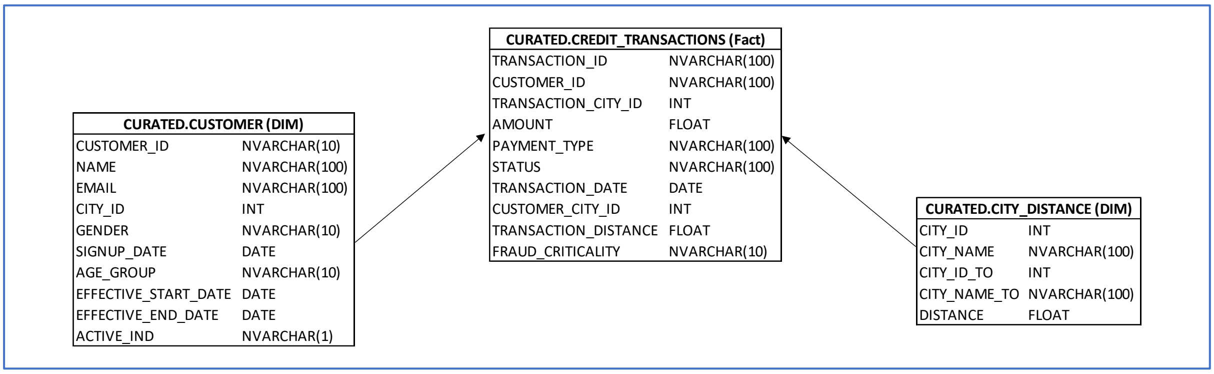
CityID_1	CityName_1	CityID_2	CityName_2	Distance_km
1	Mumbai	1	Mumbai	0
1	Mumbai	2	Delhi	852
1	Mumbai	3	Bangalore	1940

3. **Transaction Data** – Contains transaction details, including transaction city.



TransactionID	Customer	CityID	Date	Amount	PaymentType	Status
T0001	C071	6	12-02-2025	8792.32	Debit Card	Success
T0002	C029	5	11-02-2025	40834.75	UPI	Pending
T0003	C007	7	08-02-2025	37359.97	UPI	Failed
T0004	C014	2	08-02-2025	17375.03	Net Banking	Success

Data Model



CURATED.CUSTOMER (DIM)

CUSTOMER_ID	NVARCHAR(10)
NAME	NVARCHAR(100)
EMAIL	NVARCHAR(100)
CITY_ID	INT
GENDER	NVARCHAR(10)
SIGNUP_DATE	DATE
AGE_GROUP	NVARCHAR(10)
EFFECTIVE_START_DATE	DATE
EFFECTIVE_END_DATE	DATE
ACTIVE_IND	NVARCHAR(1)

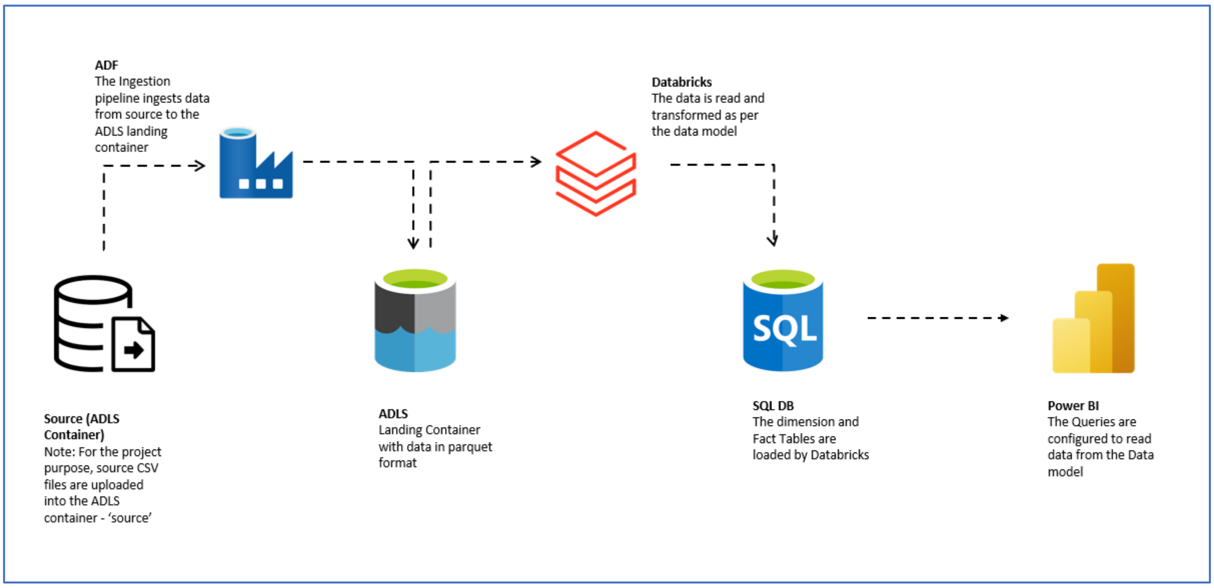
CURATED.CREDIT_TRANSACTIONS (Fact)

