**Project Title: Retail Sales Store Forecasting**

1. **Finding and Analyzing Data:**

Analysis is done on Orders and details file which as data like

[Order\_ID]

[Order\_Date]

[CustomerName]

[State]

[City]

[Amount]

[Profit]

[Quantity]

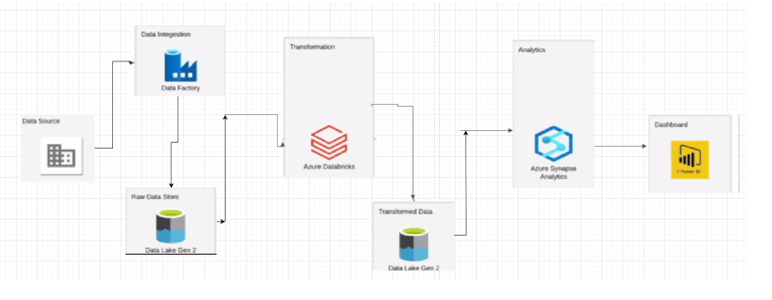
[Category]

[Sub-Category]

[PaymentMode]

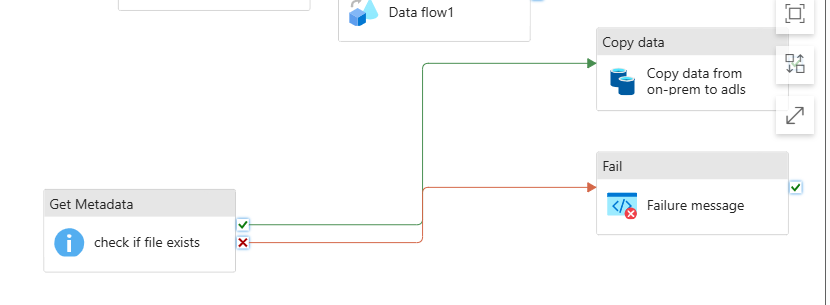
Orders.csv file we get through on-prem and details.csv file from HTTPS.

1. **Architectural Diagram: (Use tools like** [**https://draw.io/**](https://draw.io/)**)**



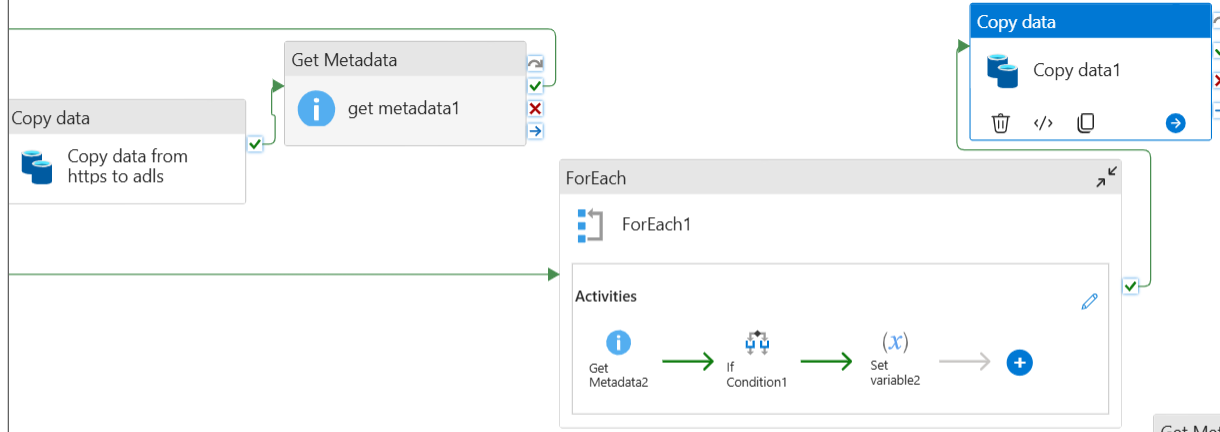
**3.Data Pipeline Creation:**

* For Orders.csv , To copy this file from On-prem to ADLS I created a Self hosted integration run time .
* Before copying the file using Copy activity, we check whether the table exists or not using Getmetadata if it’s success the file gets copied from On-Prem to ADLS Bronze container.
* If it fails it gives us a message saying the file doesn’t exist.

