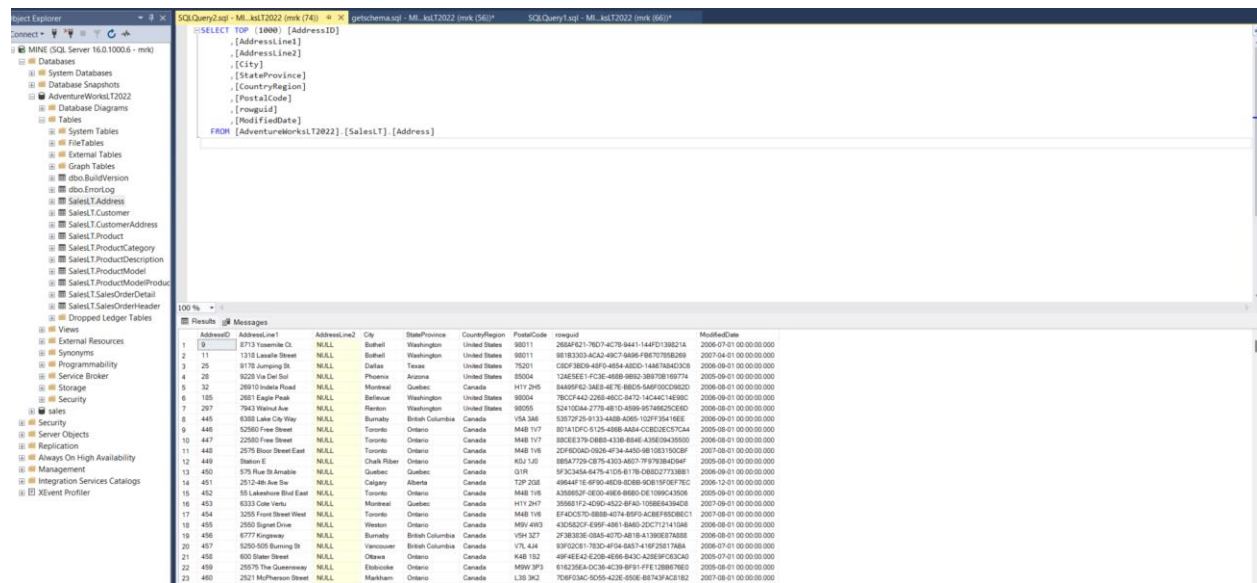


Case Study: Data Pipeline for SQL to Azure

This document describes the step-by-step implementation of a data pipeline to extract data from an on-premises Microsoft SQL database, transfer it to Azure Data Lake Storage (ADLS) using Azure Data Factory (ADF), process and clean the data using Databricks, and finally load the processed data into a "gold" storage layer.

Project Workflow Overview

1. **Data Extraction**: Extract data from the on-premises Microsoft SQL database.



The screenshot shows a SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays a tree view of a database named 'AdventureWorksLT2022'. The 'Tables' folder is expanded, showing various tables like 'SalesLT.Address', 'SalesLT.Customer', etc. The central 'Query Window' displays a SQL query:

```
SELECT TOP (1000) [AddressID], [AddressLine1], [AddressLine2], [City], [StateProvince], [CountryRegion], [PostalCode], [rowguid], [ModifiedDate] FROM [AdventureWorksLT2022].[SalesLT].[Address]
```

 The bottom pane shows the 'Results' grid, which contains 1000 rows of data. The columns are: AddressID, AddressLine1, AddressLine2, City, StateProvince, CountryRegion, PostalCode, rowguid, and ModifiedDate. The data represents a list of addresses from the AdventureWorksLT2022 database.

AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
1	1713 Yosemite Ct.	NULL	Bellevue	Washington	United States	98011	2684F421-76D7-4C7B-9441-144FD13921A	2008-07-01 00:00:00
2	11	1318 Louella Street	NULL	Bellevue	Washington	98011	98183303-AC42-48C7-9468-F8E70705B269	2007-04-01 00:00:00
3	25	8178 Jumping St.	NULL	Dallas	Texas	75201	C3DF3826-48F0-4854-A8C0-144678403C08	2008-09-01 00:00:00
4	26	9229 Via Del Sol	NULL	Phoenix	Arizona	85054	1242EE37-FC2E-488B-9B92-38B708769774	2005-09-01 00:00:00
5	32	26010 Invidia Road	NULL	Montreal	Quebec	H1T 2H5	8489F62-34E8-4E7E-8B05-5AF0CC0C62D	2008-08-01 00:00:00
6	185	2881 Eagle Peak	NULL	Bellevue	Washington	98004	78C0F442-2298-48C0-8472-14C44C14E98C	2008-09-01 00:00:00
7	297	7943 Walnut Ave	NULL	Reston	Washington	98055	52V10344-3778-4B1D-4589-957A625C8ED	2008-08-01 00:00:00
8	445	6388 Lake City Way	NULL	Burnaby	British Columbia	V5A 3A8	53E72F25-9133-4A48-A095-120FF35A16E5	2008-09-01 00:00:00
9	448	52960 Free Street	NULL	Toronto	Ontario	M4B 1Y7	8D141D4C-5129-4888-A8A4-C2B028C27C34	2008-08-01 00:00:00
10	447	22860 Free Street	NULL	Toronto	Ontario	M4B 1Y7	8DCE379E-088B-4338-B84E-A35610A35585	2008-08-01 00:00:00
11	448	2578 Blue Street East	NULL	Toronto	Ontario	M4B 1Y8	2CF8C94D-9926-4F34-4450-98108150C3F	2007-08-01 00:00:00
12	449	Station E	NULL	Chalk River	Ontario	K5J 1J5	88547729-C878-4303-A8D7-7F79384C046F	2008-08-01 00:00:00
13	450	576 Rue St Antoine	NULL	Quebec	Quebec	G1R	8F3C3454-6478-4108-B178-088E07733881	2008-09-01 00:00:00
14	451	2512-4th Ave Sw	NULL	Calgary	Alberta	T2P 2G8	49644F1E-6F90-48D9-8D8B-90B18F0E678C	2008-12-01 00:00:00
15	452	85 Lakeshore Blvd East	NULL	Toronto	Ontario	M4B 1Y8	A398562F-9E00-48E8-B88D-CE1099C43908	2008-09-01 00:00:00
16	453	6333 Cook Street	NULL	Montreal	Quebec	H1T 2H7	398681F2-429D-452D-8F4D-1088E6A38408	2007-09-01 00:00:00
17	454	3258 Frost Street West	NULL	Toronto	Ontario	M4B 1Y8	EF4DCD70-888B-4074-89F0-AC8EF6808EC1	2007-08-01 00:00:00
18	455	2550 Sigurd Drive	NULL	Weston	Ontario	M9J 4H3	43C582CF-439F-4891-B84D-25C71741586	2008-08-01 00:00:00
19	456	6777 Kingsway	NULL	Burnaby	British Columbia	V5N 3Z7	27863E3E-03A5-4C7D-4878-A1768E876388	2008-08-01 00:00:00
20	457	5250-505 Burnside Dr	NULL	Vancouver	British Columbia	V7L 4J4	93F02C81-783D-4F04-8A57-416F2581788A	2008-07-01 00:00:00
21	458	600 State Street	NULL	Ottawa	Ontario	K4B 1B2	49F4EE4D-E208-4E56-843C-A2B8F0D83CA0	2005-07-01 00:00:00
22	459	28078 The Queensway	NULL	Edmonton	Ontario	M6W 3P3	616276EA-CC38-4C28-BF51-F1E1288E7853	2008-08-01 00:00:00
23	460	2521 Mulpherson Street	NULL	Markham	Ontario	L3R 3A2	70BF03AC-8C55-422E-850E-B8743AC8182	2007-08-01 00:00:00

2. **Data Transfer to Raw Zone**: Use Azure Data Factory to move the raw data to the raw folder in Azure Data Lake Storage.

The screenshot shows the Azure Data Factory (ADF) interface. On the left, the 'Factory Resources' pane lists various resources including Pipelines, Datasets, and Data flows. The main canvas displays a pipeline named 'pipeline1' with a 'Lookup' activity (labeled 'look for all tables') and a 'ForEach' loop containing a 'copy each table' activity. The pipeline status is 'Succeeded'. Below the canvas, the 'Output' tab shows a table of pipeline run details.

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		9736f0b6-278c-49d2-be31-f3d07188f
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		f6bd763c-57b3-4e19-b399-2227a633b
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		86c820cb-5407-4905-9a92-b066327ad
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	21s	integrationRuntime1		34943b97-1b19-4f01-8be3-ab081e616
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		ccfc714-9831-4d76-a3e0-573362d5d

3. **Data Cleaning and Processing**: Utilize Databricks to clean and transform the data, storing the output in the processed folder.

