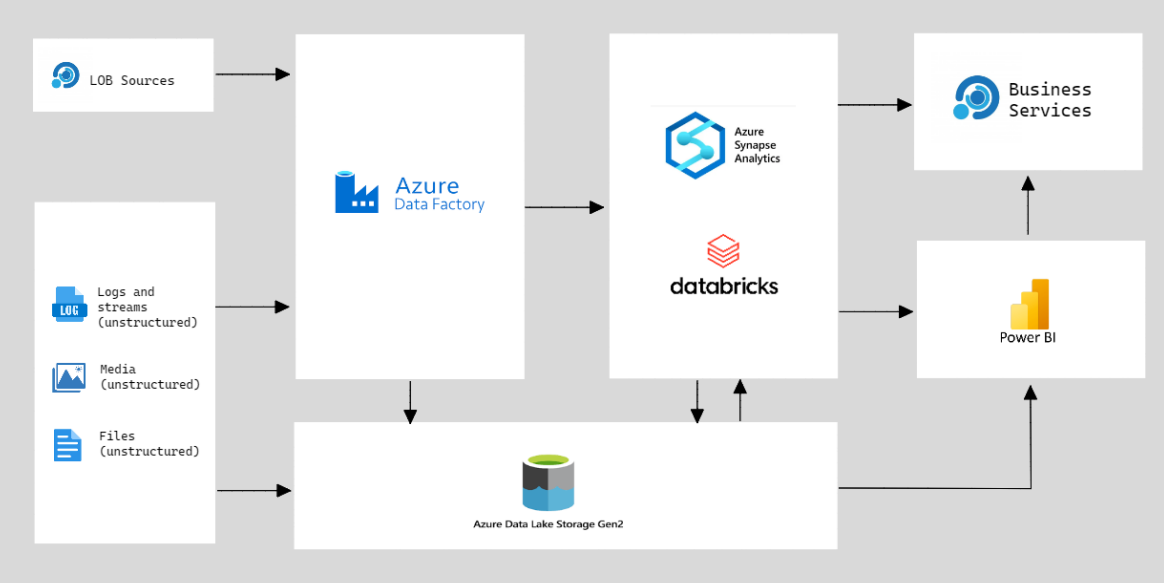
RETAIL SALES ANALYSIS

Group 06: Roshan Rawat, Gangothri Gadige, Chandan Kumar, Narendra Reddy, Faryar Memon

Modern Data Warehouse Architecture

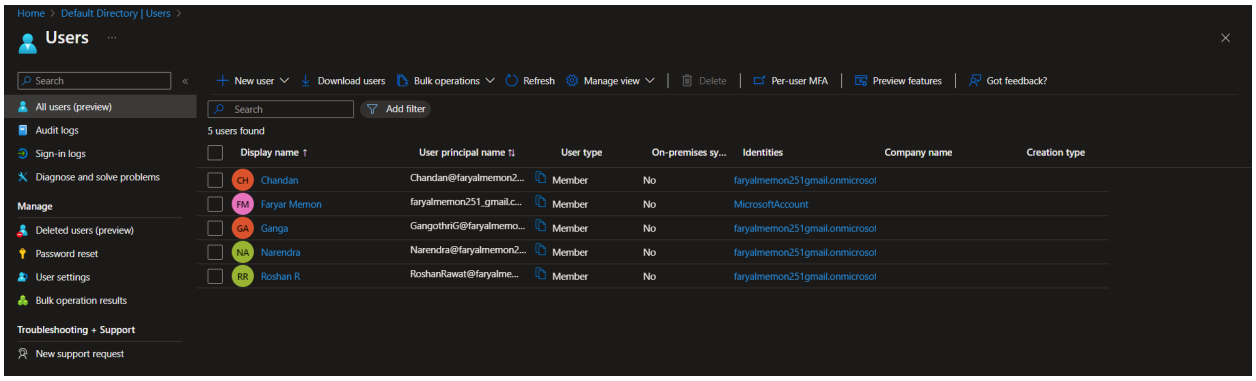


Tasks

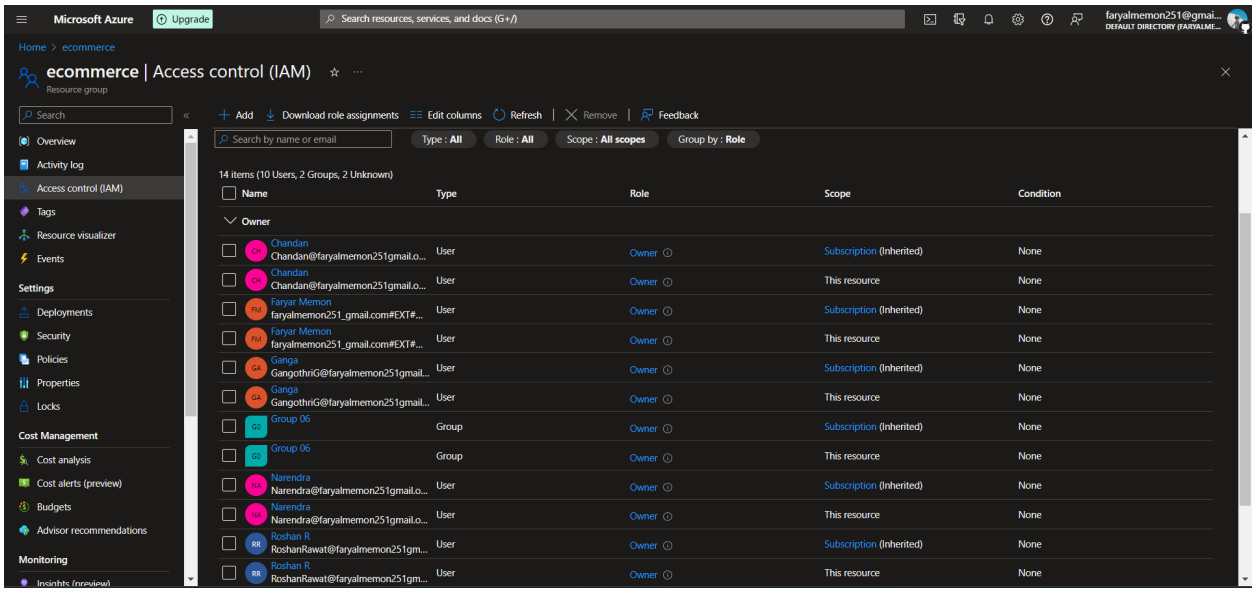
Source — Azure SQL	• Collecting the data from the client source
ETL — Azure Data Factory	• Using **ADF** to ingest data from client source **Azure SQL Database** to **Azure Data Lake Storage**
Analysis — Azure Synapse Analytics, databricks, Spark	• Integrating ADLS with **Synapse** using **Linked Services** & mounting ADLS to **Azure Databricks**
Visualization — Power BI	• Integrating the analysis results with **Power BI** for impactful visualizations and reporting