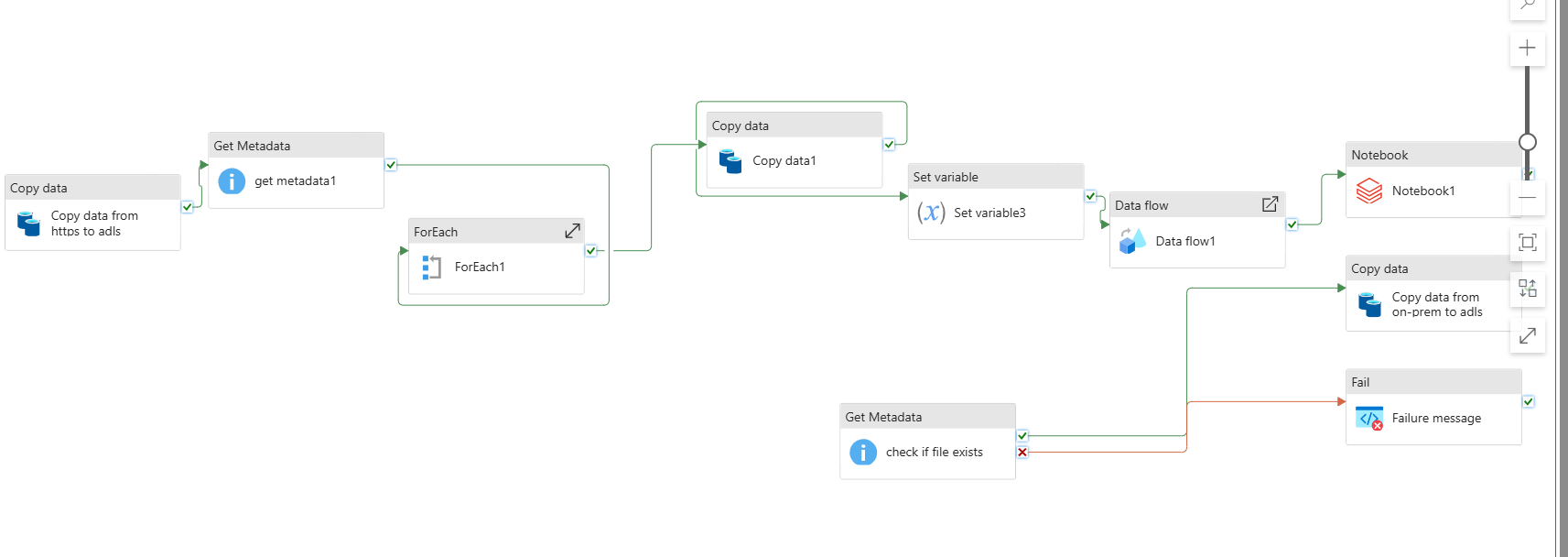
****

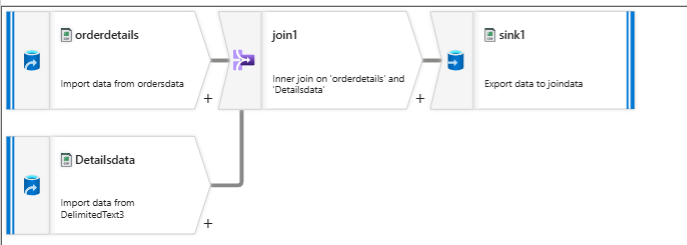
* And other file Details.csv we copy the file from HTTPS to ADLS Bronze container.
* This file we want to take the last modified file everyday and process it.
* For that once the copy activity is successful,we get child items in the path by using Get Metadata , Once its success use For each activity to process for each file , In activities add Get metadata again to get itemname and last modified for each file. Create 2 variables ‘latestfilename’and'previousmodifieddate' give default value.
* Then use If condition with this expression

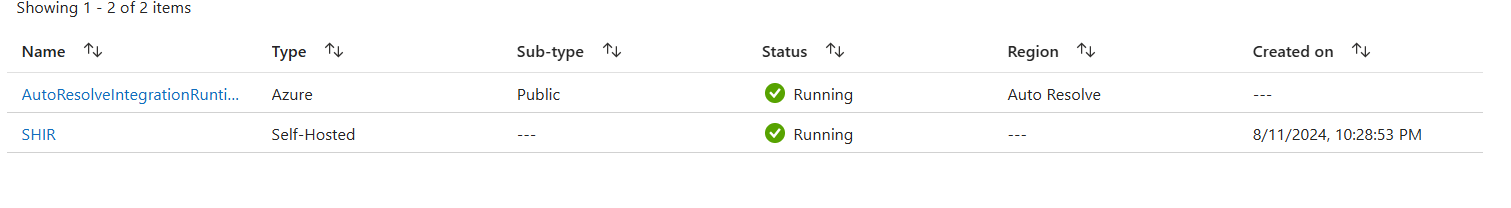
@greater(formatDateTime(activity('Get Metadata2').output.lastModified,'yyyyMMddHHmmss' ), formatDateTime(variables('previousmodifieddate'),'yyyyMMddHHmmss')) to get recent last modified



* Next in the set variable we assign 'previousmodifieddate' to last modified output.
* Next we copy activity to copy data from detailsdata folder to lastmodified folder , this folder contains only last modified file.
* We used parameters to set dynamic filenames.
* Next we used dataflow to join orders and details data based on order ID column. And copy the joined data using copy activity to silver container.

