Create Resource Group: Casestudy

- Resource Group name: “Casestudy”
- Location: “Central US”

Create Storage account for Data Lake storage Gen2: mystoragecd

- Storage account name: “mystoragecd”
- Enable hierarchical namespace: “Yes”
- Review + Create
- Create
- Created 3 Containers: (Bronze, Silver, Gold)

Create SQL Database: sourcedb

Basics
● Resource group - “Casestudy”
● Database name: “sourcedb”
● Server: - Create New
● Server name: “servercs”
● Location: “Central US”
● Authentication method: select “ SQL Authentication”
● Server admin login: “logincs”
● Password: ”****”
● Confirm Password: “*****” and select “OK”
- Back to Basics page
● Want to use SQL elastic pool?: “NO”
● Workload environment: “Production”
● Compute + storage: select “configure database”
● Service tier: “Basics (for less demanding workloads)” and drag basic to zero - select “Apply”
- Back to Basics page
● Backup storage redundancy: “Locally-redundant backup storage”
● Next Networking

Networking
● Connectivity method: “Public Endpoint”
● Allow Azure services and resources to access this server: “Yes”
● Add current client IP address: “Yes”
● Connection policy: “Default - Uses Redirect policy for all client connections originating inside of Azure (except Private Endpoint connections) and Proxy for all client connections originating outside Azure”
● Next Security (keep it default)

● Next Additional settings

Additional settings
● Use existing data: “None”
● Next Tags (no changes - keep it default)
● Next Review + create - Create

Data Organization

Step 1: Select “Query Editor” in SQL dashboard: “**Sourcedb”** and login
● User name: “logincs”
● Password: *******
Step 2: Select “Tables”
Step 3: In the query dashboard create a customer table.

Creating Tables in SQL Database:

● Create a customers table:

```
CREATE TABLE customers (
customer_id INT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
address VARCHAR(100),
city VARCHAR(50),
state VARCHAR(50),
zip VARCHAR(20)
);
```

● Create a Accounts table:

```
CREATE TABLE accounts (
account_id INT PRIMARY KEY,
customer_id INT,
account_type VARCHAR(50),
balance DECIMAL(10, 2),
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

● Create a transactions table:

```
CREATE TABLE transactions (
```

```
transaction_id INT PRIMARY KEY,
account_id INT,
transaction_date DATE,
transaction_amount DECIMAL(10, 2),
transaction_type VARCHAR(50),
FOREIGN KEY (account_id) REFERENCES accounts(account_id)
);
```

● Create a loan table:

```
CREATE TABLE loans (
loan_id INT PRIMARY KEY,
customer_id INT,
loan_amount DECIMAL(10, 2),
interest_rate DECIMAL(5, 2),
loan_term INT,
FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

● Create a loan_payments table:

```
CREATE TABLE loan_payments (
payment_id INT PRIMARY KEY,
loan_id INT,
payment_date DATE,
payment_amount DECIMAL(10, 2),
FOREIGN KEY (loan_id) REFERENCES loans(loan_id)
);
```

Inserting values into created tables:

Step 1: Insert sample data into all Tables(customer, account, transaction, loan, loan payment).

● Insert 10 values into the customer table.

```
INSERT INTO customers (customer_id, first_name, last_name, address, city,
state,
zip) VALUES
(1, 'John', 'Doe', '123 Elm St', 'Springfield', 'IL', '62701'),
(2, 'Jane', 'Smith', '456 Oak St', 'Chicago', 'IL', '60614'),
(3, 'Emily', 'Johnson', '789 Pine St', 'Dallas', 'TX', '75201'),
(4, 'Michael', 'Williams', '101 Maple St', 'Seattle', 'WA', '98101'),
```

```
(5, 'Sarah', 'Brown', '202 Birch St', 'New York', 'NY', '10001'),
(6, 'David', 'Jones', '303 Cedar St', 'Los Angeles', 'CA', '90001'),
(7, 'Laura', 'Garcia', '404 Willow St', 'San Francisco', 'CA', '94101'),
(8, 'James', 'Martinez', '505 Redwood St', 'Houston', 'TX', '77001'),
(9, 'Olivia', 'Davis', '606 Fir St', 'Boston', 'MA', '02101'),
(10, 'Daniel', 'Rodriguez', '707 Spruce St', 'Philadelphia', 'PA',
'19101');
```

- Insert 10 values into the accounts table.

```
INSERT INTO accounts (account_id, customer_id, account_type, balance)
VALUES
(1, 1, 'Checking', 1000.00),
(2, 1, 'Savings', 5000.00),
(3, 2, 'Checking', 1500.00),
(4, 2, 'Investment', 7500.00),
(5, 3, 'Savings', 2000.00),
(6, 4, 'Checking', 3000.00),
(7, 5, 'Checking', 2500.00),
(8, 6, 'Savings', 6000.00),
(9, 7, 'Investment', 8000.00),
(10, 8, 'Checking', 1200.00);
```

- Insert 10 values into the transaction table.

```
INSERT INTO transactions (transaction_id, account_id, transaction_date,
transaction_amount, transaction_type) VALUES
(1, 1, '2024-09-01', 200.00, 'Deposit'),
(2, 1, '2024-09-03', -100.00, 'Withdrawal'),
(3, 2, '2024-09-02', 300.00, 'Deposit'),
(4, 2, '2024-09-04', -50.00, 'Withdrawal'),
(5, 3, '2024-09-05', 150.00, 'Deposit'),
(6, 4, '2024-09-06', -200.00, 'Withdrawal'),
(7, 5, '2024-09-07', 250.00, 'Deposit'),
(8, 6, '2024-09-08', -300.00, 'Withdrawal'),
(9, 7, '2024-09-09', 400.00, 'Deposit'),
(10, 8, '2024-09-10', -150.00, 'Withdrawal');
```

- Insert 10 values into the loans table.

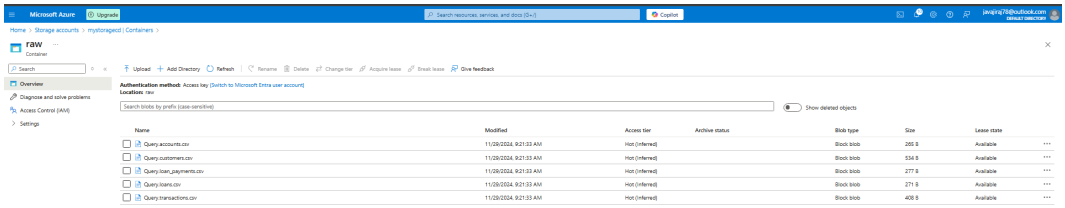
```
INSERT INTO loans (loan_id, customer_id, loan_amount, interest_rate,
loan_term) VALUES
(1, 1, 5000.00, 3.50, 12),
```

```
(2, 2, 7500.00, 4.00, 24),
(3, 3, 6000.00, 3.75, 18),
(4, 4, 10000.00, 4.25, 36),
(5, 5, 12000.00, 4.50, 48),
(6, 6, 8000.00, 3.90, 24),
(7, 7, 9500.00, 4.10, 30),
(8, 8, 11000.00, 4.00, 42),
(9, 9, 13000.00, 4.20, 54),
(10, 10, 7000.00, 3.85, 20);
```

● Insert 10 values into the loan_payments table.

```
INSERT INTO loan_payments (payment_id, loan_id, payment_date,
payment_amount)
VALUES
(1, 1, '2024-01-15', 250.00),
(2, 1, '2024-02-15', 200.00),
(3, 2, '2024-01-20', 150.00),
(4, 2, '2024-02-20', 400.00),
(5, 3, '2024-01-25', 400.00),
(6, 3, '2024-02-25', 200.00),
(7, 4, '2024-03-01', 250.00),
(8, 4, '2024-04-01', 200.00),
(9, 5, '2024-05-10', 400.00),
(10, 5, '2024-06-10', 400.00);
```

● Exporting this data in csv format and uploading it to Azure Data Lake storage Gen2 - Container: mystoragecd - Folder: raw

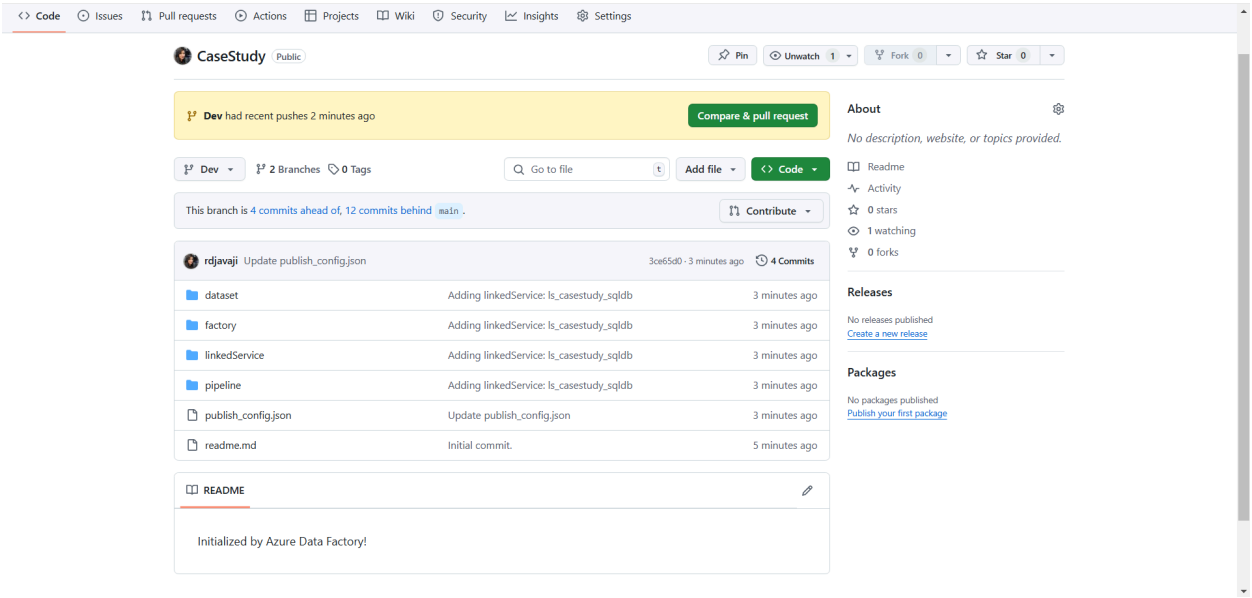


Create Azure Data Factory: source-raw

- Azure Data Factory name: “source-raw”
- Launch studio

Configure GitHub:

Select - Manage
● Select - Git Configuration - Configure
● Repository type : “GitHub”
● GitHub repository owner: “rdjavaji”
It will ask for login details email & password - Once logged in
● Repository name: “ADF-Key”
● Collaboration branch: create new : “Dev” or you can also keep the existing one which is “Main” or
you can also create a new branch.
● Cross check that the Dev branch is created in GitHub.



Creating Linked service for SQL Database: ls_casestudy_sqldb:

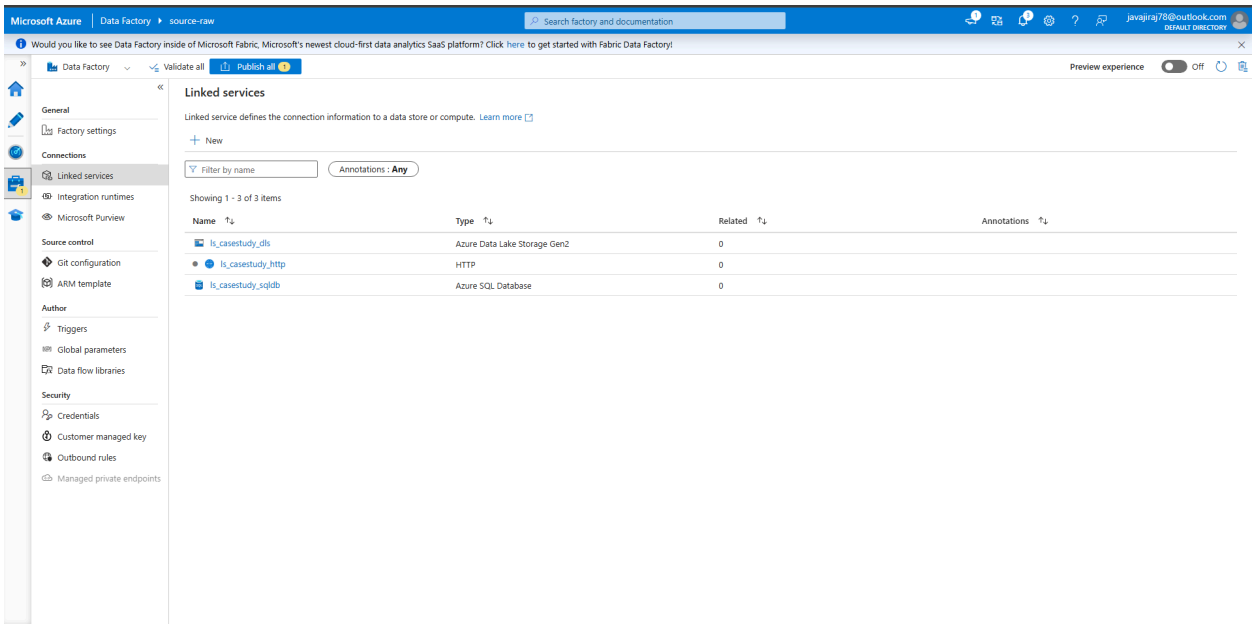
- Manage - Linked service - New
- Name: “ls_casestudy_sqldb”
- Server name: “servercs”
- Database name: “Sourcedb”
- Username: “logincs”
- Password: *******
- Test Connection - Create

Creating Linked service for HTTP API: ls_casestudy_http

- Create **Linked service**
- Open ADF - Manage - Linked services - New
- Search - HTTP - Select
- Name: “ls_casestudy_http”
- Base URL: https://github.com
- Server Certificate authentication - Enable
- Authentication Type - Anonymous
- Test Connection
- Create

Creating Linked service for DataLake Gen2: ls_casestudy_dl

- Manage - Linked Service - New
- Search: Data Lake storage Gen2
- Name: “ls_casestudy_dl”
- Authentication method: Account Key
- Connection String
- Azure Subscription select
- Storage account name: “mystoragecd” from dropdown
- Test Connection
- Create



Creating Datasets for SQL Database: ds_casestudy_sqldb

- Author - Datasets - New Dataset

- Search: **SQL Database** - Delimited text
- Name: “ds_casestudy_dl”
- Linked service: “ls_casestudy_dl”
- Table name: dbo.customers
- OK

Creating Datasets for Azure DataLake Storage gen2: ds_casestudy_dl

- Author - Datasets - New Dataset
- Search: Azure DataLake Storage gen2 - Delimited text
- Name: “ds_casestudy_dl”
- Linked service: “ls_casestudy_dl”
- File Path: Curated
- OK

Creating Datasets for HTTP: ds_casestudy_http

- Author - Datasets - New Dataset
- Search: HTTP - Delimited text
- Name: “ds_casestudy_http”
- Linked service: “ls_casestudy_http”
- Relative URL: rdjavaji/CaseStudy/blob/main/Data/Query.customers.csv
- Check First row as Header
- Import Schema - None
- Ok

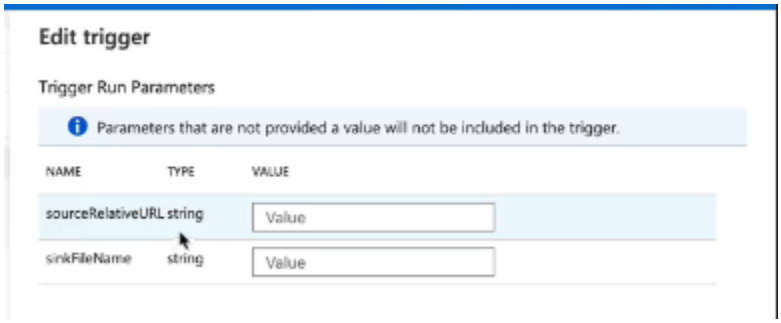
Creating Pipelines1:

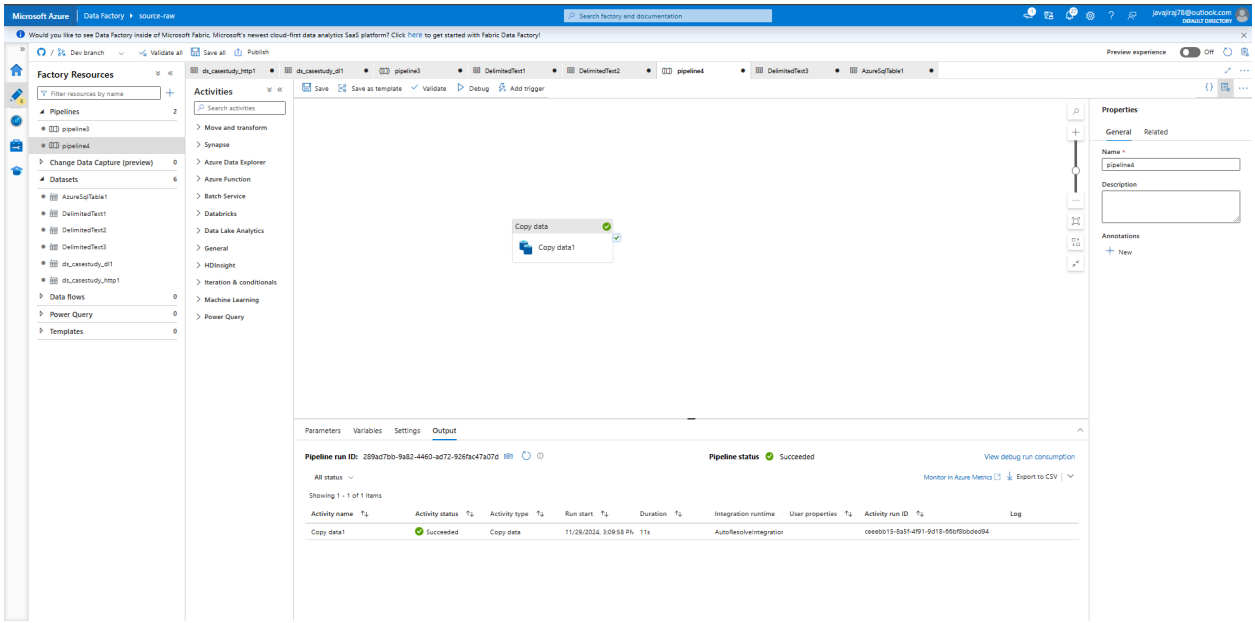
- From Activity - Copy Data - Drag and Drop
- **Source**: New - sql database - Name: “ds_casestudy_dl”
- Linked service: “ls_casestudy_dl”
- Table name: dbo.customers
- OK
- **Sink**: Search: Azure DataLake Storage gen2 - Delimited text
- Name: “ds_casestudy_dl”
- Linked service: “ls_casestudy_dl”
- File Path: bronze
- OK

Creating Pipelines2:

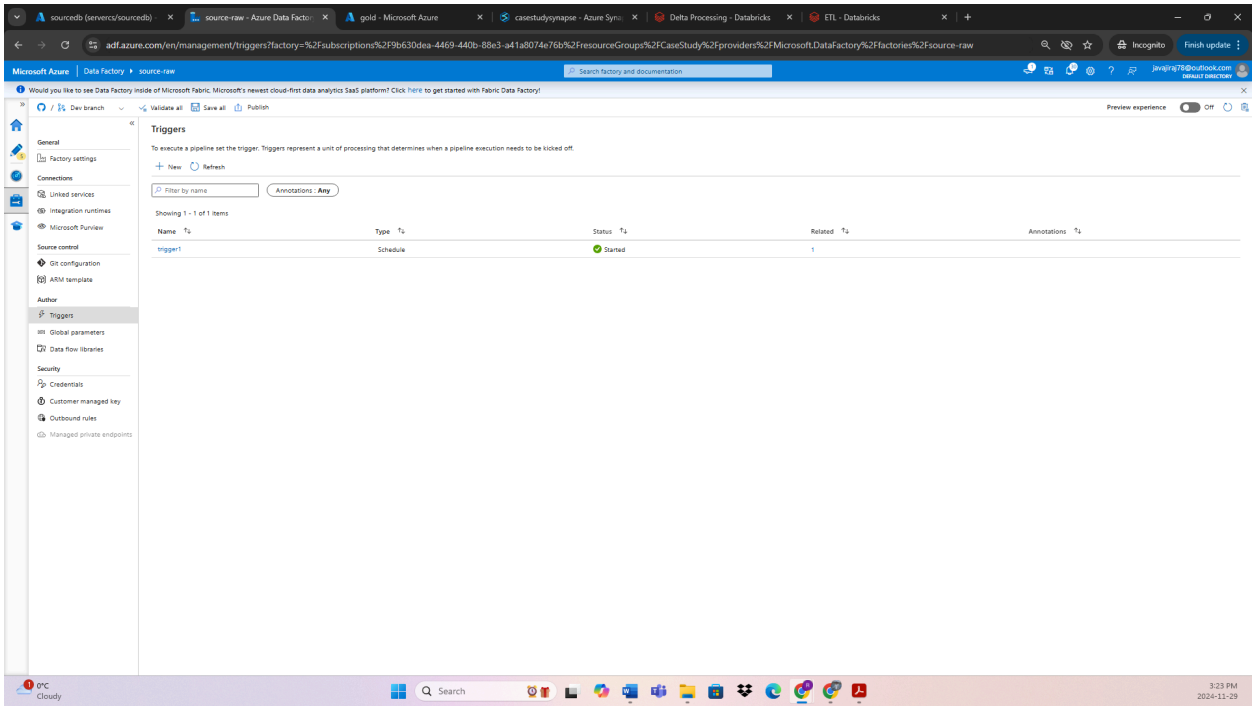
- **Source**: New - Http - Name: “ds_casestudy_httpl”
- Linked service: “ls_casestudy_http”
- Curated URL: rdjavaji/CaseStudy/blob/main/Data/Query.customers.csv
- OK
- **Sink**: Search: Azure DataLake Storage gen2 - Delimited text
- Name: “ds_casestudy_dl2”

- Linked service: “ls_casestudy_dl”
- File Path: curated
- OK
- Created Parameterised pipeline
- Validate
- Debug
- Publish all





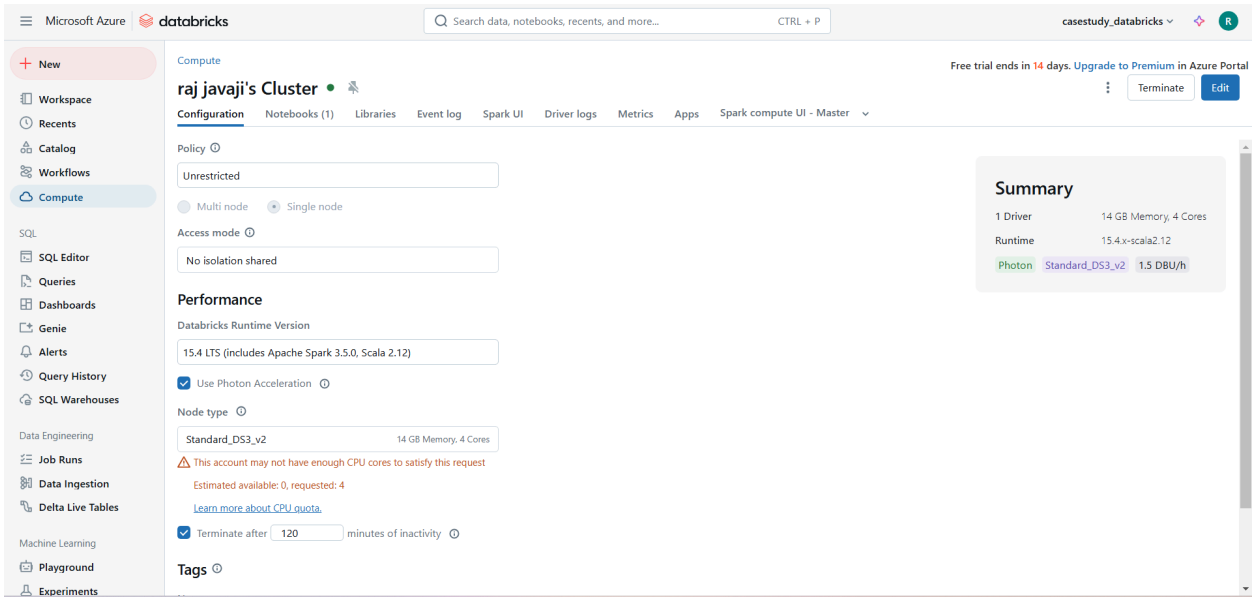
- Scheduled Trigger:



Creating Azure databricks: casestudy_databricks

Creating a cluster

- Select compute
- Select - create compute



Create a notebook:

- Select workspace - Create - folder

- Folder name “casestudy”
- In this folder creating notebook - Delta Processing

- Defining the Spark environment with Access Key

```
spark.conf.set(
"fs.azure.account.key.mystoragecd.dfs.core.windows.net",
"cohyF/qhtXtSyWejUVgk0hbR03J7============")
```

- Displaying the File from the curated container

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

- Reading Data from curated container

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("DataCleaning").getOrCreate()

# Read data from curated container

customers_df =
spark.read.csv("abfss://curated@mystoragecd.dfs.core.windows.net/customers.csv"
, header=True)

customers_df.show(5)
```

- Defining the Data from Curated Container

```
# Defining data from curated container
customers_df = customers_df.dropna()
```

- Removing the Duplicates

```
# Remove Duplicates
customers_df = customers_df.dropDuplicates()
```

- Defining Path

```
Define the paths to the Silver container
silver_customers =
"abfss://silver@mystoragecd.dfs.core.windows.net/delta/customers_delta"
```

- Moning Cleaned Data to silver container

```
# # Write the cleaned data back to the Silver container
# Rename columns to remove invalid characters
for col in customers_df.columns:
    new_col = col.replace(' ', '_').replace(';', '_').replace('{',
'_').replace('}', '_') \
                 .replace('(', '_').replace(')', '_').replace('\n',
'_').replace('\t', '_') \
                 .replace('=', '_')
    customers_df = customers_df.withColumnRenamed(col, new_col)

# Write the DataFrame to Delta format
customers_df.write.format("delta").mode("overwrite").save(silver_customers)
```

ETL Processing:

```
spark.conf.set(
"fs.azure.account.key.mystoragecd.dfs.core.windows.net",
"cohyF/qhtXtSyWejUVgk0hbR03J7============")
```

```
# Read Data from Silver Container:
customers_df
=spark.read.format("delta").load("abfss://silver@mystoragecd.dfs.core.windows.n
et/delta/customers_delta")
```

```
#defining the schema
columns = ["customer_id", "first_name", "last_name", "address", "city",
"state", "zip"]
```

```
# Define the data and columns
data = [
    (1, 'John', 'Doe', '123 Elm St', 'Springfield', 'IL', '62701'),
    (2, 'Jane', 'Smith', '456 Oak St', 'Chicago', 'IL', '60614'),
    (3, 'Emily', 'Johnson', '789 Pine St', 'Dallas', 'TX', '75201'),
    (4, 'Michael', 'Williams', '101 Maple St', 'Seattle', 'WA', '98101'),
    (5, 'Sarah', 'Brown', '202 Birch St', 'New York', 'NY', '10001'),
    (6, 'David', 'Jones', '303 Cedar St', 'Los Angeles', 'CA', '90001'),
    (7, 'Laura', 'Garcia', '404 Willow St', 'San Francisco', 'CA', '94101'),
    (8, 'James', 'Martinez', '505 Redwood St', 'Houston', 'TX', '77001'),
```

```
    (9, 'Olivia', 'Davis', '606 Fir St', 'Boston', 'MA', '02101'),
    (10, 'Daniel', 'Rodriguez', '707 Spruce St', 'Philadelphia', 'PA',
'19101'),
    (11, 'Andy', 'Josheny', '123 yulm St', 'Chicago', 'IL', '627098'),
    (12, 'Eva', 'antony', '153 balliol St', 'Chicago', 'PL', '667098')
]
columns = ["customer_id", "first_name", "last_name", "address", "city",
"state", "zip"]

# Create a DataFrame
df = spark.createDataFrame(data, columns)

# Display the DataFrame
display(df)
```

```
# Group customers by city and state, and count distinct zip codes
city_zip_count = df.groupBy("city",
"state").agg(countDistinct("zip").alias("distinct_zip_count"))

# Display the result
display(city_zip_count)

# Show the result
city_zip_count.show(truncate=False)
```

```
# Define the path for gold container

gold_delta = "abfss://gold@mystoragecd.dfs.core.windows.net/delta/gold_delta"

# Save the DataFrame in Delta format, overwriting if it exists

city_zip_count.write.format("delta").mode("overwrite").save(gold_delta)
```

- ETL performed data has been moved to the Gold container

Azure Synapse Analytics:

Step 1: Create azure synapse analytics

Basics
● Resource group - Create New - “casestudy”
● Workspace name: “casestudysynapse”
● Region: “Central US”
● Select Data Lake Storage Gen2: “From Subscription”
● Account name - create new - “mystoragecd”
● File system name - create new - “silver”
● Next Security - Login
● Next Networking (no changes - keep it default)
● Next Tags (no changes - keep it default)
● Next Review + create

Launch Synapse studio
- Data - + New SQL database - sql pool type: serverless - name: mybd1

● Develop - usedatabase: mybd1 (from dropdown) + SQL script - copy paste the script

```
-- Example for creating an external data source for the Silver container
CREATE EXTERNAL DATA SOURCE SilverDataSource
WITH (
LOCATION = 'abfss://silver@strgacc2831.dfs.core.windows.net'
);
```

● Data + sql database - serverless - name: database-my-rg1

● For the above script - run the script.

● Run the 2nd script:

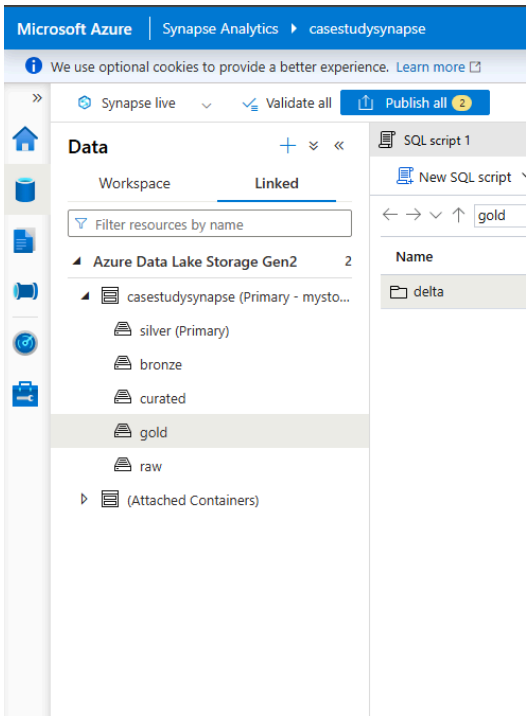
```
-- Example for creating an external data source for the Gold container
CREATE EXTERNAL DATA SOURCE GoldDataSource
WITH (
LOCATION = 'abfss://silver@strgacc2831.dfs.core.windows.net'
);
```

● Run the 3rd Script:

```
-- Example for creating a CSV file format
CREATE EXTERNAL FILE FORMAT CsvFileFormat
WITH (
FORMAT_TYPE = DELIMITEDTEXT,
FORMAT_OPTIONS (FIELD_TERMINATOR = ',', STRING_DELIMITER = '"', FIRST_ROW = 2)
);
```

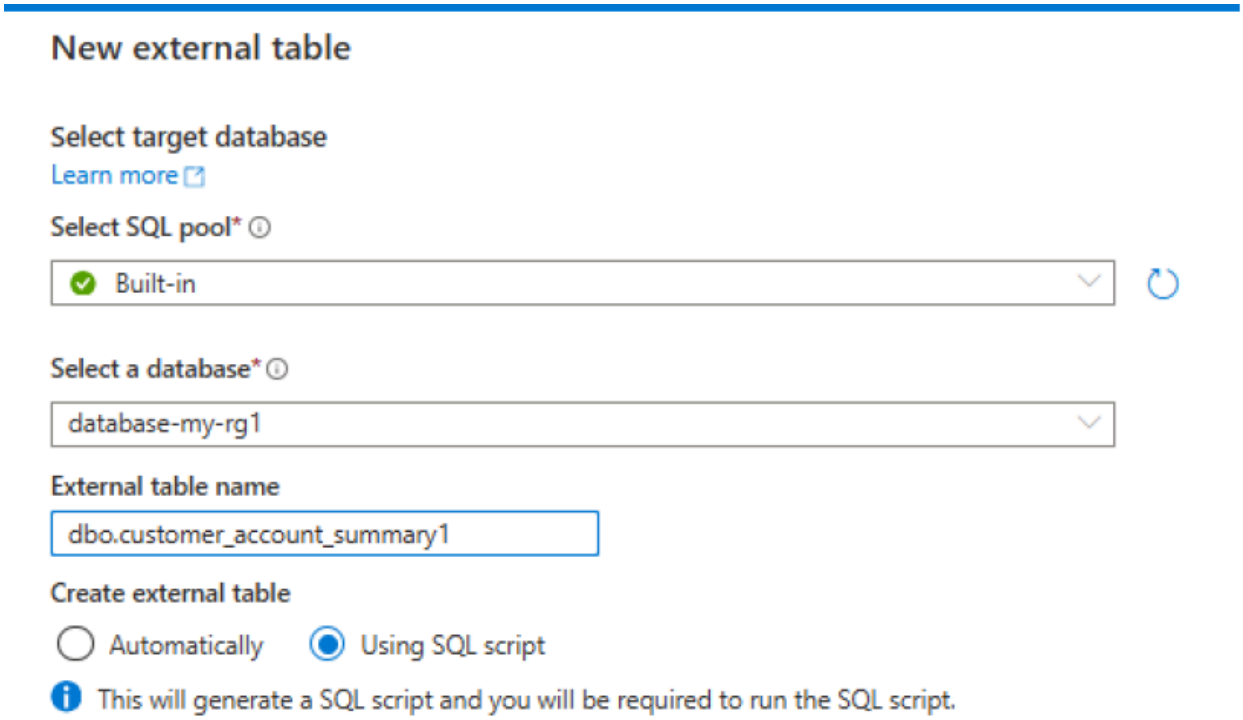
- Data - Linked

● Select - Linked - Select : “casestudysynapse” - You will be able to see the lists of containers which is already created.

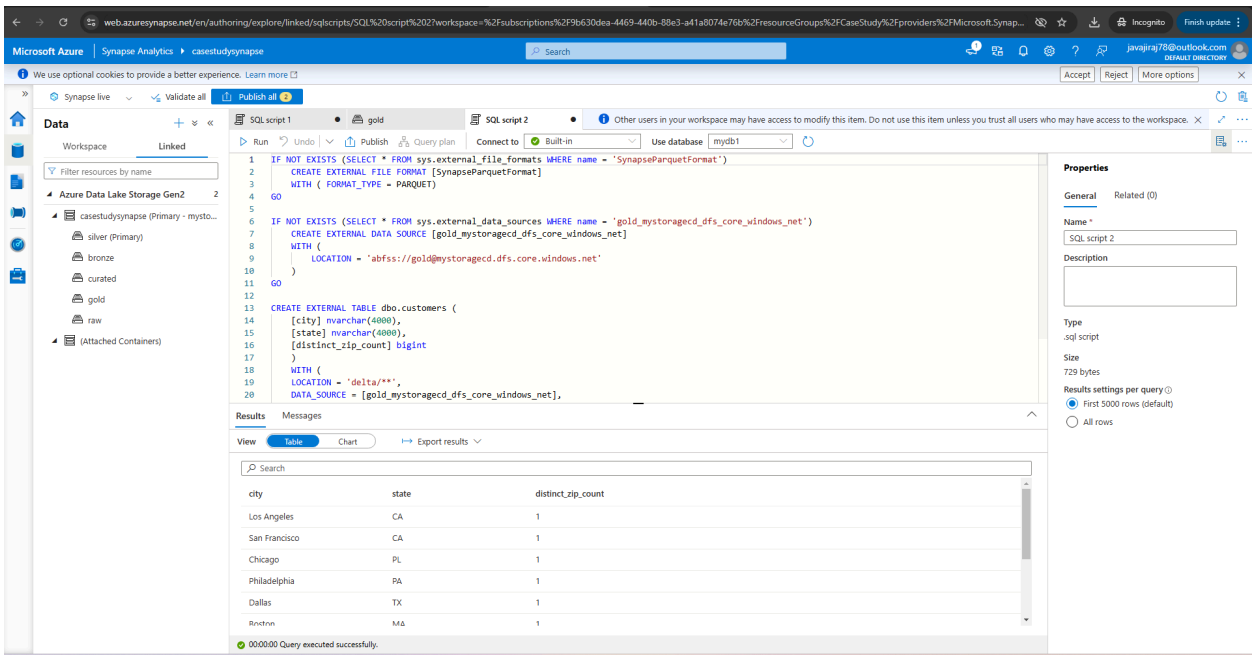


● Select - gold - Right click on select : “delta“ select : new sql script - create external table

● Continue - External table name: dbo.customer_account_summary1



- Open script - Run the entire script.



- Created New Resource group: cs-git
- Created New Azure Data Factory: cs-git-df
- Manage create new - Git configure - Create a branch as “QA”
- Go to Git-Hut Cross check the branch and create a Pull request to it.