Data Governance

1. Create multiple users to work under the same resource group



Give privileges to the users in Resource Group and Subscriptions



2. Create Storage Account and Database

Data Ingestion

Client Source Azure SQL Database config

create table retail(region varchar(1000), country varchar(1000), item_type varchar(1000), sales_channel varchar(1000), order_priority varchar(1000), order_date varchar(2000), order_id int, ship_date varchar(1000), units_sold int, unit_price float, unit_cost float, total_revenue float, total_cost float, total_profit float)

CREATE MASTER KEY ENCRYPTION BY PASSWORD ='Roshan@3003'

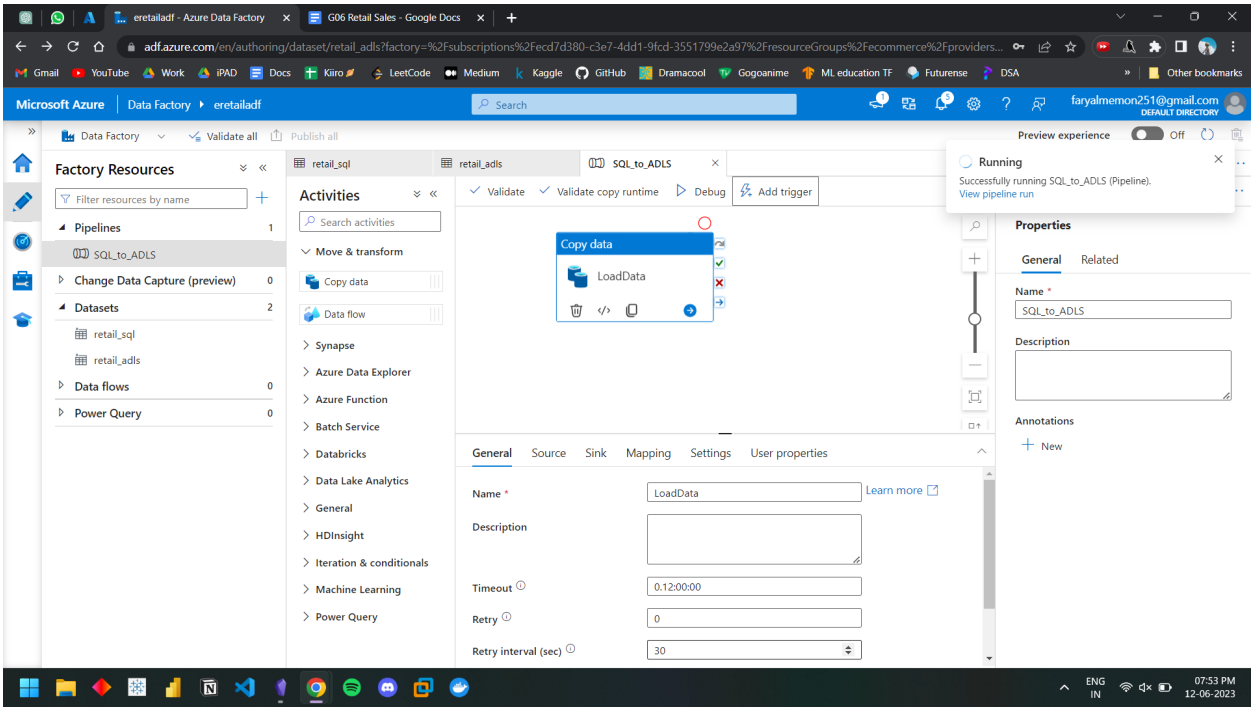
create database scoped credential MyAzureBlobStorageCredential
WITH IDENTITY ='SHARED ACCESS SIGNATURE',
SECRET = '<SAS token>'

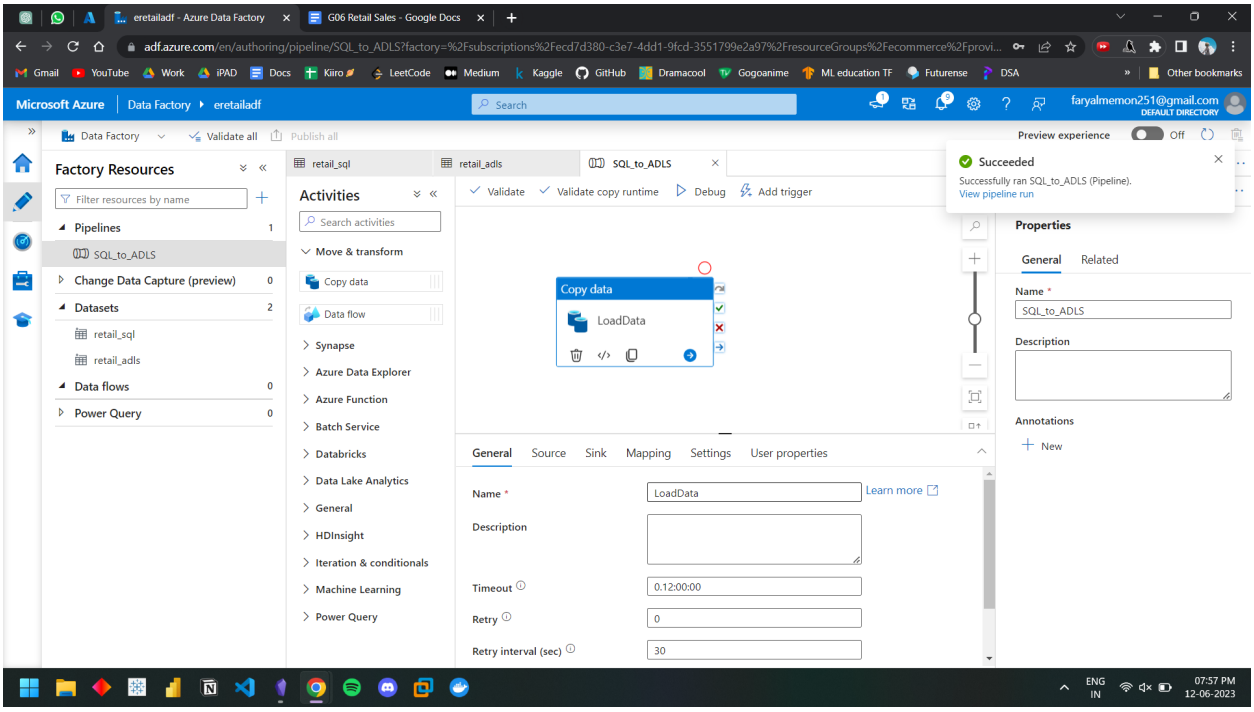
CREATE EXTERNAL DATA SOURCE MyAzureBlobStorage
WITH (TYPE =BLOB_STORAGE,
LOCATION = 'https://ecomfiles1.blob.core.windows.net/csvfiles/',
credential = MyAzureBlobStorageCredential)

