# Bootcamp Project 4

Project Title: Incremental Data Loading and Automated Notifications using Microsoft Fabric

Problem Statement:

In modern data ecosystems, organizations need to efficiently ingest, transform, and load data from various sources into centralized platforms for analytics, while also ensuring timely monitoring and notification upon successful data refreshes. This project addresses the challenge of incrementally loading data from on-premises sources to Microsoft Fabric Lakehouse, processing it through a structured transformation pipeline, and triggering automated notifications upon successful execution.

Architecture diagram:
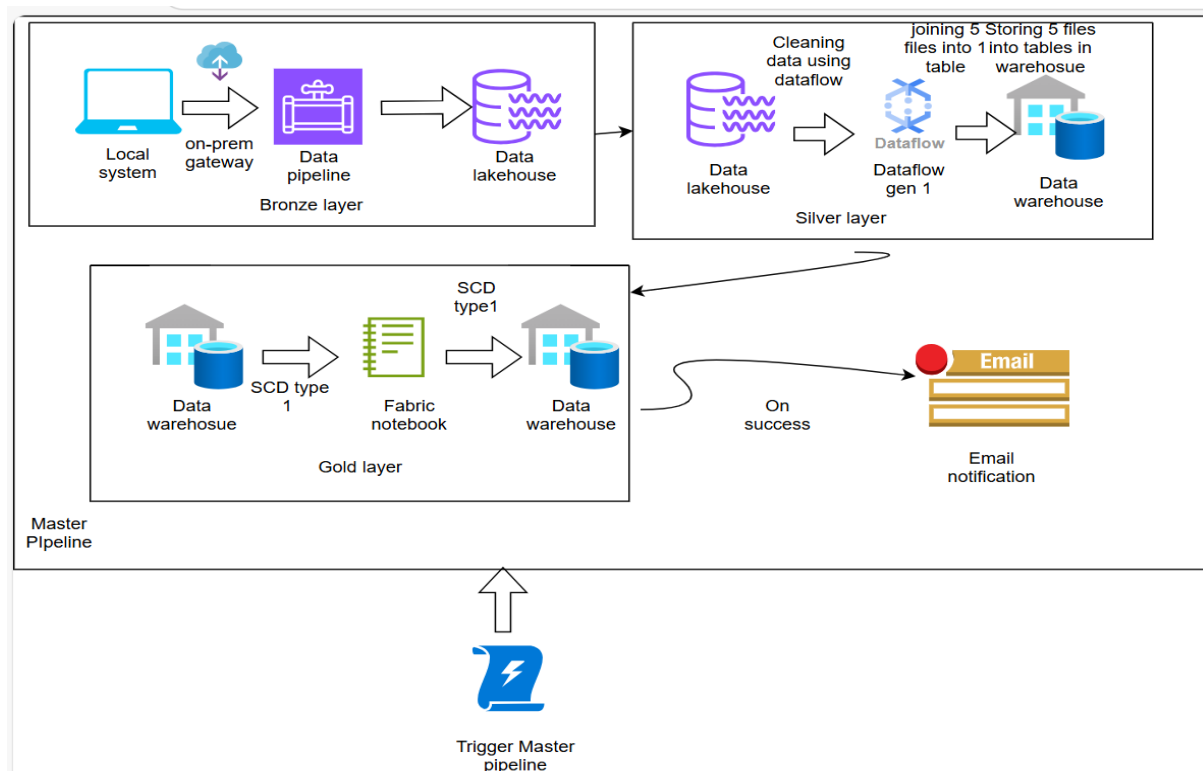
Tools & Technologies:
• Microsoft Fabric
• On-Premises Data Gateway
• Fabric Lakehouse and Warehouse
• Fabric Dataflow Gen 1
• Fabric Notebook
• Email Notification Task (in-built)
• Draw.io / Visio for architecture diagram

Bronze layer:

To connect on-prem to fabric pipeline we need to download and install ON-prem Gateway.

On-premises data gateway installation

Getting ready to install the on-premises data gateway.

Please review minimum requirements for installing the On-premises data gateway. Not meeting these requirements may result in performance bottlenecks.

Install to

C:\Program Files\On-premises data gateway

☑ I accept the terms of use and privacy statement .

Install   Close

On-premises data gateway installation

Installing your on-premises data gateway.

Please wait

Cancel

On-premises data gateway

Almost done.