****





**4.Transformation and Analytics:**

* Creating an Azure Data Lake Storage Gen2 storage account and containers named “Bronze” ,”Silver and “Gold” in ADLS Gen 2 storage.
* Creating an Azure Databricks workspace from Azure portal and creating a Spark cluster within that Azure Databricks workspace.
* Open a new notebook within Azure databricks for data cleaning and transformation.
* connecting one storage location (ADLS Gen2) to another within Azure Databricks using service principle. This makes data stored in ADLS Gen2 accessible within the Databricks environment.
* service\_credential = dbutils.secrets.get(scope="secretscope", key="secretid")
* spark.conf.set("fs.azure.account.auth.type.salesprojectstorage.dfs.core.windows.net", "OAuth")
* spark.conf.set("fs.azure.account.oauth.provider.type.salesprojectstorage.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
* spark.conf.set("fs.azure.account.oauth2.client.id.salesprojectstorage.dfs.core.windows.net", "cfdf2f4b-0dd6-4003-85c6-42f61c6e1fe7")
* spark.conf.set("fs.azure.account.oauth2.client.secret.salesprojectstorage.dfs.core.windows.net", service\_credential)
* spark.conf.set("fs.azure.account.oauth2.client.endpoint.salesprojectstorage.dfs.core.windows.net", "https://login.microsoftonline.com/80d8b5a9-a2e9-4997-a38b-d0e9c44ef5d6/oauth2/token")
* Using Pyspark, I got only last modified file dynamically from path .

schema = StructType([

StructField('path', StringType()),

StructField('name', StringType()),

StructField('size', IntegerType()),

StructField('modificationTime', LongType())

])

directory\_path = "abfs://silver@salesprojectstorage.dfs.core.windows.net"

file\_list = dbutils.fs.ls(directory\_path)

df = spark.createDataFrame(data=file\_list, schema=schema)

df = df.select('path', 'name', 'size', (df.modificationTime/1000).cast('timestamp').alias('UpdatedTime'))

df\_latestfile = df.select('path', 'name', 'size', 'UpdatedTime').orderBy(df.UpdatedTime.desc()).limit(1)

* To get the path from the dataframe where last modified file is stored we use collect().

# Extract the file path from df\_latestfile

file\_path = df\_latestfile.select('path').collect()[0][0]

* Using PySpark , I perfomed data cleaning techniques which were removal of unnecessary columns and ensuring that all the columns have the correct datatype .

Schema = StructType([

    StructField("Order\_ID", StringType()),

    StructField("Order\_Date", DateType()),

    StructField("CustomerName", StringType()),

    StructField("State", StringType()),

    StructField("City", StringType()),

    StructField("Order ID", StringType()),

    StructField("Amount", IntegerType()),

    StructField("Profit", IntegerType()),

    StructField("Quantity", IntegerType()),

    StructField("Category", StringType()),

    StructField("Sub-Category", StringType()),

    StructField("PaymentMode", StringType()),

])

df\_readingfile = spark.read.csv(file\_path, header=True, schema=Schema)

df\_readingfile = df\_readingfile.drop("Order ID")

df\_readingfile = df\_readingfile.withColumn("Timestamp", current\_timestamp()) \

                               .withColumn("quarter", quarter(df\_readingfile['Order\_Date']))\

                               .withColumn("insert\_timestamp", date\_format(col("Timestamp"), "yyyy-MM-dd HH:mm:ss"))

* Next step , I performed aggregation on the cleaned data using Pyspark .

df\_profit\_Bymonth = df\_readingfile.groupBy(date\_format(df\_readingfile['Order\_Date'],"MMMM").alias("month")).agg(sum("Profit").alias("Total profit"))

df\_profit\_Bymonth.show()

df\_totalamount\_Bystate = df\_readingfile.groupBy("State").agg(sum("Amount").alias("Total Amount"))

df\_totalamount\_Bystate.show()

df\_totalquantity\_paymentmode = df\_readingfile.groupBy("PaymentMode").agg(sum("Quantity").alias("Total Quantity"))

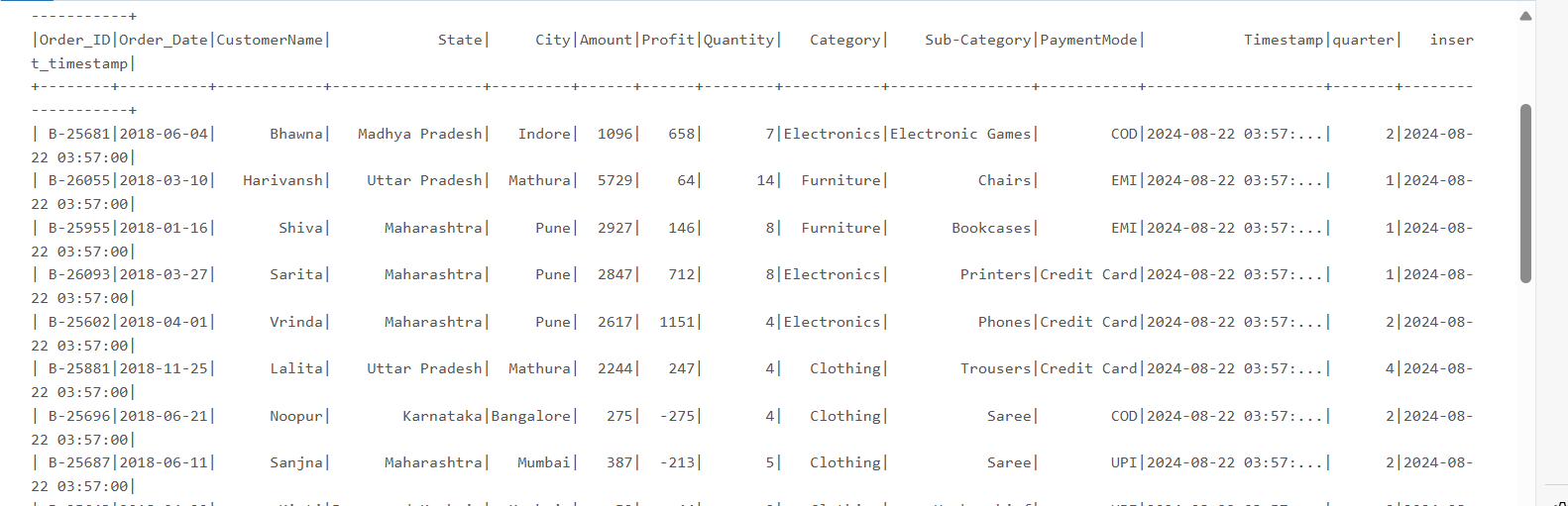
* .And I uploaded structured data to Gold container. This Gold container holds structured format data which is suitable for reporting, visualization.

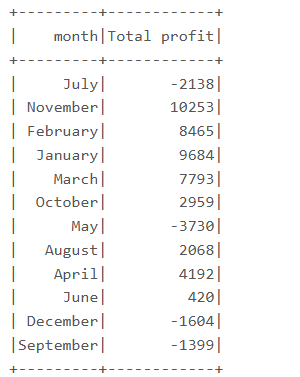
df\_readingfile.write.mode("append").parquet("abfs://gold@salesprojectstorage.dfs.core.windows.net/transformedfile")

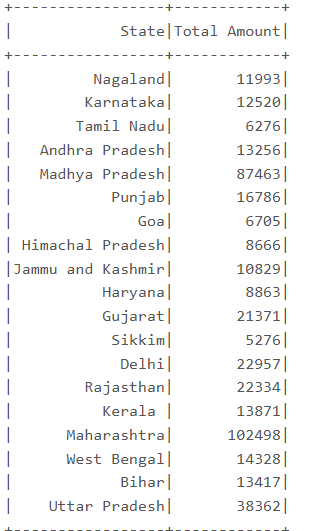
df\_profit\_Bymonth.write.mode("append").parquet("abfs://gold@salesprojectstorage.dfs.core.windows.net/profitBymonthtransformedfile")

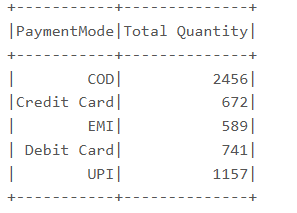
df\_totalamount\_Bystate.write.mode("append").parquet("abfs://gold@salesprojectstorage.dfs.core.windows.net/totalAmountByStatetransformedfile")

df\_totalquantity\_paymentmode.write.mode("append").parquet("abfs://gold@salesprojectstorage.dfs.core.windows.net/totalquantityBypaymentmodetransformedfile")



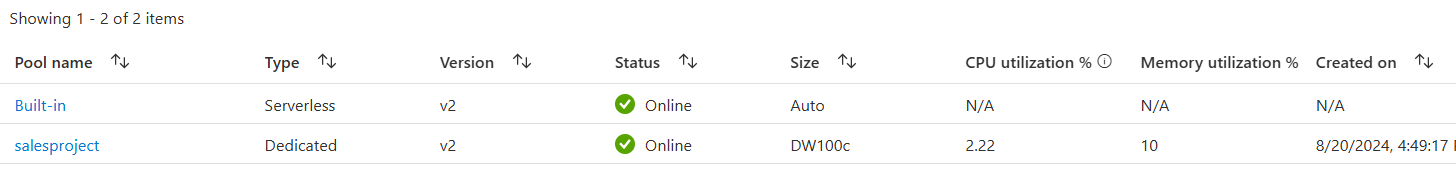






Synapse:

* Created Azure Synapse workspace , In there created dedicated SQL pool .
* And create External tables to pull data from ADLS Gold container , On which we can perform analysis .



CREATE DATABASE SCOPED CREDENTIAL sasToken2

WITH IDENTITY ='SHARED ACCESS SIGNATURE',

SECRET='sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupyx&se=2024-08-21T07:01:59Z&st=2024-08-20T23:01:59Z&spr=https&sig=xDgcanPWtv%2BB8ZU4qYc2xlAJ7linLbv7i2ED4OQjr2E%3D'

CREATE EXTERNAL DATA SOURCE retailprojectsrc2

WITH

(

LOCATION='https://salesprojectstorage.blob.core.windows.net/gold',

CREDENTIAL=sasToken2

)

CREATE EXTERNAL FILE FORMAT parquetFileFormat2 WITH

(

FORMAT\_TYPE=PARQUET,

DATA\_COMPRESSION='org.apache.hadoop.io.compress.SnappyCodec'

)

CREATE EXTERNAL TABLE Retailsalesprojectdemooo

(

[Order\_ID] varchar(200),

[Order\_Date] varchar(300),

[CustomerName] varchar(100),

[State] varchar(100),

[City] varchar(100),

[Amount] int,

[Profit] int,

[Quantity] int,

[Category] varchar(300),

[Sub-Category] varchar(1000),

[PaymentMode] varchar(2000),

[Timestamp] varchar,

[quarter] int)

WITH (

LOCATION='/transformedfile',

DATA\_SOURCE=retailprojectsrc2,

FILE\_FORMAT=parquetFileFormat2

)

CREATE EXTERNAL TABLE profitBymonthtransformed

(

[month] varchar(200),

[Total profit] int

WITH (

LOCATION='/profitBymonthtransformedfile',

DATA\_SOURCE=retailprojectsrc2,

FILE\_FORMAT=parquetFileFormat2

)

CREATE EXTERNAL TABLE totalAmountByStatetransformed

(

[State] varchar(200),

[Total Amount] int

WITH (

LOCATION='/totalAmountByStatetransformedfile',

DATA\_SOURCE=retailprojectsrc2,

FILE\_FORMAT=parquetFileFormat2

)

CREATE EXTERNAL TABLE totalquantityBypaymentmodetransformed

(

[PaymentMode] varchar(200),

[Total Quantity] int

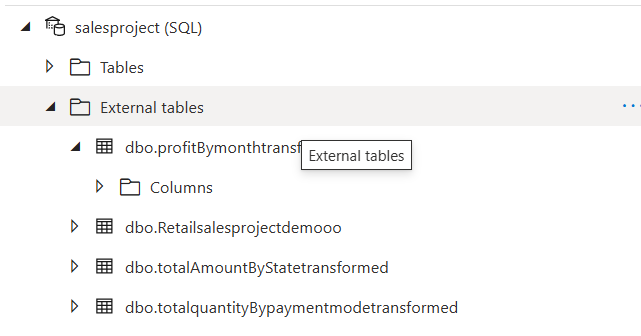
WITH (

LOCATION='/totalquantityBypaymentmodetransformedfile',

DATA\_SOURCE=retailprojectsrc2,

FILE\_FORMAT=parquetFileFormat2

)



**5.Visualizations:**

* I Used Power BI to create bar charts ,pie charts ,cards, slicer depicting trends by connecting to azure Synapse analytics using Dedicated SQL end point URL.
* These visualizations offer insightful representations of the data.



**6.Conclusion:**

By following these steps, a robust ETL pipeline is built using Azure Databricks, Azure Data Factory and ADLS Gen2. This setup allows efficient data processing, transformation, and storage, facilitating advanced analytics and reporting using power BI and Azure Synapse.