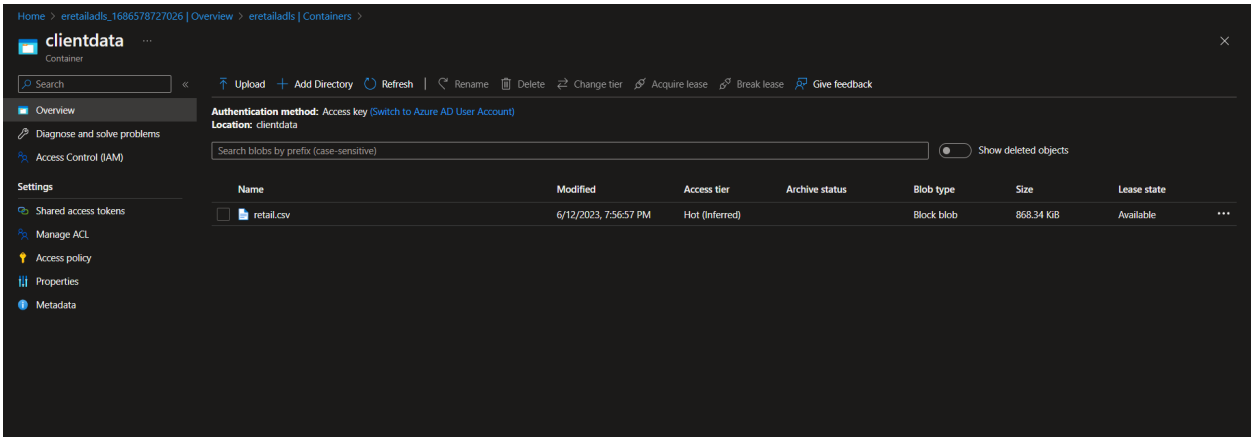
BULK INSERT dbo.retail from 'sales_data.csv'
with (check_constraints,
DATA_SOURCE = 'MyAzureBlobStorage',
datafiletype ='char',
fieldterminator =',',
rowterminator='0x0a',
firstrow=2,
keepidentity,
tablock)

Rule name	Start IPv4 address	End IPv4 address
ClientIPAddress_2023-6-4_14-48-11	122.171.16.101	122.171.16.101
Faryar	192.168.159.1	192.168.159.1
Ganga	192.168.1.20	192.168.1.20
Narendra	192.168.40.1	192.168.40.1
Roshan	192.168.56.1	192.168.56.1