⊗ A valid email address must be specified (e.g.: abc@example.com).

Installation was successful!

Email address to use with this gateway *

fabricnisha@nisha1207reddygmail.onmicrosoft.com

Next, you need to sign in to register your gateway.

Sign in options

☑ Use my default web browser

Export logs   Sign in   Cancel

On-premises data gateway

? ✕

You are signed in as fabricnisha@nisha1207reddygmail.onmicrosoft.com and are ready to register the gateway.

◉ Register a new gateway on this computer.

○ Migrate, restore, or takeover an existing gateway.
  • Move a gateway to a new computer
  • Recover a damaged gateway
  • Take ownership of a gateway
  The old gateway will be disconnected.

[Next]   [Cancel]

On-premises data gateway

? ✕

You are signed in as fabricnisha@nisha1207reddygmail.onmicrosoft.com and are ready to register the gateway.

New on-premises data gateway name *

Deepthi

☐ Add to an existing gateway cluster   Learn more

Recovery key (8 character minimum) *

••••••••••••

ⓘ This key is needed to restore the gateway and can't be changed. Record it in a safe place.

Confirm recovery key *

••••••••••••

We'll use this region to connect the gateway to cloud services: Canada Central Change Region
Provide relay details (optional) By default, Azure Relays are automatically provisioned

[Configure]   [Cancel]

On-premises data gateway

? ✕

Status	✓ The gateway Deepthi is online and ready to be used.
Service Settings	Gateway version number: 3000.266.4 (April 2025)
Diagnostics	☑ Help us improve the on-premises data gateway by sending usage information to Microsoft.
Network	Read the privacy statement online
Connectors	
Recovery Keys	

Logic Apps, Azure Analysis Services Canada Central	Create a gateway in Azure
Power Apps, Power Automate Canada Central	✓ Ready
Microsoft Fabric Default environment	✓ Ready

[Close]

Now we have to create a pipeline with copy data activity to copy data from on-prem to fabric lake house.