The screenshot shows the Databricks interface. On the left, the 'Overview' pane is active, displaying a table of data. The table has columns: Name, Modified, Access tier, Archive status, Blob type, Size, and Lease state. The data is organized into folders: Address, Customer, and CustomerAddress. The 'Product' folder contains a table with 6 rows of data.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
Address						...
Customer						...
CustomerAddress						...
Product						...
Address	11/29/2024, 6:33:26 PM	Hot (inferred)		Block blob	0 B	Available
Customer	11/29/2024, 6:46:22 PM	Hot (inferred)		Block blob	0 B	Available
CustomerAddress	11/29/2024, 6:46:26 PM	Hot (inferred)		Block blob	0 B	Available
Product	11/29/2024, 6:46:30 PM	Hot (inferred)		Block blob	0 B	Available

4. **ETL and Gold Zone Storage**: Perform ETL operations and move the final transformed data to the gold folder in Azure Data Lake Storage.

Prerequisites

- An Azure account with access to Azure Data Factory, Azure Data Lake Storage, and Databricks.
- On-premises Microsoft SQL database with necessary credentials.

- Installed and configured **Azure Data Gateway** to connect the on-premises SQL database to Azure.
- Databricks cluster configured and running.
- Python and/or PySpark knowledge for Databricks scripts.
- Proper **Access Control** setup to secure data access in Azure Data Lake Storage and Databricks.
- Proper login protection using azure key vault

Step-by-Step Implementation

Step 1: Extract Data from On-Premises SQL Database

1. **Set Up Integration Runtime in Azure Data Factory**:

- Go to Azure Data Factory > Manage > Integration Runtimes.
- Set up a self-hosted integration runtime to connect to the on-premises SQL database.

The screenshot displays the Azure Data Factory 'Integration runtimes' management interface. The left sidebar shows the navigation menu with 'Integration runtimes' selected. The main area shows a table of existing runtimes:

Name	Type	Sub-type	Status	Related
AutofresolveIntegrationRuntime	Azure	Public	Running	0
IntegrationRuntime1	Self-Hosted	---	Running	1

The right-hand pane shows the 'Edit integration runtime' configuration for 'IntegrationRuntime1'. It includes fields for Name, Description, and authentication keys. The 'Name' field is set to 'IntegrationRuntime1' and the 'Description' is 'for Sql database server'. The 'Authentication key' section shows two keys, Key1 and Key2, both with values starting with 'IR@x328262b-49db-425b-9ff6-2cc4b402d792@transferdata214@servi...'.

- Ensure the local runtime is installed on a machine with access to the on-premises SQL server. Follow the Azure documentation to configure it properly.

Microsoft Integration Runtime Configuration Manager

Home Settings Diagnostics Update Help

✓ Self-hosted node is connected to the cloud service

Data Factory: transferdata214
Integration Runtime: integrationRuntime1
Node: MINE

Stop Service

Data Source Credential ⓘ

Credential store: On-premises
Credential status: In sync
Last backup time: N/A

Generate Backup Import Backup

✓ Connected to the cloud service (Data Factory V2)

2. **Create a Linked Service for SQL Database**:

- In ADF, create a new linked service pointing to the on-premises SQL database.
- Test the connection to ensure proper integration.

Microsoft Azure | Data Factory | transferdata214

Search factory and documentation

main branch Validate all Save all Publish

General
Factory settings
Connections
Linked services
Integration runtimes
Microsoft Purview
Source control
Git configuration
ARM template
Author
Triggers
Global parameters
Data flow libraries
Security
Credentials
Customer managed key
Outbound rules
Managed private endpoints
Workflow orchestration manager
Apache Airflow

Linked services

Linked service defines the connection information to a data store or compute. Learn more ⓘ

+ New

Filter by name Annotations: Any

Showing 1 - 6 of 6 items

Name	Type	Related
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	2
AzureDataLakeHttp	Azure Data Lake Storage Gen2	1
AzureKeyVault1	Azure Key Vault	0
AzureSqlDatabase1	Azure SQL Database	2
HttpServer1	HTTP	2
onpremise	SQL server	1

Edit linked service

SQL server Learn more ⓘ

Name * onpremise

Description

Connect via integration runtime * ⓘ integrationRuntime1

Version Recommended Legacy

Server name * MINE

Database name * AdventureWorksLT2022

Authentication type SQL authentication

User name * mik

Password * Password Azure Key Vault

Always encrypted Always encrypted

Encrypt Mandatory

Trust server certificate

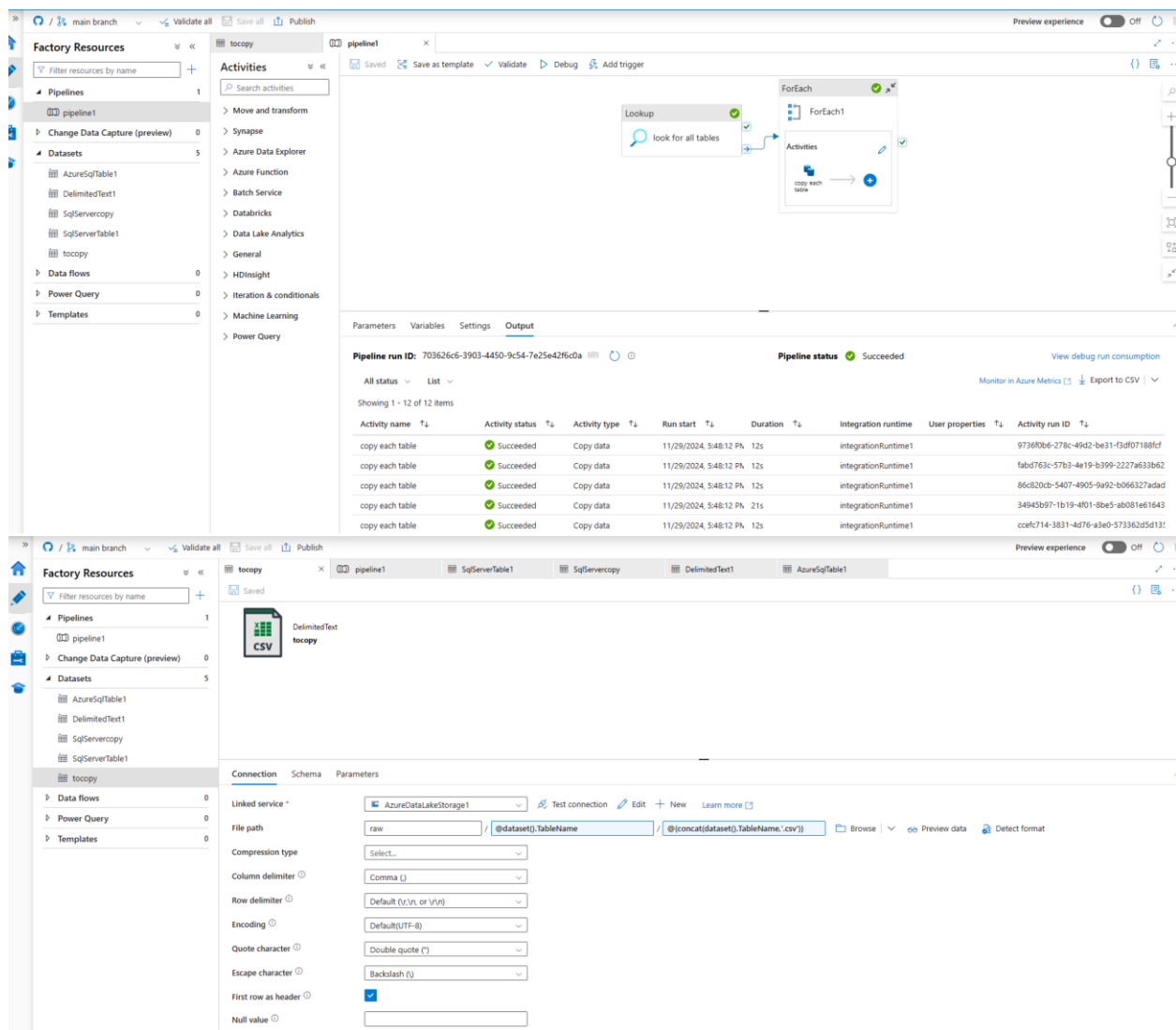
Save Cancel Test connector

3. **Define Access Control**:

- Ensure the self-hosted runtime has the necessary permissions to access the on-premises database.
- Configure secure credentials, such as a managed identity or key vault, to protect sensitive information.

4. **Build a Pipeline to Extract Data**:

- Create a new pipeline in ADF.
- Add a "Copy Data" activity to pull data from the SQL database.
- Define the source as the SQL database linked service.



Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		9736f0b6-278c-49d2-b631-f3df07188cf
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		fabd763c-57b3-4e19-b399-9227a633b62
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		86c820cb-5407-4905-9a92-b066327adad
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	21s	integrationRuntime1		34945b97-1b19-4f01-8be5-ab081e61643
copy each table	Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		cc0c714-3831-4d76-a3e0-573362d5d131

Connection	Schema	Parameters
Linked service *		Test connection Edit + New Learn more
File path	raw	@dataset().TableName @@concat(dataset().TableName,'.csv')
Compression type	Select...	
Column delimiter	Comma (,)	
Row delimiter	Default (\n or \r\n)	
Encoding	Default(UTF-8)	
Quote character	Double quote (")	
Escape character	Backslash (\)	
First row as header	<input checked="" type="checkbox"/>	
Null value		

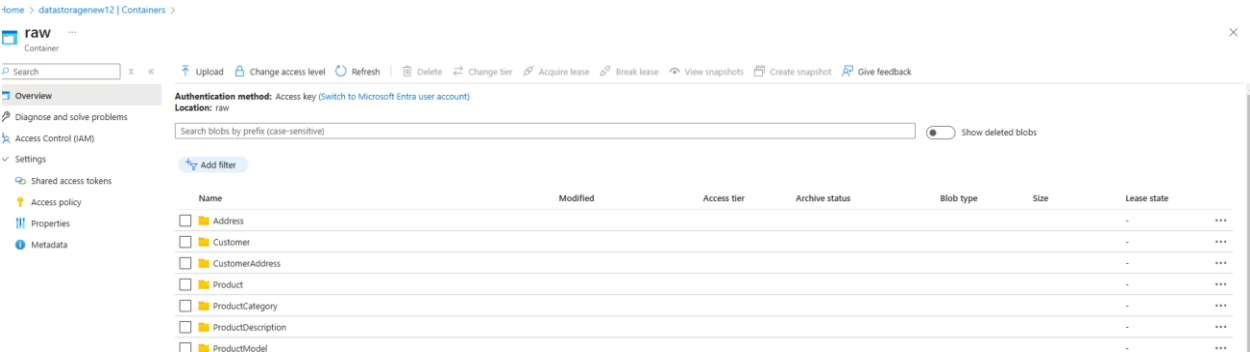
Step 2: Transfer Data to Raw Folder in Azure Data Lake Storage

1. **Set Up Azure Data Lake Linked Service**:

- In ADF, create a linked service for Azure Data Lake Storage.

2. **Configure the Destination in Copy Activity**:

- In the "Copy Data" activity, set the destination as the raw folder in ADLS.
- Define the folder structure to store the raw data



3. **Trigger the Pipeline**:

- Run the pipeline manually or schedule it for periodic runs.
- Monitor pipeline execution to ensure successful data transfer.

Showing 1 - 12 of 12 items							
Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		9736f0b6-278c-49d2-be31-f3df07188cf
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		fabd763c-57b3-4e19-b399-2227a633b62
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		86c820cb-5407-4905-9a92-b066327adad
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	21s	integrationRuntime1		34945b97-1b19-4f01-8be5-ab081e61643
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	12s	integrationRuntime1		cccf714-3831-4d76-a3e0-573362d5d13
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	20s	integrationRuntime1		299ccabc-df0b-4f66-9447-66ea0d6d80c9
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	18s	integrationRuntime1		8462d471-c115-44a8-9e98-1f4e88e173ef
copy each table	✔ Succeeded	Copy data	11/29/2024, 5:48:12 PM	20s	integrationRuntime1		37f9d4eb-23bc-4782-9d79-a94a081d389

Step 3: Clean and Process Data with Databricks

1. ADLS in Databricks**:

- Use Databricks forADLS raw folder:

2. **Load Raw Data in Databricks Notebook**:

```
07:24 PM (1s) 1 Python
secret=dbutils.secrets.get('databricklogin','databricklogin')

07:25 PM (<1s) 2
application_id="88dec17f-62f0-4cab-82b7-b9022c2cc21c"
service_credential=secret
directory_id="037e0179-cd0e-4201-951c-167b7554d77a"
spark.conf.set("fs.azure.account.auth.type.datastoragenew12.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.datastoragenew12.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.datastoragenew12.dfs.core.windows.net", application_id)
spark.conf.set("fs.azure.account.oauth2.client.secret.datastoragenew12.dfs.core.windows.net", service_credential)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.datastoragenew12.dfs.core.windows.net", f"https://login.microsoftonline.com/{directory_id}/oauth2/token")

07:25 PM (1s) 3
display(dbutils.fs.ls("abfss://raw@datastoragenew12.dfs.core.windows.net"))

(2) Spark Jobs
```

3. **Perform Data Cleaning and Transformation**:

```
06:53 PM (1s) 4
df_a = spark.read \
    .option("header", "true") \
    .option("delimiter", ",") \
    .csv("abfss://raw@datastoragenew12.dfs.core.windows.net/Address/Address.csv")
display(df_a)

(2) Spark Jobs
df_a: pyspark.sql.dataframe.DataFrame = [AddressID: string, AddressLine1: string ... 7 more fields]
```

	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid
1	9	8713 Yosemite Ct.	null	Bothell	Washington	United States	98011	260af621-76d7-4c78-9441-144fd139821a
2	11	1318 Lasalle Street	null	Bothell	Washington	United States	98011	981b3303-aca2-49c7-9a96-b670785b269
3	25	9178 Jumping St.	null	Dallas	Texas	United States	75201	c8df3bd9-48f0-4654-a8dd-14a67a843c6
4	28	9228 Via Del Sol	null	Phoenix	Arizona	United States	85004	12ae5ee1-fc3e-468b-9e92-3b970b169774
5	32	26910 Indela Road	null	Montreal	Quebec	Canada	H1Y 2H5	84a95f62-3ae8-4e7e-bbd5-5a6f00cd982d
6	185	2681 Eagle Peak	null	Bellevue	Washington	United States	98004	7bcc4442-2268-46cc-8472-14c44c14e98c
7	297	7943 Walnut Ave	null	Renton	Washington	United States	98055	52410da4-2778-4b1d-a599-95746625ce6d
8	445	6388 Lake City Way	null	Burnaby	British Columbia	Canada	V5A 3A6	53572f25-9133-4a8b-a065-102f35416ee
9	446	52560 Free Street	null	Toronto	Ontario	Canada	M4B 1V7	801a1dfc-5125-486b-aa84-ccb2ec57ca4
10	447	22580 Free Street	null	Toronto	Ontario	Canada	M4B 1V7	88cee379-dbb8-433b-b84e-a35e09435500
11	448	2575 Bloor Street East	null	Toronto	Ontario	Canada	M4B 1V6	2df6d0ad-0926-4f34-a450-9b1083150cbf
12	449	Station E	null	Chalk River	Ontario	Canada	K0J 1J0	8b5a7729-cb75-4303-a607-7f9793b4e94f
13	450	575 Rue St Amable	null	Quebec	Quebec	Canada	G1R	5f3c345a-6475-41d5-b17b-db8d27733bb1
14	451	2512-4th Ave Sw	null	Calgary	Alberta	Canada	T2P 2G8	49644f1e-6f90-46d9-8dbb-9db15f0ef7ec

```

from pyspark.sql.functions import col, date_format

# Extract the date from the ModifiedDate column and create a new column 'Date' and drop the original column 'ModifiedDate'
df_ac = df_a.withColumn("Date", date_format(col("ModifiedDate"), "yyyy-MM-dd")) \
              .drop("ModifiedDate")

display(df_ac)

```

(1) Spark Jobs

df_ac: pyspark.sql.dataframe.DataFrame = [AddressID: string, AddressLine1: string ... 7 more fields]

	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid
1	9	8713 Yosemite Ct.	null	Bothell	Washington	United States	98011	268af621-76d7-4c78-9441-144fd139821a
2	11	1318 Lasalle Street	null	Bothell	Washington	United States	98011	981b3303-aca2-49c7-9a96-fb670785b269
3	25	9178 Jumping St.	null	Dallas	Texas	United States	75201	c8df3bd9-48f0-4654-a8dd-14a67a84d3c6
4	28	9228 Via Del Sol	null	Phoenix	Arizona	United States	85004	12ae5ee1-fc3e-468b-9b92-3b970b169774
5	32	26910 Indela Road	null	Montreal	Quebec	Canada	H1Y 2H5	84a93f62-3ae8-4e7e-bbd5-5a6f00cd982d
6	185	2681 Eagle Peak	null	Bellevue	Washington	United States	98004	7bccf442-2268-46cc-8472-14c44c14e98c
7	297	7943 Walnut Ave	null	Renton	Washington	United States	98055	52410da4-2778-4b1d-a599-95746625ce6d
8	445	6388 Lake City Way	null	Burnaby	British Columbia	Canada	V5A 3A6	53572f25-9133-4a8b-a065-102ff35416ee
9	446	52560 Free Street	null	Toronto	Ontario	Canada	M4B 1V7	801a1dfc-5125-486b-aa84-ccb2ec57ca4
10	447	22580 Free Street	null	Toronto	Ontario	Canada	M4B 1V7	88cee379-dbb8-433b-b84e-a35e09435500
11	448	2575 Bloor Street East	null	Toronto	Ontario	Canada	M4B 1V6	2df6d0ad-0926-4f34-a450-9b1083150cbf
12	449	Station E	null	Chalk River	Ontario	Canada	K0J 1J0	8b5a7729-cb75-4303-a607-7f9793b4d94f
13	450	575 Rue St Amable	null	Quebec	Quebec	Canada	G1R	5f3c345a-6475-41d5-b17b-db8d27733bb1
14	451	2512-4th Ave Sw	null	Calgary	Alberta	Canada	T2P 2G8	49644f1e-6f90-46d9-8dbb-9db15f0ef7ec

4. **Save Cleaned Data to Processed Folder**:

```

df_ac.write.format("delta").mode("append").save("abfss://proccesed@datastoragenew12.dfs.core.windows.net/Address")

```

(4) Spark Jobs

Step 4: ETL and Load Data into Gold Zone

1. **Load Processed Data**:

07:25 PM (<1s)

2

Python

```
service_credential=secret
directory_id="037e0179-cd0e-4201-951c-167b7554d77a"
spark.conf.set("fs.azure.account.auth.type.datastoragenew12.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.datastoragenew12.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.
ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.datastoragenew12.dfs.core.windows.net", application_id)
spark.conf.set("fs.azure.account.oauth2.client.secret.datastoragenew12.dfs.core.windows.net", service_credential)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.datastoragenew12.dfs.core.windows.net", f"https://login.microsoftonline.com/{directory_id}/
oauth2/token")
```

07:09 PM (1s)

3

```
display(dbutils.fs.ls("abfss://proccesed@datastoragenew12.dfs.core.windows.net"))
```

(2) Spark Jobs

Table

+

Q

Y

□

	path	name	size	modificationTime
1	abfss://proccesed@datastoragenew12.dfs.core.windows.net/Address/	Address/	0	1732923206000
2	abfss://proccesed@datastoragenew12.dfs.core.windows.net/Customer/	Customer/	0	1732923982000
3	abfss://proccesed@datastoragenew12.dfs.core.windows.net/CustomerAddres...	CustomerAddres...	0	1732923986000
4	abfss://proccesed@datastoragenew12.dfs.core.windows.net/Product/	Product/	0	1732923990000

4 rows | 1.39 seconds runtime

Refreshed 2 hours ago

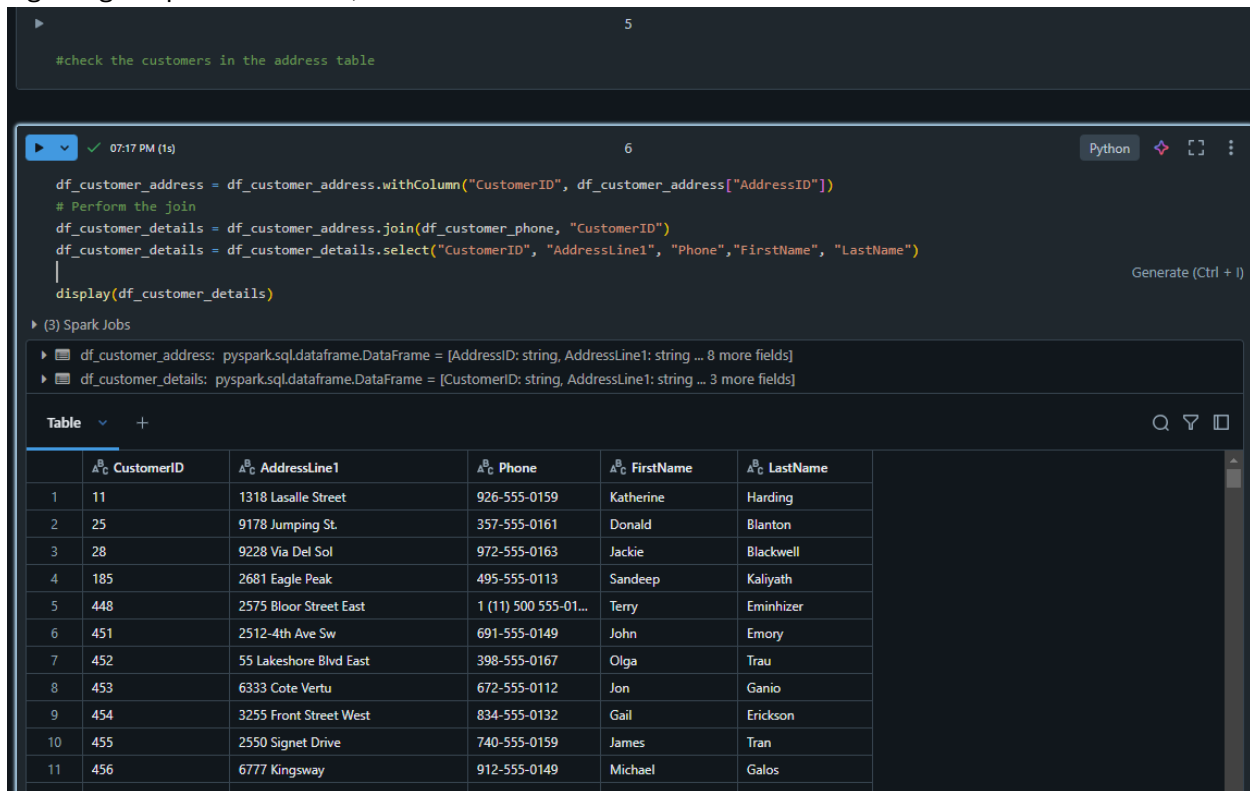
07:12 PM (1s)

4

2. **Perform ETL Transformations**:

- Apply final transformations, aggregations, or calculations for analytics.

--getting the phonenummer , address and name of the customers



The screenshot shows a Databricks notebook interface. At the top, a code cell (5) contains the comment "#check the customers in the address table". Below it, another code cell (6) contains Python code for a join operation. The code defines `df_customer_address` with a new column "CustomerID", performs a join with `df_customer_phone`, selects specific columns, and displays the result. The output shows two DataFrames: `df_customer_address` and `df_customer_details`. Below the code, a table view of `df_customer_details` is displayed, showing columns: CustomerID, AddressLine1, Phone, FirstName, and LastName. The table contains 11 rows of customer data.

```
#check the customers in the address table
```

```
df_customer_address = df_customer_address.withColumn("CustomerID", df_customer_address["AddressID"])
# Perform the join
df_customer_details = df_customer_address.join(df_customer_phone, "CustomerID")
df_customer_details = df_customer_details.select("CustomerID", "AddressLine1", "Phone", "FirstName", "LastName")
display(df_customer_details)
```


(3) Spark Jobs

- df_customer_address: pyspark.sql.dataframe.DataFrame = [AddressID: string, AddressLine1: string ... 8 more fields]
- df_customer_details: pyspark.sql.dataframe.DataFrame = [CustomerID: string, AddressLine1: string ... 3 more fields]

	CustomerID	AddressLine1	Phone	FirstName	LastName
1	11	1318 Lasalle Street	926-555-0159	Katherine	Harding
2	25	9178 Jumping St.	357-555-0161	Donald	Blanton
3	28	9228 Via Del Sol	972-555-0163	Jackie	Blackwell
4	185	2681 Eagle Peak	495-555-0113	Sandeep	Kaliyath
5	448	2575 Bloor Street East	1 (11) 500 555-01...	Terry	Eminhizer
6	451	2512-4th Ave Sw	691-555-0149	John	Emory
7	452	55 Lakeshore Blvd East	398-555-0167	Olga	Trau
8	453	6333 Cote Vertu	672-555-0112	Jon	Ganio
9	454	3255 Front Street West	834-555-0132	Gail	Erickson
10	455	2550 Signet Drive	740-555-0159	James	Tran
11	456	6777 Kingsway	912-555-0149	Michael	Galos

3. **Write Final Data to Gold Folder**:

- Save the transformed data to the gold folder:



The screenshot shows a Databricks notebook interface. A code cell (7) contains a command to write the transformed data to a gold folder. The command uses `df_customer_details.write.format("delta").mode("overwrite").save("abfss://gold@datastoragenew12.dfs.core.windows.net/customer_details")`. Below the code, the output shows "(5) Spark Jobs".

```
df_customer_details.write.format("delta").mode("overwrite").save("abfss://gold@datastoragenew12.dfs.core.windows.net/customer_details")
```

(5) Spark Jobs

Monitoring and Maintenance

1. **Pipeline Monitoring**:

- Use Azure Data Factory's monitoring dashboard to track pipeline execution.

2. **Databricks Job Scheduling**:

- Schedule Databricks notebooks as jobs for periodic processing.

3. **Error Handling**:

- Set up alerts and notifications in ADF and Databricks to handle errors.

Conclusion

This pipeline enables seamless extraction, transformation, and loading of data from an on-premises SQL database to Azure Data Lake Storage. By leveraging Azure Data Factory and Databricks, the process is automated, scalable, and suitable for advanced analytics and reporting.