Transferring Data from Client Source (Azure SQL Database) to ADLS using ADF



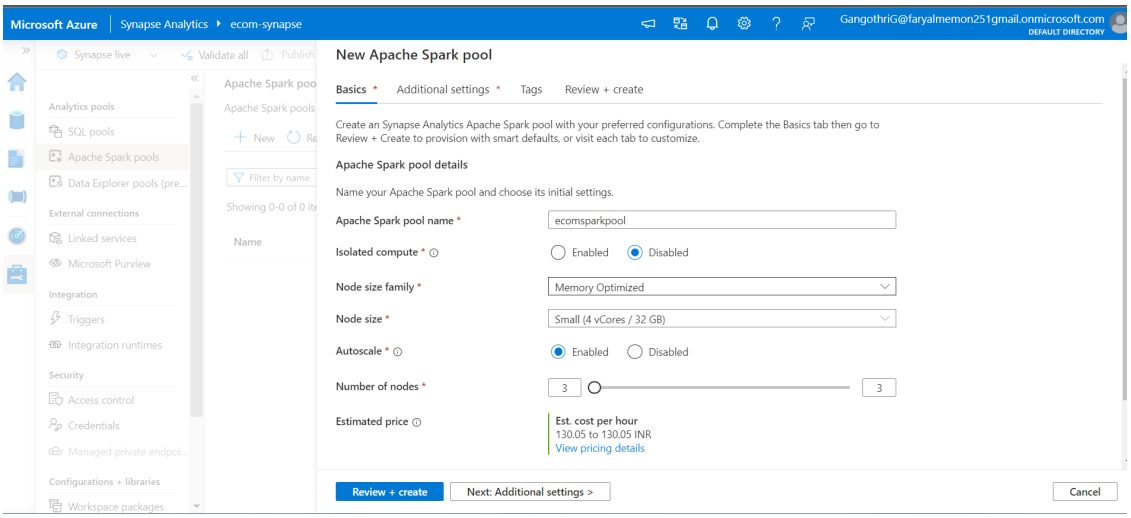


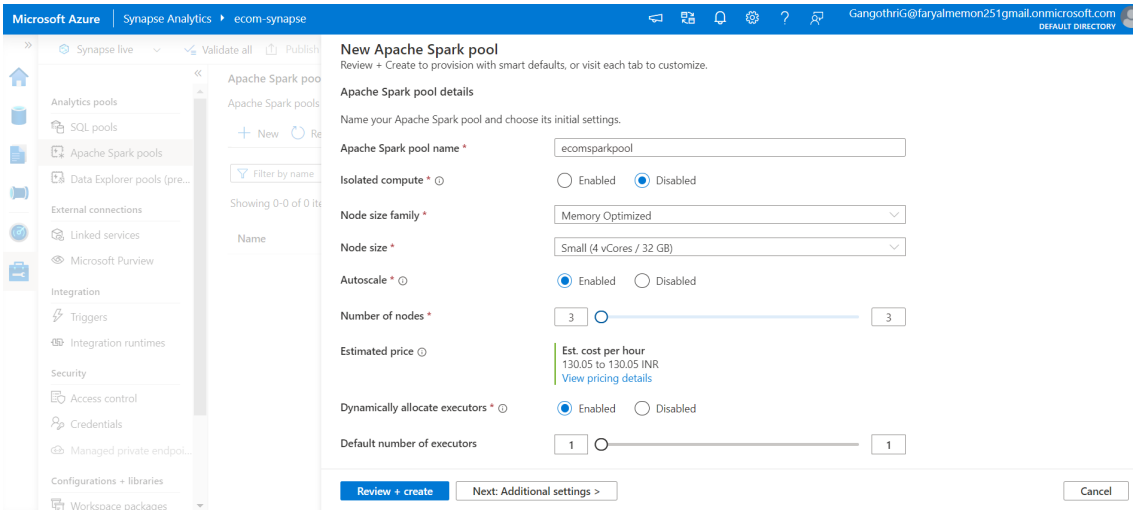

Home > eretailadls_1686578727026 | Overview > eretailadls | Containers >
clientdata
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Manage ACL
Access policy
Properties
Metadata
Upload
Add Directory
Refresh
Rename
Delete
Change tier
Acquire lease
Break lease
Give feedback
Authentication method: Access key (Switch to Azure AD User Account)
Location: clientdata
Search blobs by prefix (case-sensitive)
Show deleted objects
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
retail.csv
6/12/2023, 7:56:57 PM
Hot (Inferred)
Block blob
868.34 KiB
Available

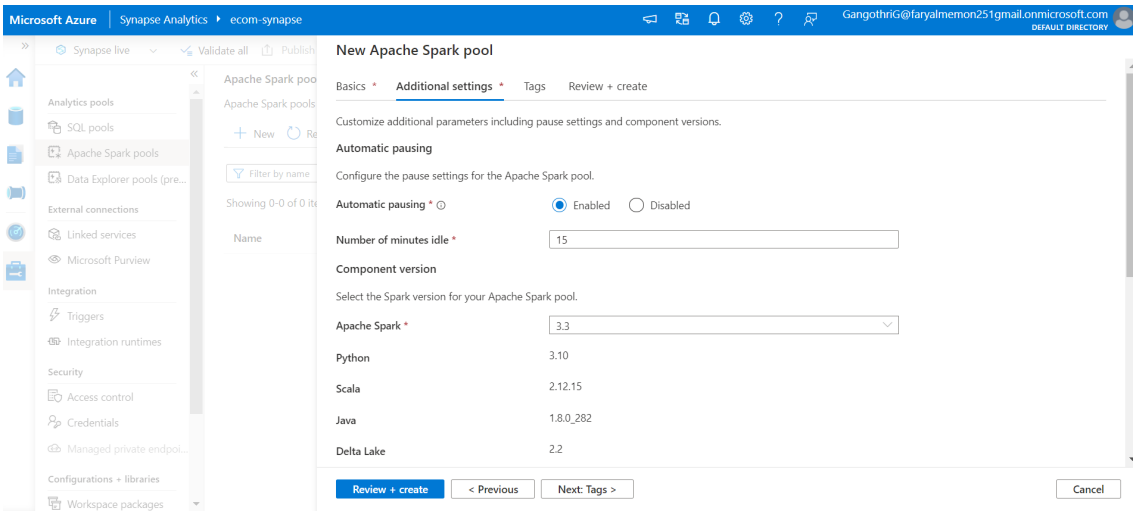
Analysis

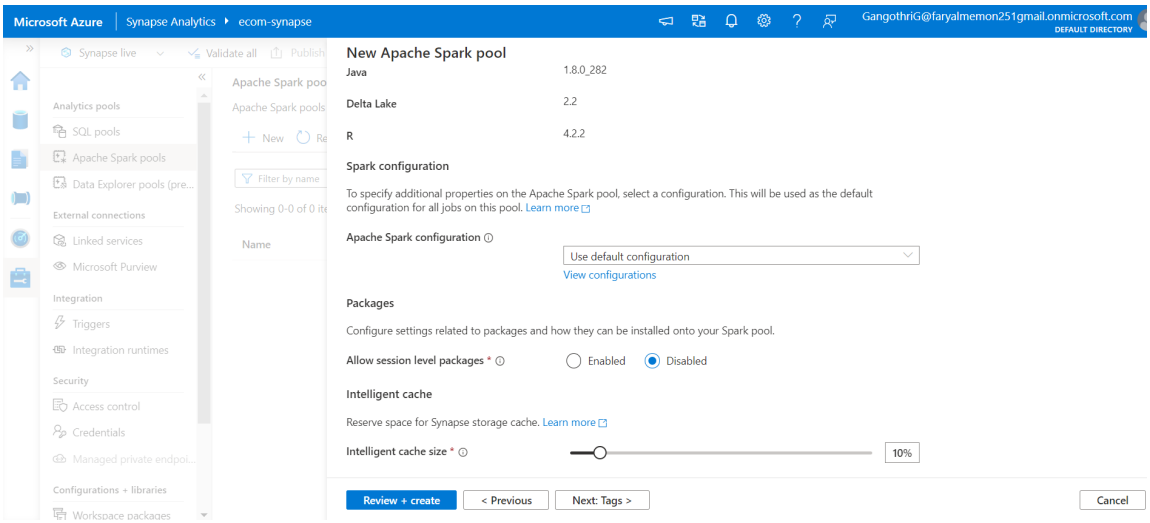
Synapse Studio

1. Create Spark Pool and give each user the appropriate permissions of the Synapse Workspace
2.









3. Analyzing Queries

 # A.7.Display the number of sales year wise .(Using pyspark)

```
from operator import add
year_and_revenue = rdd_df.map(lambda l: (l[5].year, l[11]))
yearly_sales = year_and_revenue.reduceByKey(add).sortBy(lambda x: x[0])
yearly_sales.collect()
```

 # A.8. Display the number of orders for each item.(Using pyspark)

```
from operator import add
each_item = rdd_df.map(lambda l: (l[2], 1))
item_ocount = each_item.reduceByKey(add).sortBy(lambda x: x[1], ascending=False)
item_ocount.collect()
```

 # A.9. Display the country with highest sale (Using pyspark)

```
from operator import add
country_order = rdd_df.map(lambda l: (l[1], l[11]))
country_revenue = country_order.reduceByKey(add).sortBy(lambda x: x[1], ascending=False)
country_revenue.take(10)
```

 # B.4 Yearly sum for specific country

```
from operator import add
iyear = 2013
year_and_revenue = rdd_df.map(lambda l: (l[5].year, l[11]))
yearly_sales = year_and_revenue.reduceByKey(add).sortBy(lambda x: x[0])
yearly_sales.filter(lambda l: l[0]==iyear).map(lambda x: x[1]).first()
```

 # B.6. monthly sales report of a particular year

```
from operator import add

iyear = 2014
each_month = rdd_df.map(lambda l: ((l[5].year, l[5].month), l[11]))
yearlymonth_sales = each_month.reduceByKey(add).sortBy(lambda x: x[0])
yearlymonth_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: (x[0][1],x[1])).collect()
```

 # B. 7) Quarterly sales report of a particular year

```
import math
def quarter(date):
   return math.ceil(date.month/3)

iyear = 2014
each_quarterly = rdd_df.map(lambda l: ((l[5].year, quarter(l[5])), l[11]))
yearlyquarter_sales = each_quarterly.reduceByKey(add).sortBy(lambda x: x[0])
yearlyquarter_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: (x[0][1], x[1])).collect()
```

 # B. 8, 11) For each quarter report for each year Write udf function for that. And for all year

```
import math
def quarter(date):
   return math.ceil(date.month/3)

each_quarter = rdd_df.map(lambda l: ((l[5].year, quarter(l[5])), l[11]))
yearquarter_sales = each_quarter.reduceByKey(add).sortBy(lambda x: x[0])
yearquarter_sales.collect()
```

 # B.15. Compare the quarterly sales of each country of the particular year.

```
import math
def quarter(date):
    return math.ceil(date.month/3)

iyear = 2014
country_quarterly = rdd_df.map(lambda l: ((l[5].year, l[1], quarter(l[5])), l[11]))
cyearlyquarter_sales = country_quarterly.reduceByKey(add).sortBy(lambda x: x[0])
cyearlyquarter_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: ((x[0][1], x[0][2]),
x[1])).take(20)
```

```
1  df.printSchema()
```
✓ <1 sec - Command executed in 177 ms by gangothrigadige985 on 1

```
root
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Item Type: string (nullable = true)
 |-- Sales Channel: string (nullable = true)
 |-- Order Priority: string (nullable = true)
 |-- Order Date: string (nullable = true)
 |-- Order ID: integer (nullable = true)
 |-- Ship Date: string (nullable = true)
 |-- Units Sold: integer (nullable = true)
 |-- Unit Price: double (nullable = true)
 |-- Unit Cost: double (nullable = true)
 |-- Total Revenue: double (nullable = true)
 |-- Total Cost: double (nullable = true)
 |-- Total Profit: double (nullable = true)
```

```
1  from pyspark.sql.functions import *
```
✓ <1 sec - Command executed in 192 ms by gangothrigadige985 on 12:38:25 AM, 6/12/23

```
1  spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
```
✓ <1 sec - Command executed in 172 ms by gangothrigadige985 on 12:38:26 AM, 6/12/23

```
DataFrame[key: string, value: string]
```

```
1  retail_df = df.withColumn('order date', to_date(col('Order Date'), 'MM/dd/yyyy')).withColumn('ship date', to_date(col('Ship Date'), 'MM/dd/yyyy'))
```
✓ <1 sec - Command executed in 170 ms by gangothrigadige985 on 12:38:26 AM, 6/12/23

```
1  retail_df.printSchema()
```

✓ <1 sec - Command executed in 199 ms by gangothrigac

```
root
 |-- Region: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Item Type: string (nullable = true)
 |-- Sales Channel: string (nullable = true)
 |-- Order Priority: string (nullable = true)
 |-- order date: date (nullable = true)
 |-- Order ID: integer (nullable = true)
 |-- ship date: date (nullable = true)
 |-- Units Sold: integer (nullable = true)
 |-- Unit Price: double (nullable = true)
 |-- Unit Cost: double (nullable = true)
 |-- Total Revenue: double (nullable = true)
 |-- Total Cost: double (nullable = true)
 |-- Total Profit: double (nullable = true)
```

```
1  display(retail_df.limit(10))
```

✓ <1 sec - Command executed in 1 sec 108 ms by gangothrigadige985 on 12:38:28 AM, 6/12/23

Job execution Succeeded **Spark** 2 executors 8 cores View in monitoring Open

View Table Chart Export results

Region	Country	Item Type	Sales Channel	Order Priority	order date
Central America and the Caribbe...	Antigua and Barbuda	Baby Food	Online	M	2013-12-20
Central America and the Caribbe...	Panama	Snacks	Offline	C	2010-07-05
Europe	Czech Republic	Beverages	Offline	C	2011-09-12
Asia	North Korea	Cereal	Offline	L	2010-05-13
Asia	Sri Lanka	Snacks	Offline	C	2015-07-20
Middle East and North Africa	Morocco	Personal Care	Offline	L	2010-11-08
Australia and Oceania	Federated States of Micronesia	Clothes	Offline	H	2011-03-28
Europe	Bosnia and Herzegovina	Clothes	Online	M	2013-10-14
Middle East and North Africa	Afghanistan	Clothes	Offline	M	2016-08-27

```
1  retail_df.createOrReplaceTempView("retail_table")
```

✓ <1 sec - Command executed in 183 ms by gangothrigadige985 on 12:38:28 AM, 6/12/23

```
1  products_with_a_occurances=spark.sql("SELECT distinct `item type` FROM retail_table WHERE LOWER(`item type`) LIKE '%a%a%' ")
```

✓ <1 sec - Command executed in 170 ms by gangothrigadige985 on 12:38:28 AM, 6/12/23

```
1  products_with_a_occurances.show()
```

✓ <1 sec - Command executed in 600 ms by gangothrigadige985 on 1:16:14 AM, 6/12/23

> **Job execution** Succeeded **Spark** 2 executors 8 cores

```
+-------------+
|    item type|
+-------------+
|Personal Care|
+-------------+
```

```
yearly_sales = retail_df.groupBy(year("Order Date").alias("Year")).agg(sum("Total Revenue").alias("Sales")).orderBy("Year")
```

<1 sec - Command executed in 171 ms by gangothrigadige985 on 1:16:24 AM, 6/12/23

```
adls_account_name = "retailsales123"
adls_container = "source"
adls_folder = "output"
# adls_output_path = f"abfs://{adls_account_name}.blob.core.windows.net/{adls_container}/{adls_folder}/products_with_a_occurances.csv"
adls_output_path = r'abfs://source@retailsales123.dfs.core.windows.net/output/yearly_sales'

# r'abfs://retailsales123.blob.core.windows.net/source/output/products_with_a_occurances.csv'

# Save the DataFrame to ADLS
yearly_sales.write \
    .mode("overwrite") \
    .csv(adls_output_path)
```

A.9. Display the country with highest sale (Using pyspark)

```
1  from operator import add
2  country_order = rdd_df.map(lambda l: (l[1], l[11]))
3  country_revenue = country_order.reduceByKey(add).sortBy(lambda x: x[1], ascending=False)
4  countrywise_sales = spark.createDataFrame(country_revenue, ['country', 'sales'])
5  countrywise_sales.show()
```

✓ 1 sec - Command executed in 1 sec 161 ms by gangothrigadige985 on 12:46:41 AM, 6/12/23

> **Job execution** Succeeded **Spark** 2 executors 8 cores

```
+--------------------+--------------------+
|             country|               sales|
+--------------------+--------------------+
|              Rwanda| 6.039873958999999E7|
|             Myanmar|       5.883846785E7|
|         South Korea| 5.7434355429999999E7|
|               Ghana| 5.627138265000001E7|
|               Niger| 5.529821127999998E7|
|             Grenada| 5.498818455000001E7|
|Republic of the C...| 5.437980838999998E7|
```

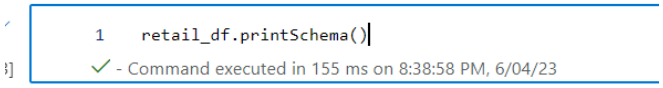
```
adls_account_name = "retailsales123"
adls_container = "source"
adls_folder = "output"
adls_output_path =  r'abfs://source@retailsales123.dfs.core.windows.net/output/countrywise_sales'

# Save the DataFrame to ADLS
countrywise_sales.write \
    .mode("overwrite") \
    .csv(adls_output_path)
```

```
%%pyspark
df = spark.read.load('abfss://retaildb@ecomfiles1.dfs.core.windows.net/dbo.retail.csv',
format='csv'
## If header exists uncomment line below
, header=True,inferSchema=True
)
display(df.limit(10))
```

```
from pyspark.sql.functions import *

spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
retail_df = df.withColumn('order_date', to_date(col('order_date'),
'MM/dd/yyyy')).withColumn('ship_date', to_date(col('ship_date'), 'MM/dd/yyyy'))
```

```
retail_df.printSchema()
```

```
root
 |-- region: string (nullable = true)
 |-- country: string (nullable = true)
 |-- item_type: string (nullable = true)
 |-- sales_channel: string (nullable = true)
 |-- order_priority: string (nullable = true)
 |-- order_date: date (nullable = true)
 |-- order_id: integer (nullable = true)
 |-- ship_date: date (nullable = true)
 |-- units_sold: integer (nullable = true)
 |-- unit_price: double (nullable = true)
 |-- unit_cost: double (nullable = true)
 |-- total_revenue: double (nullable = true)
 |-- total_cost: double (nullable = true)
 |-- total_profit: double (nullable = true)
```




```
root
 |-- region: string (nullable = true)
 |-- country: string (nullable = true)
 |-- item_type: string (nullable = true)
 |-- sales_channel: string (nullable = true)
 |-- order_priority: string (nullable = true)
 |-- order_date: date (nullable = true)
 |-- order_id: integer (nullable = true)
 |-- ship_date: date (nullable = true)
 |-- units_sold: integer (nullable = true)
 |-- unit_price: double (nullable = true)
 |-- unit_cost: double (nullable = true)
 |-- total_revenue: double (nullable = true)
 |-- total_cost: double (nullable = true)
 |-- total_profit: double (nullable = true)
```

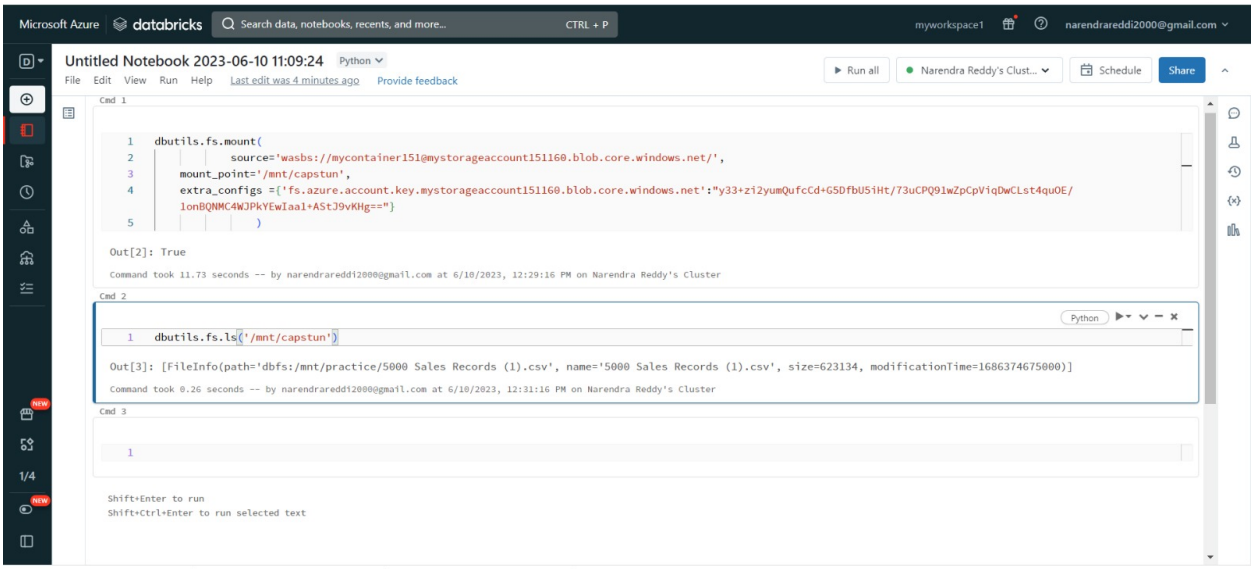
```
retail_df.createOrReplaceTempView("retail_table")
products_with_a_occurances=spark.sql("SELECT distinct item_type FROM retail_table WHERE
LOWER(item_type) LIKE '%a%a%' ")
```