First, we will add foreach activity to get all files from on-prem.
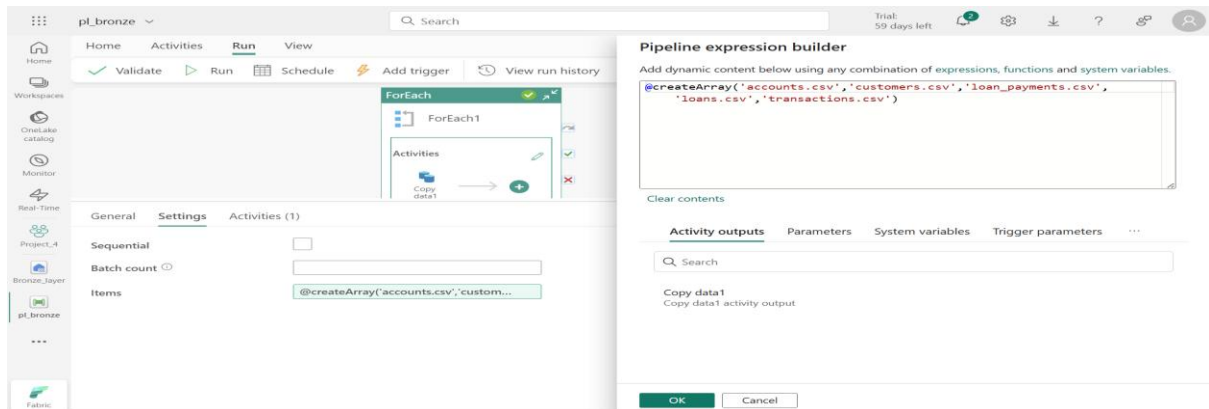
Now add copy activity and connect to on prem gateway.
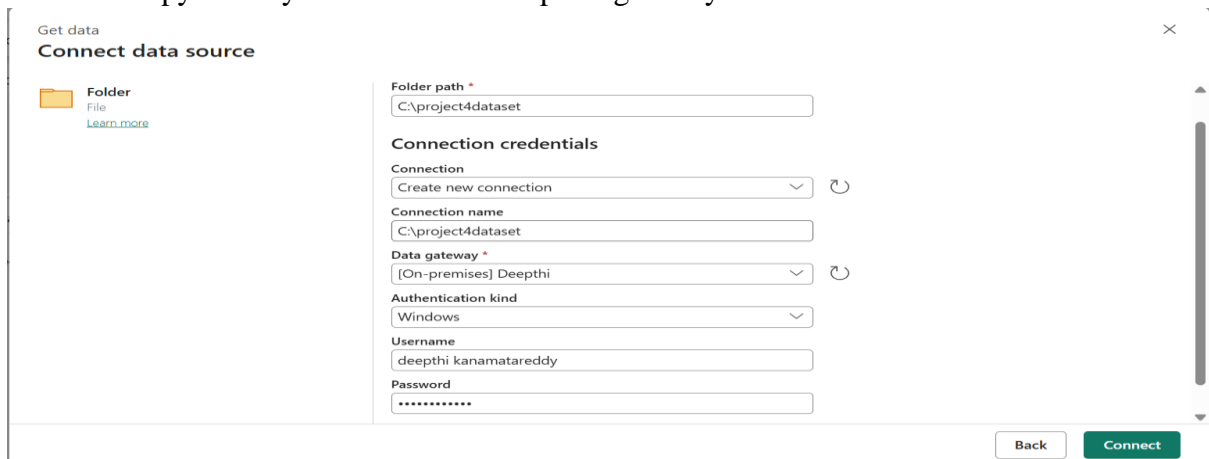
Source:
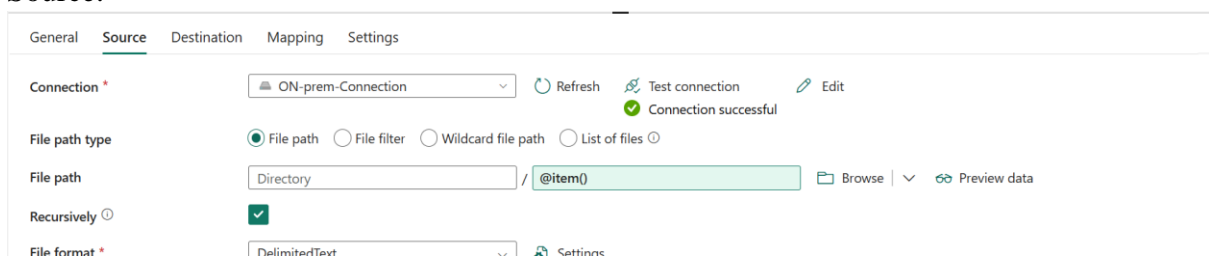
Destination, here we are connecting to lake house and storing data in tables.
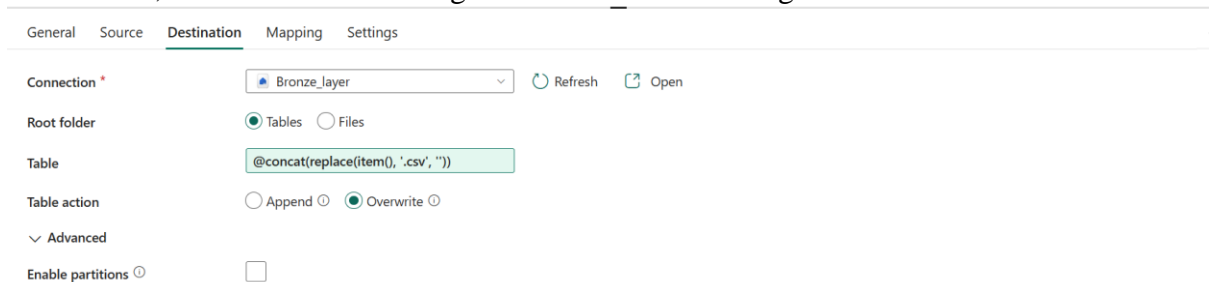
Pipeline ran successfully.

Data is loaded successfully into lake house:

Silver Layer:

Using Dataflow to clean data (Removing duplicates, correcting data types, Replacing nulls).
Connecting source as lake house.
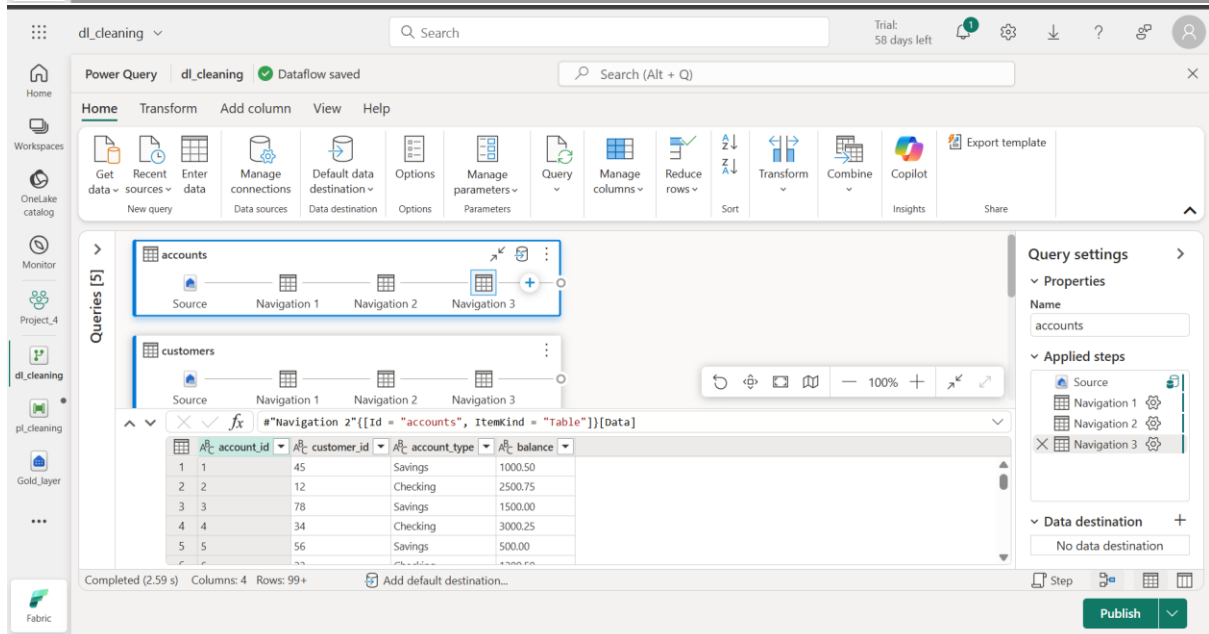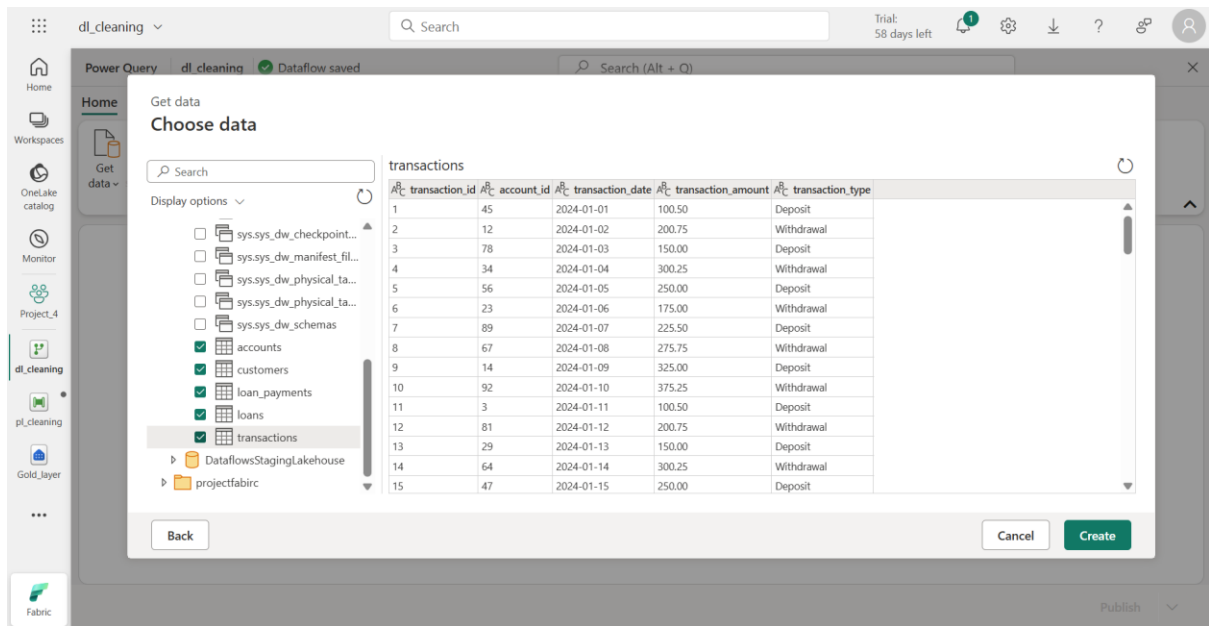
Data type changed

Replacing nulls

Remove duplicates

Adding destination:

All changes applied to all files.