TRANSACTION_ID	NVARCHAR(100)
CUSTOMER_ID	NVARCHAR(100)
TRANSACTION_CITY_ID	INT
AMOUNT	FLOAT
PAYMENT_TYPE	NVARCHAR(100)
STATUS	NVARCHAR(100)
TRANSACTION_DATE	DATE
CUSTOMER_CITY_ID	INT
TRANSACTION_DISTANCE	FLOAT
FRAUD_CRITICALITY	NVARCHAR(10)

CURATED.CITY_DISTANCE (DIM)

CITY_ID	INT
CITY_NAME	NVARCHAR(100)
CITY_ID_TO	INT
CITY_NAME_TO	NVARCHAR(100)
DISTANCE	FLOAT

Architecture



ADF
The Ingestion pipeline ingests data from source to the ADLS landing container

Databricks
The data is read and transformed as per the data model

Source (ADLS Container)
Note: For the project purpose, source CSV files are uploaded into the ADLS container - 'source'

ADLS
Landing Container with data in parquet format

SQL DB
The dimension and Fact Tables are loaded by Databricks

Power BI
The Queries are configured to read data from the Data model

Detailed Implementation:

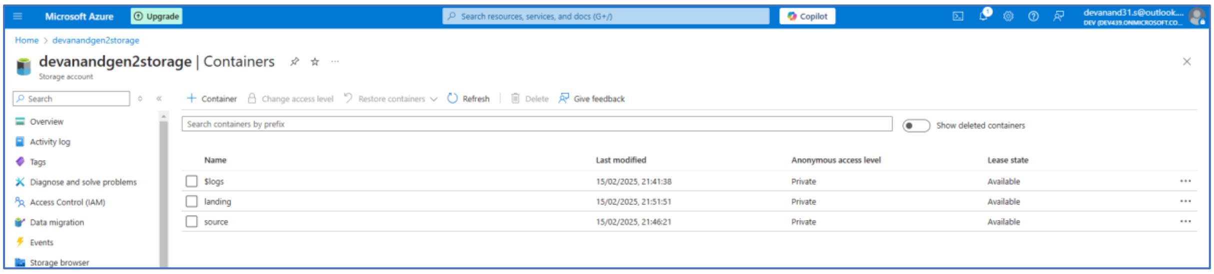
Azure Data Lake Storage:

Two containers are created,

Source – to store the raw csv source files.

Landing – to store the parquet file ingested by ADF

NOTE: This action is implemented to mimic the data ingestion from source to ADLS



devanandgen2storage | Containers

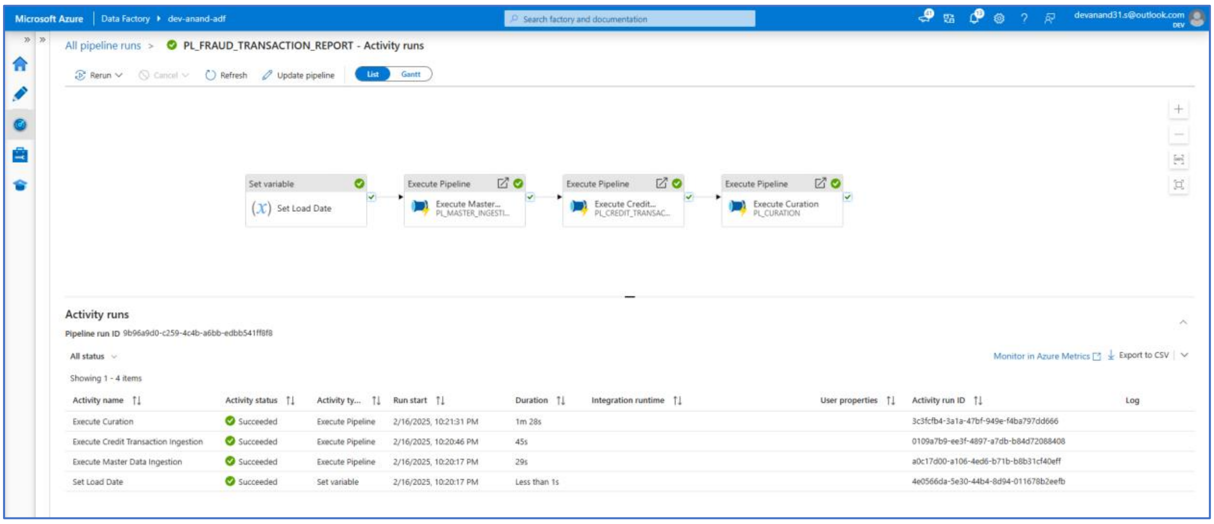
Name	Last modified	Anonymous access level	Lease state
$logs	15/02/2025, 21:41:38	Private	Available
landing	15/02/2025, 21:51:51	Private	Available
source	15/02/2025, 21:46:21	Private	Available

The transaction data files are stored in YYYY/MM/DD date folder format.

Azure Data Factory:

The ADF contains a main pipeline which starts with setting the load date (current date -1 day). It then executes two ingestion pipeline (Master data & Transaction data) and one curation pipeline.

PL_FRAUD_TRANSACTION_REPORT (main pipeline):



PL_FRAUD_TRANSACTION_REPORT - Activity runs

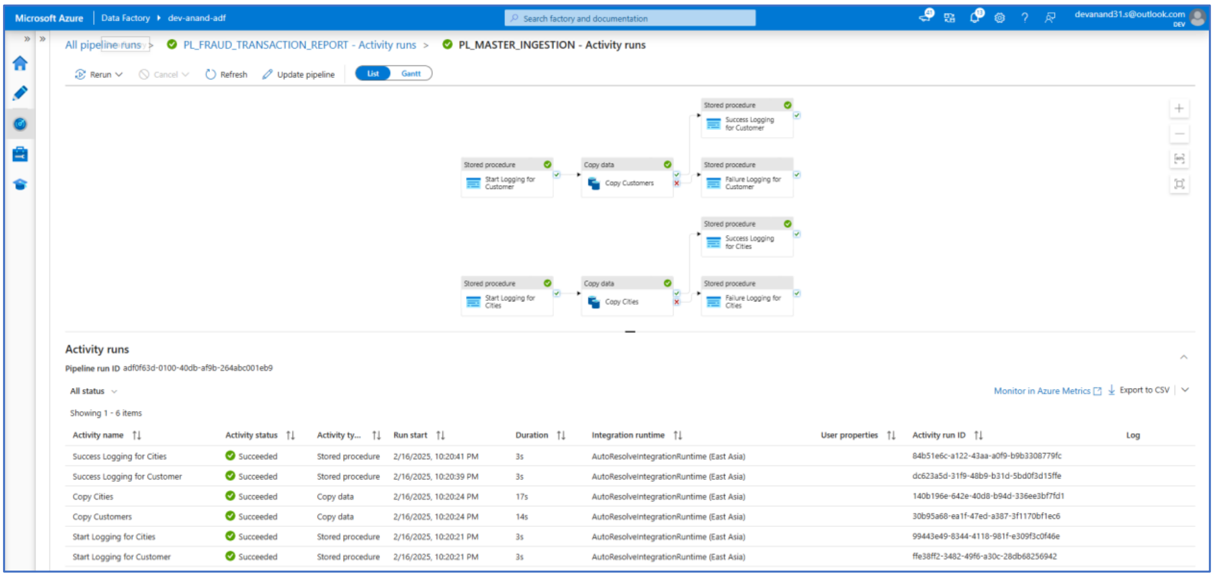
Activity runs

Pipeline run ID 9b96a9d0-c259-4c4b-a6bb-edbb541ff8f8

Showing 1 - 4 items

Activity name	Activity status	Activity ty...	Run start	Duration	Integration runtime	User properties	Activity run ID	Log
Execute Curation	Succeeded	Execute Pipeline	2/16/2025, 10:21:31 PM	1m 28s			3c3fcfb4-3a1a-47bf-949e-f4ba797dd666	
Execute Credit Transaction Ingestion	Succeeded	Execute Pipeline	2/16/2025, 10:20:46 PM	45s			0109a7b9-ee3f-4897-a7db-b84d72088408	
Execute Master Data Ingestion	Succeeded	Execute Pipeline	2/16/2025, 10:20:17 PM	29s			a0c17d00-a106-4ed6-b71b-b8b31cf40eff	
Set Load Date	Succeeded	Set variable	2/16/2025, 10:20:17 PM	Less than 1s			4e0566da-5e30-44b4-8d94-011678b2eefb	

PL_MASTER_INGESTION: Ingests Customer and Cities data into landing container on full load basis



Activity runs

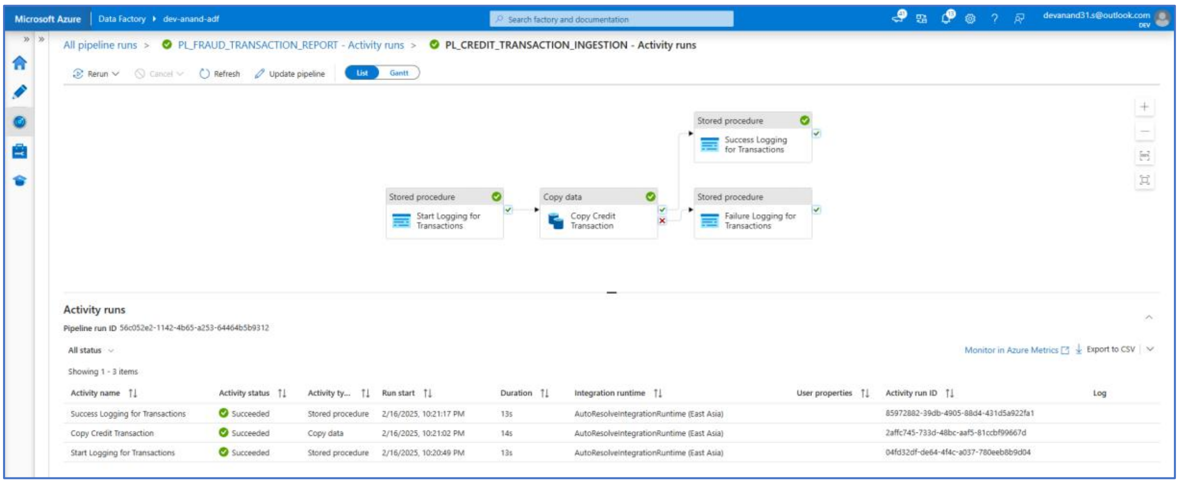
Pipeline run ID adf0f63d-0100-40db-af9b-264abc001eb9

All status

Showing 1 - 6 items

Activity name	Activity status	Activity ty...	Run start	Duration	Integration runtime	User properties	Activity run ID	Log
Success Logging for Cities	✓ Succeeded	Stored procedure	2/16/2025, 10:20:41 PM	3s	AutoResolveIntegrationRuntime (East Asia)		84b51e6c-a122-43aa-a0f9-b9b3308779fc	
Success Logging for Customer	✓ Succeeded	Stored procedure	2/16/2025, 10:20:39 PM	3s	AutoResolveIntegrationRuntime (East Asia)		dc623a5d-31f9-48b9-b31d-5bd0f3d15ffe	
Copy Cities	✓ Succeeded	Copy data	2/16/2025, 10:20:24 PM	17s	AutoResolveIntegrationRuntime (East Asia)		140b196e-642e-40d8-b94d-336ee3bf7fd1	
Copy Customers	✓ Succeeded	Copy data	2/16/2025, 10:20:24 PM	14s	AutoResolveIntegrationRuntime (East Asia)		30b95a68-ea1f-47ed-a387-3f1170bf1ec6	
Start Logging for Cities	✓ Succeeded	Stored procedure	2/16/2025, 10:20:21 PM	3s	AutoResolveIntegrationRuntime (East Asia)		99443e49-8344-4118-981f-e309f3c0f46e	
Start Logging for Customer	✓ Succeeded	Stored procedure	2/16/2025, 10:20:21 PM	3s	AutoResolveIntegrationRuntime (East Asia)		ffe38ff2-3482-49f6-a30c-28db68256942	

PL_CREDIT_TRANSACTION_INGESTION: For ingesting the transaction data on an incremental basis



Activity runs

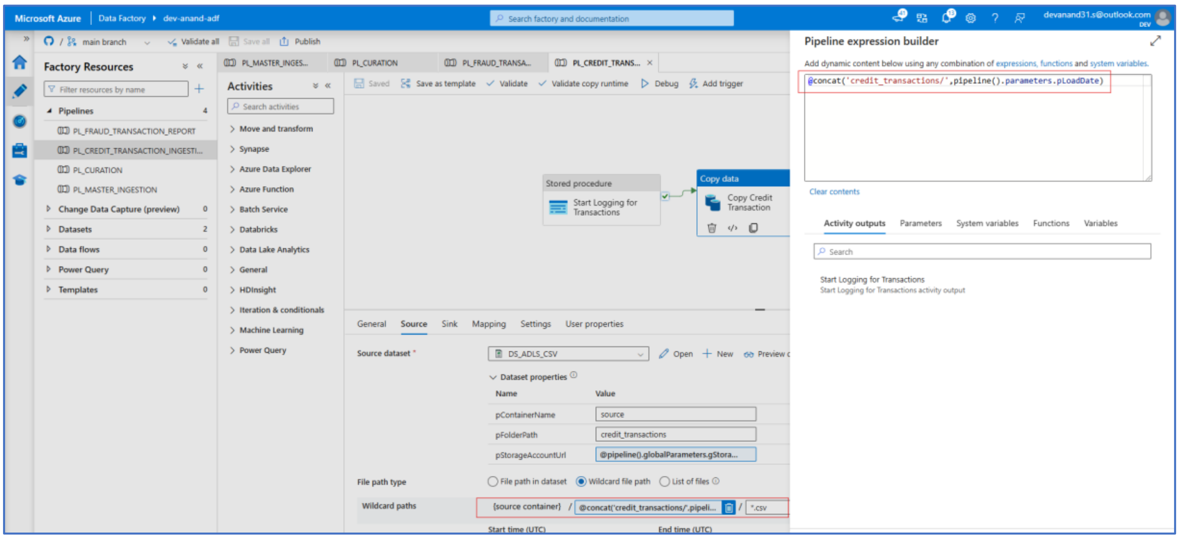
Pipeline run ID 56c052e2-1142-4b65-a253-64464b5b9312

All status

Showing 1 - 3 items

Activity name	Activity status	Activity ty...	Run start	Duration	Integration runtime	User properties	Activity run ID	Log
Success Logging for Transactions	✓ Succeeded	Stored procedure	2/16/2025, 10:21:17 PM	13s	AutoResolveIntegrationRuntime (East Asia)		85972882-39db-4905-88d4-431d5a922fa1	
Copy Credit Transaction	✓ Succeeded	Copy data	2/16/2025, 10:21:02 PM	14s	AutoResolveIntegrationRuntime (East Asia)		2affc745-733d-48bc-aaf5-81ccbf99667d	
Start Logging for Transactions	✓ Succeeded	Stored procedure	2/16/2025, 10:20:49 PM	13s	AutoResolveIntegrationRuntime (East Asia)		04fd32df-de64-4f4c-a037-780eeb8b9d04	

Incremental logic: The Load date is passed as parameter from the main pipeline which is used in the wild card path



Pipeline expression builder

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

```
@concat('credit_transactions/',pipeline().parameters.pLoadDate)
```

Clear contents

Source dataset * DS_ADLS_CSV

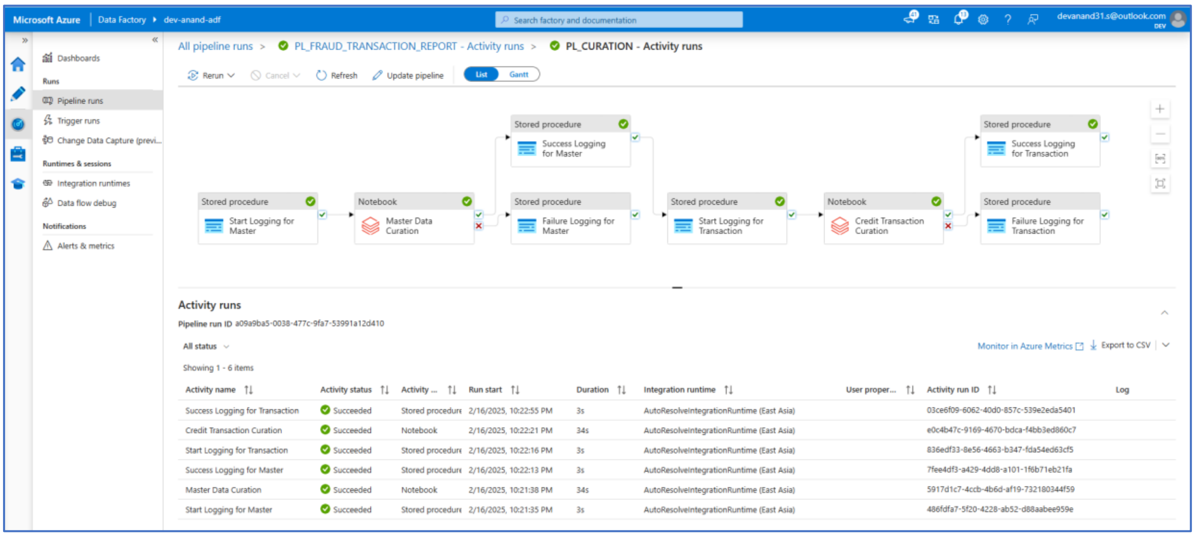
Dataset properties

Name	Value
pContainerName	source
pFolderPath	credit_transactions
pStorageAccountUrl	@pipeline().globalParameters.gStora...

File path type ○ File path in dataset ● Wildcard file path ○ List of files

Wildcard paths {source container} / @concat('credit_transactions/',pipeli... / *.csv

PL_CURATION: The curation pipeline contains two notebook activity. One for executing the master curation and another on for executing the transaction curation



Activity runs

Pipeline run ID a09a9ba5-0038-477c-9fa7-53991a12d410

All status

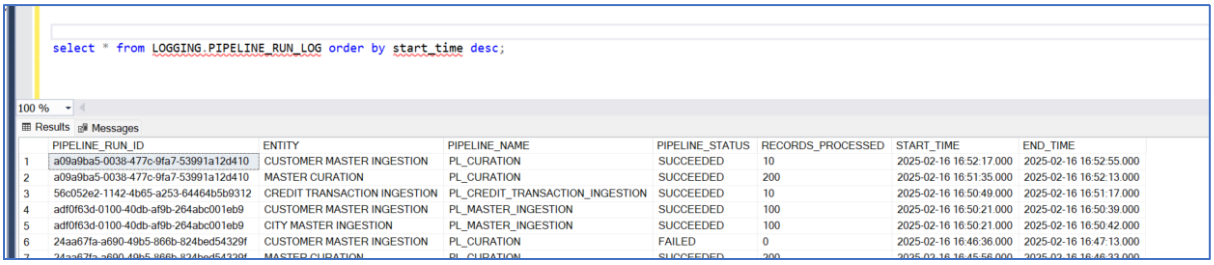
Monitor in Azure Metrics | Export to CSV

Showing 1 - 6 items

Activity name	Activity status	Activity ...	Run start	Duration	Integration runtime	User proper...	Activity run ID	Log
Success Logging for Transaction	✓ Succeeded	Stored procedure	2/16/2025, 10:22:55 PM	3s	AutoResolveIntegrationRuntime (East Asia)		03ce6f09-6062-40d0-857c-539e2eda5401	
Credit Transaction Curation	✓ Succeeded	Notebook	2/16/2025, 10:22:21 PM	34s	AutoResolveIntegrationRuntime (East Asia)		e0c4b47c-9169-4670-bdca-f4bb3ed860c7	
Start Logging for Transaction	✓ Succeeded	Stored procedure	2/16/2025, 10:22:16 PM	3s	AutoResolveIntegrationRuntime (East Asia)		836edf33-8e56-4663-b347-fda54ed63cf5	
Success Logging for Master	✓ Succeeded	Stored procedure	2/16/2025, 10:22:13 PM	3s	AutoResolveIntegrationRuntime (East Asia)		7fee4df3-a429-4dd8-a101-1f6b71eb21fa	
Master Data Curation	✓ Succeeded	Notebook	2/16/2025, 10:21:38 PM	34s	AutoResolveIntegrationRuntime (East Asia)		5917d1c7-4ccb-4b6d-af19-732180344f59	
Start Logging for Master	✓ Succeeded	Stored procedure	2/16/2025, 10:21:35 PM	3s	AutoResolveIntegrationRuntime (East Asia)		486fdfa7-5f20-4228-ab52-d88aabee959e	

All pipelines are implemented with logging logic in which the stored procedure is called. This stored procedure INSERTs and UPDATEs the necessary logs at the appropriate steps.

Logging Table:



```
select * from LOGGING.PIPELINE_RUN_LOG order by start_time desc;
```

	PIPELINE_RUN_ID	ENTITY	PIPELINE_NAME	PIPELINE_STATUS	RECORDS_PROCESSED	START_TIME	END_TIME
1	a09a9ba5-0038-477c-9fa7-53991a12d410	CUSTOMER MASTER INGESTION	PL_CURATION	SUCCEEDED	10	2025-02-16 16:52:17.000	2025-02-16 16:52:55.000
2	a09a9ba5-0038-477c-9fa7-53991a12d410	MASTER CURATION	PL_CURATION	SUCCEEDED	200	2025-02-16 16:51:35.000	2025-02-16 16:52:13.000
3	56c052e2-1142-4b65-a253-64464b5b9312	CREDIT TRANSACTION INGESTION	PL_CREDIT_TRANSACTION_INGESTION	SUCCEEDED	10	2025-02-16 16:50:49.000	2025-02-16 16:51:17.000
4	adf0f63d-0100-40db-af9b-264abc001eb9	CUSTOMER MASTER INGESTION	PL_MASTER_INGESTION	SUCCEEDED	100	2025-02-16 16:50:21.000	2025-02-16 16:50:39.000
5	adf0f63d-0100-40db-af9b-264abc001eb9	CITY MASTER INGESTION	PL_MASTER_INGESTION	SUCCEEDED	100	2025-02-16 16:50:21.000	2025-02-16 16:50:42.000
6	24aa67fa-a690-49b5-866b-824bed54329f	CUSTOMER MASTER INGESTION	PL_CURATION	FAILED	0	2025-02-16 16:46:36.000	2025-02-16 16:47:13.000
7	24aa67fa-a690-49b5-866b-824bed54329f	MASTER CURATION	PL_CURATION	SUCCEEDED	200	2025-02-16 16:45:56.000	2025-02-16 16:46:33.000

Databricks:

Connection and Helper Function:

- This notebook contains the necessary properties declared for establishing connections from Databricks notebook to ADLS and SQL DB.
- This notebook also contains the helped function that will accept the query string. This query string is executed in the SQL DB through ODBC Driver 17 for SQL Server.

Master Table Curation

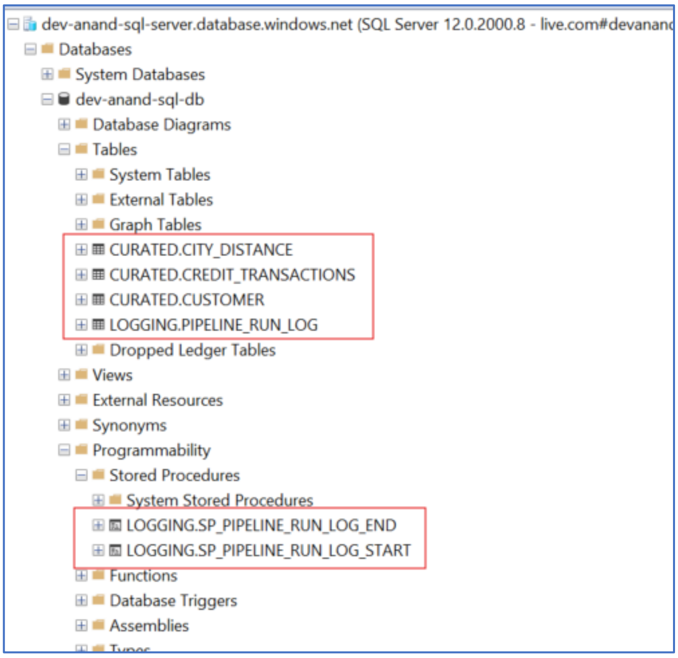
- The customers and cities files from landing container are read, processed and loaded to the CURATED.CUSTOMER and CURATED.CITY_DISTANCE tables.
- The cities data are overwritten every time the notebook is executed.
- From the customer data, the age group of each customer is calculated. Every time when the notebook is executed, the data is compared against existing data in the table and instead of rewriting the updated data, the old data is marked as Inactive and the new updated data is inserted as Active. This helps in maintaining the customer history (SCD2) for future analysis.

Transaction Curation:

- This notebook reads the incremental transactions on a (current date – 1 day) basis.
- Using the customer and city curated data, the distance between the customer city and the city where the transaction has happened is calculated. Based on this distance, the criticality for fraud is calculated.
    - **Low:** Distance < 500 km
    - **Medium:** Distance 5000-1000 km
    - **High:** Distance > 1000 km

SQL Database:

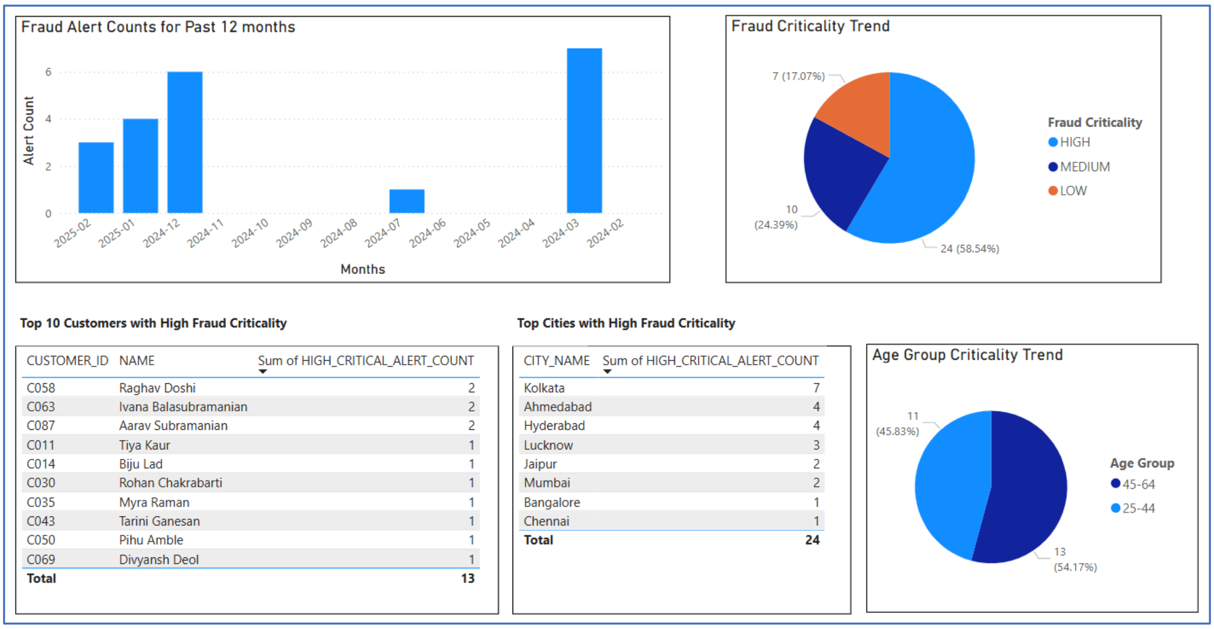
- CURATED.CREDIT_TRANSACTIONS – This is the fact tables loaded with transaction containing the calculated fraud criticality column. It gets loaded on an incremental basis.
- CURATED.CUSTOMER – Contains customer master data. Follows SCD2.
- CURATED.CITY_DISTANCE – Contains city master data and is loaded on full load basis.
- LOGGING.SP_PIPELINE_RUN_LOG_START, LOGGING.SP_PIPELINE_RUN_LOG_END – Stored Procedures used to INSERT and UPDATE logs respectively
- LOGGING.PIPELINE_RUN_LOG – Contains pipeline execution logs



```
⊟ dev-anand-sql-server.database.windows.net (SQL Server 12.0.2000.8 - live.com#devananc
  ⊟ Databases
    ⊞ System Databases
    ⊟ dev-anand-sql-db
      ⊞ Database Diagrams
      ⊟ Tables
        ⊞ System Tables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊞ CURATED.CITY_DISTANCE
        ⊞ CURATED.CREDIT_TRANSACTIONS
        ⊞ CURATED.CUSTOMER
        ⊞ LOGGING.PIPELINE_RUN_LOG
        ⊞ Dropped Ledger Tables
      ⊞ Views
      ⊞ External Resources
      ⊞ Synonyms
      ⊟ Programmability
        ⊟ Stored Procedures
          ⊞ System Stored Procedures
          ⊞ LOGGING.SP_PIPELINE_RUN_LOG_END
          ⊞ LOGGING.SP_PIPELINE_RUN_LOG_START
        ⊞ Functions
        ⊞ Database Triggers
        ⊞ Assemblies
        ⊞ Types
```

Power BI:

Finally, in the Power BI, 5 different queries are created to generate reports as per the requirement.



Top 10 Customers with High Fraud Criticality

CUSTOMER_ID	NAME	Sum of HIGH_CRITICAL_ALERT_COUNT
C058	Raghav Doshi	2
C063	Ivana Balasubramanian	2
C087	Aarav Subramanian	2
C011	Tiya Kaur	1
C014	Biju Lad	1
C030	Rohan Chakrabarti	1
C035	Myra Raman	1
C043	Tarini Ganesan	1
C050	Pihu Amble	1
C069	Divyansh Deol	1
Total		**13**

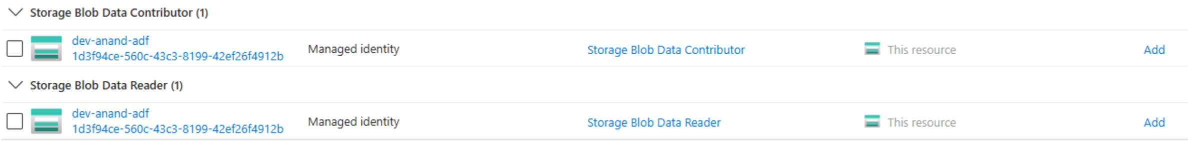
Top Cities with High Fraud Criticality

CITY_NAME	Sum of HIGH_CRITICAL_ALERT_COUNT
Kolkata	7
Ahmedabad	4
Hyderabad	4
Lucknow	3
Jaipur	2
Mumbai	2
Bangalore	1
Chennai	1
Total	**24**

Security:

ADF <-> ADLS:

The Connection for the ADF to read and write files from and to ADLS is performed using managed identity with below access roles.



Storage Blob Data Contributor (1)

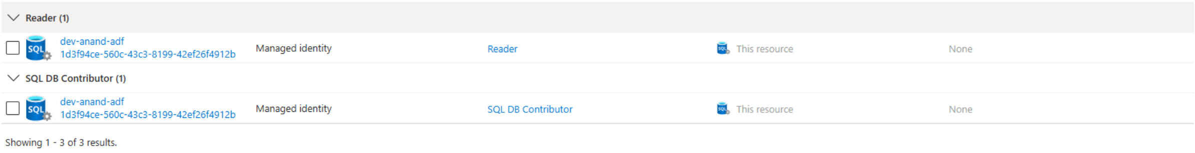
	Name	Type	Role	Scope	Condition
□	dev-anand-adf 1d3f94ce-560c-43c3-8199-42ef26f4912b	Managed identity	Storage Blob Data Contributor	This resource	Add

Storage Blob Data Reader (1)

	Name	Type	Role	Scope	Condition
□	dev-anand-adf 1d3f94ce-560c-43c3-8199-42ef26f4912b	Managed identity	Storage Blob Data Reader	This resource	Add

ADF <-> SQL DB:

The Connection for the ADF to read and write files from and to SQL DB is performed using managed identity with below access roles.



Reader (1)

	Name	Type	Role	Scope	Condition
□	dev-anand-adf 1d3f94ce-560c-43c3-8199-42ef26f4912b	Managed identity	Reader	This resource	None

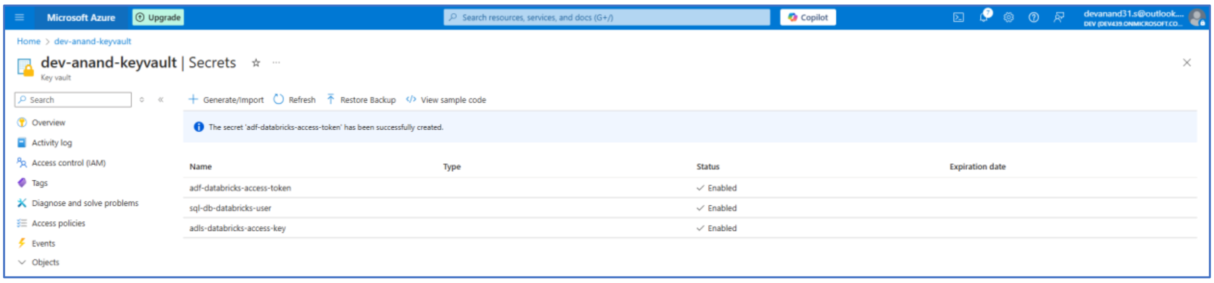
SQL DB Contributor (1)

	Name	Type	Role	Scope	Condition
□	dev-anand-adf 1d3f94ce-560c-43c3-8199-42ef26f4912b	Managed identity	SQL DB Contributor	This resource	None

Showing 1 - 3 of 3 results.

Databricks with ADF, ADLS and SQL DB:

Since this is a personal account, only the Standard edition of Databricks workspace is created. This version of Databricks does not allow Managed identity. So, all the Databricks related access are established used access tokens/access keys via Key Vault.



dev-anand-keyvault | Secrets

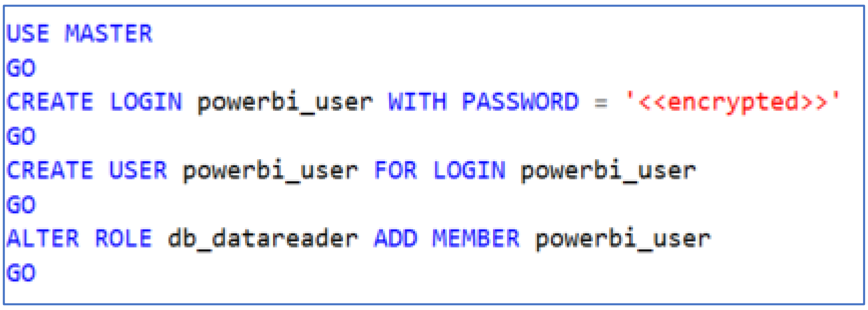
Key vault

The secret 'adf-databricks-access-token' has been successfully created.

Name	Type	Status	Expiration date
adf-databricks-access-token		✓ Enabled	
sql-db-databricks-user		✓ Enabled	
adls-databricks-access-key		✓ Enabled	

Power BI <-> SQL DB

A SQL user – powerbi_user is created for the Power BI to access SQL DB.



```
USE MASTER
GO
CREATE LOGIN powerbi_user WITH PASSWORD = '<<encrypted>>'
GO
CREATE USER powerbi_user FOR LOGIN powerbi_user
GO
ALTER ROLE db_datareader ADD MEMBER powerbi_user
GO
```