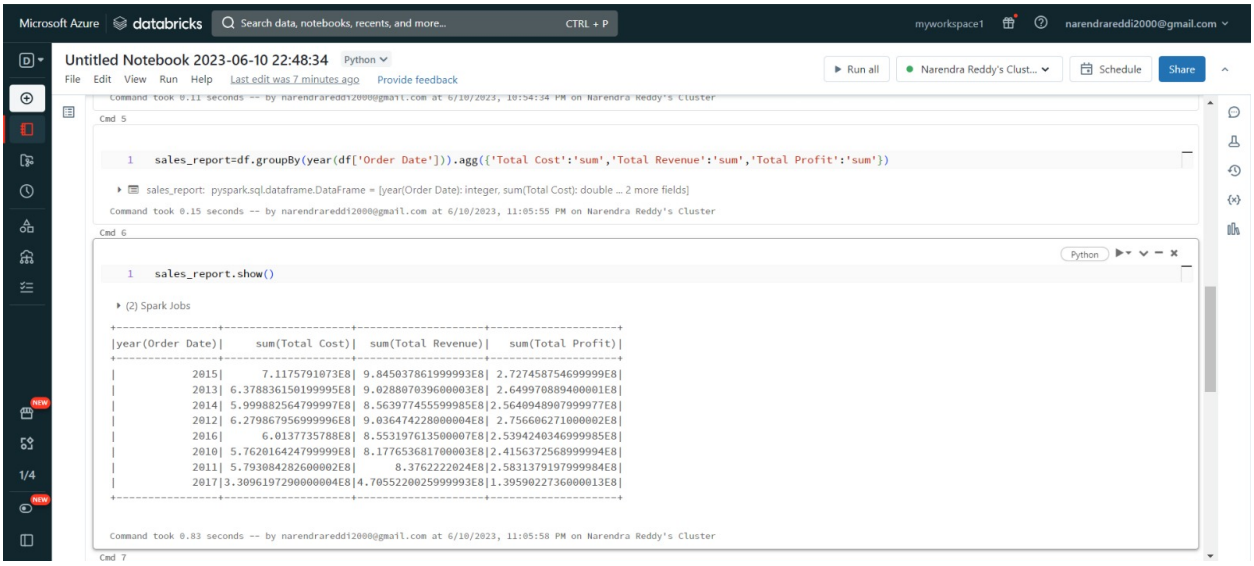
```
1    products_with_a_occurances.show()
```

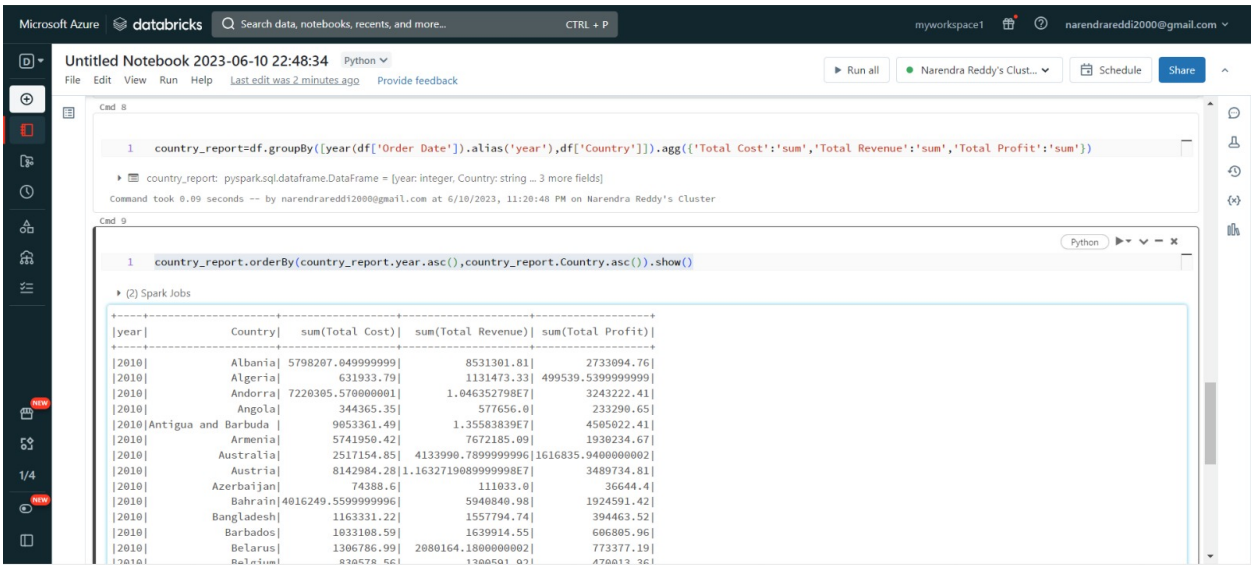
✓ - Command executed in 548 ms on 8:53:59 PM, 6/04/23

```
+-------------+
|    item_type|
+-------------+
|Personal Care|
+-------------+
```

Databricks Workspace








```
In [0]: double_a=df.filter(df['Item Type'].like('%a%a%'))
```

```
In [0]: double_aa=double_a.select(double_a['Item Type'])
```

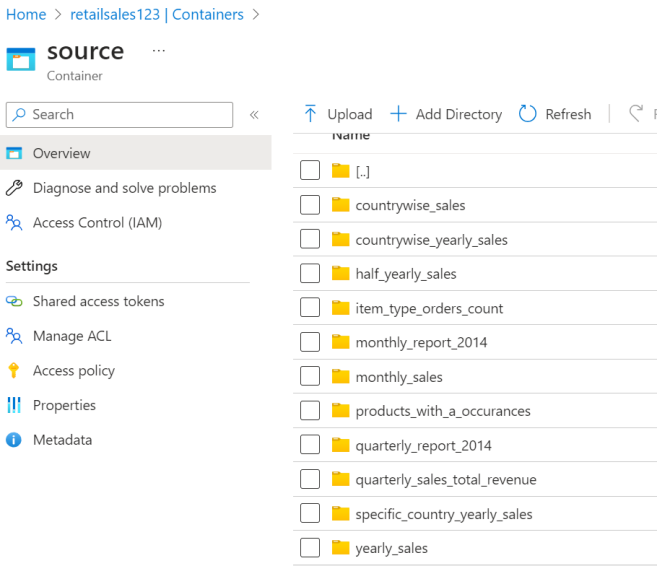
```
In [0]: double_aa.distinct().show()
```

```
+-------------+
|    Item Type|
+-------------+
|Personal Care|
+-------------+
```

Data Storage

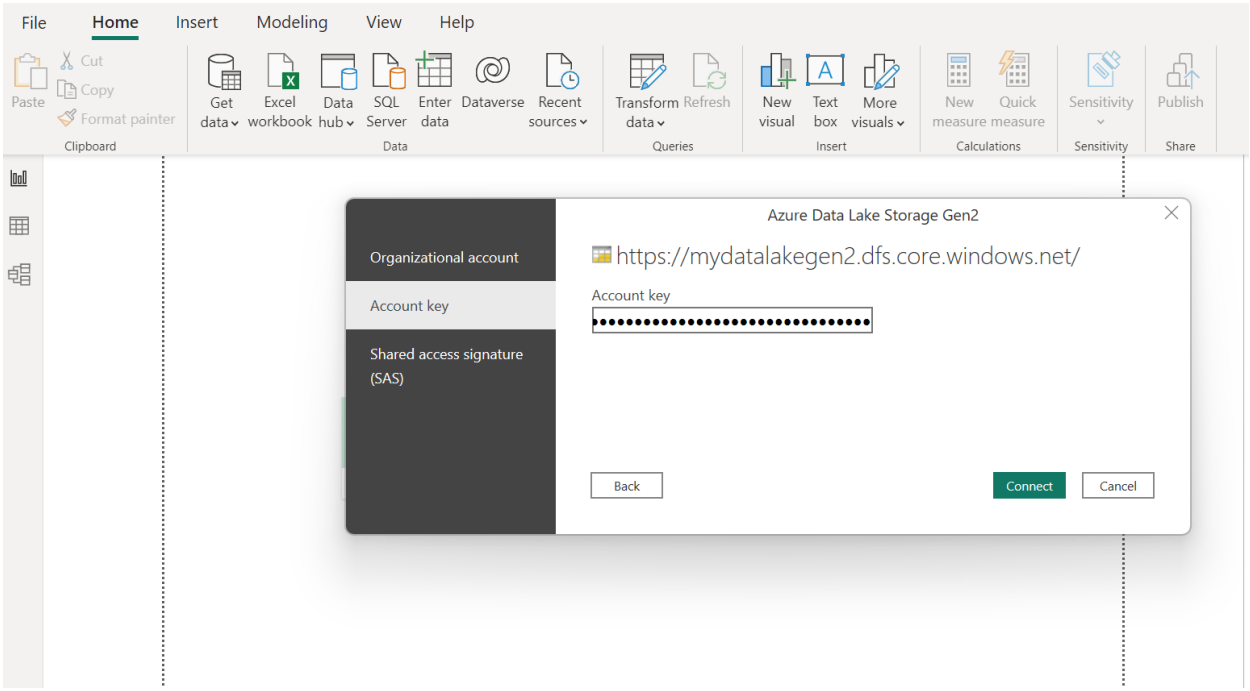
The output of the analysis queries were written into a container inside the ADLS itself

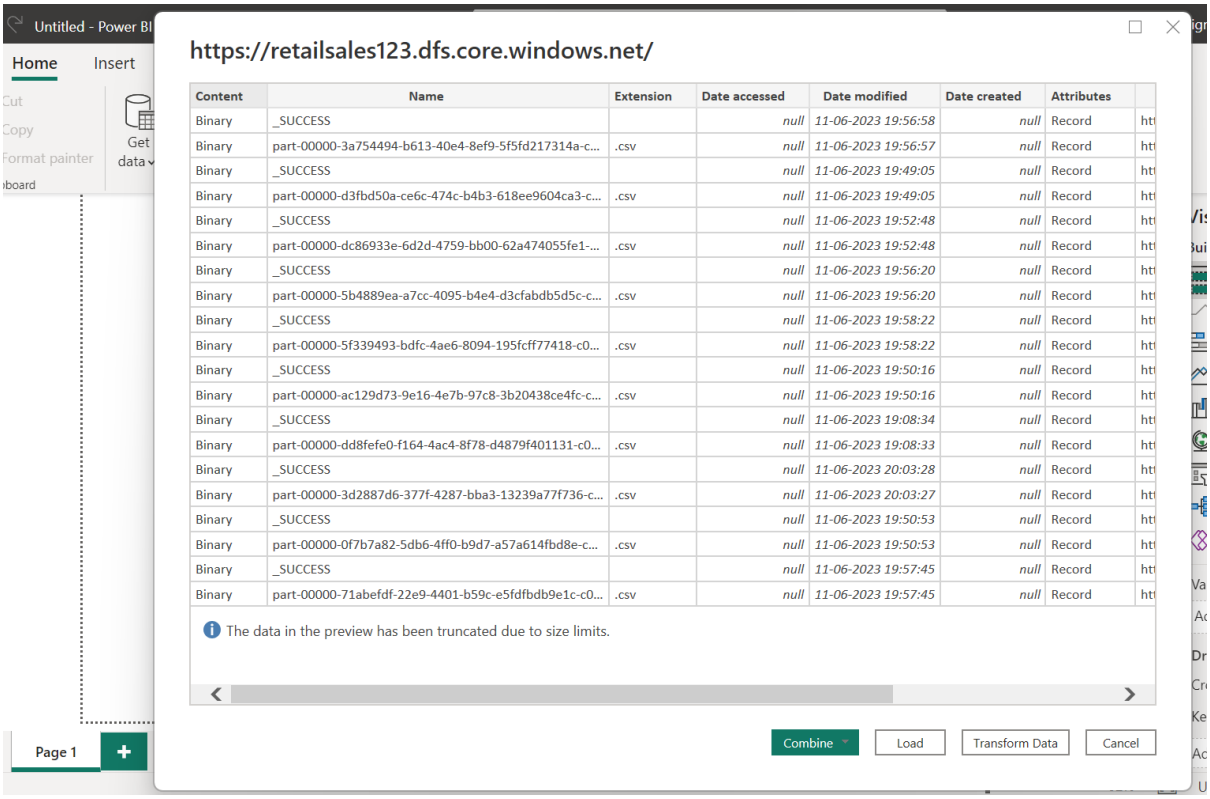
```
adls_account_name = "retailsales123"
adls_container = "source"
adls_folder = "output"
adls_output_path =  r'abfs://source@retailsales123.dfs.core.windows.net/output/countrywise_sales'

# Save the DataFrame to ADLS
countrywise_sales.write \
    .mode("overwrite") \
    .csv(adls_output_path)
```



Data Visualization and Reporting

ADLS to Power BI Connection





Power BI Dashboard