Tables created in warehouse.

Notebook:

creating short cut
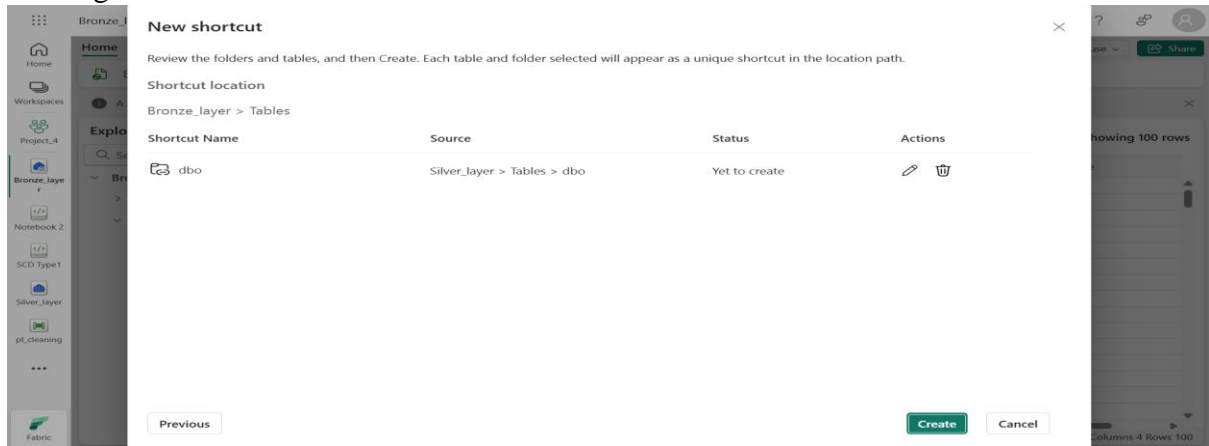
Code:
```
df_source = spark.read.format("delta").load("Tables/dbo/accounts")
df_source.createOrReplaceTempView("source_view")
src_path="Tables/dbo/accounts"

create_table_query = """
CREATE TABLE Account_SCD (
    account_id int,
    customer_id int,
    account_type string,
    balance float,
    hash_key BIGINT,
    created_by STRING,
    created_date TIMESTAMP,
    updated_by STRING,
    updated_date TIMESTAMP
)
USING DELTA
LOCATION 'Tables/Gold_layer/Account_SCD'
```

```
"""
# Execute the query to create the table
spark.sql(create_table_query)

from delta.tables import DeltaTable
target_path = "Tables/Gold_layer/Account_SCD"
delta_target = DeltaTable.forPath(spark, target_path)

from pyspark.sql.functions import *
df_src1= df_source.withColumn("hash_key",crc32(concat(*df_source.columns)))
display(df_src1)

df_src1=df_src1.alias("src").join(delta_target.toDF().alias("tgt"),((col("src.account_id")==col
("tgt.account_id"))&(col("src.hash_key")==col("tgt.hash_key"))),"anti").select(col("src.*"))
df_src1.show()

from pyspark.sql.functions import col
delta_target.alias("tgt").merge(df_src1.alias("src"),"tgt.account_id = src.account_id")\
.whenMatchedUpdate(set={"tgt.account_id":"src.account_id","tgt.customer_id":"src.custome
r_id","tgt.account_type":"src.account_type","tgt.balance":"src.balance","tgt.hash_key":"src.h
ash_key","tgt.updated_date":current_timestamp(),"tgt.updated_by":lit("databricks_Updated")
})\
.whenNotMatchedInsert(values={"tgt.account_id":"src.account_id","tgt.customer_id":"src.cu
stomer_id","tgt.account_type":"src.account_type","tgt.balance":"src.balance","tgt.hash_key":
"src.hash_key","tgt.created_date":current_timestamp(),"tgt.created_by":lit("databricks"),"tgt.
updated_date":current_timestamp(),"tgt.updated_by":lit("databricks")}).execute()

display(spark.read.format("delta").option("header","true").load(target_path))
```

Output:

Accounts output:

Customers Output:

Loan_payment output:

Loans output:

Transactions Output: