# **Project Title: Retail Store Sales**

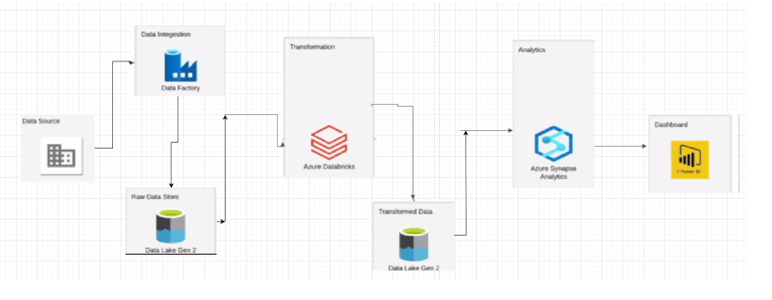
**Problem Statement:** Retail store sales is the dataset that provides clear and concise overview of the store’s sales performance enabling the owner to gain actionable insights.

1. **Finding and Analysing Data:**

In this dataset, we have 2 csv files orders and details on which analysis is done. I have used SSMS as on premises database and created Retail database and stored Orders.csv file. Moreover, details.csv file is available on my GitHub i.e HTTPS. These two files contain following data:

* [Order\_ID]
* [Order\_Date]
* [CustomerName]
* [State]
* [City]
* [Amount]
* [Profit]
* [Quantity]
* [Category]
* [Sub-Category]
* [PaymentMode]

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



* Created all the resources that will be used in the project.

A screenshot of a computer

Description automatically generated

* Created self-hosted integration runtime to access on premises database.

A screenshot of a computer

Description automatically generated

* To establish the connection between Azure and SQL Server, I have the script for creation of credentials. Stored these credentials in Azure key vault.

A screenshot of a computer

Description automatically generated

**3.Data Pipeline Creation:**

* To copy orders.csv file from On-prem to ADLS, I have used Self-hosted integration run time.
* Before copying the file using Copy activity, we check whether the table exists or not by using Get metadata activity. If Get metadata activity is success, I have added If condition activity where file gets copied from On-Prem to ADLS Bronze container under true branch.
* If it fails it gives us a message saying File orders.csv does not exist in SQL Server under false branch.

A screenshot of a computer

Description automatically generated

* And other file Details.csv we copy the file from HTTPS to ADLS Bronze container.
* For this, I have added get metadata to check whether bronze container is available or not. Then added If condition and added 2 set variable activities in it.
* If bronze container is found, set variable activity which is under true case will give a message “found bronze container. Proceed with copy activity”.
* If bronze container is not found, set variable activity which is under false case will give a message “can't found bronze container. Not proceeding with copy activity”.
* Then, I have added foreach activity where it checks for childItems of bronze container: @activity('check for bronze container'). output.childItems
* Within that foreach, added copy activity which completes copying data from HTTPS to ADLS Bronze container.

A screenshot of a computer

Description automatically generated

* Once the copy activity is successful, I have used dataflow to join orders and details data based on order ID column. And copy the joined data using copy activity to silver container.

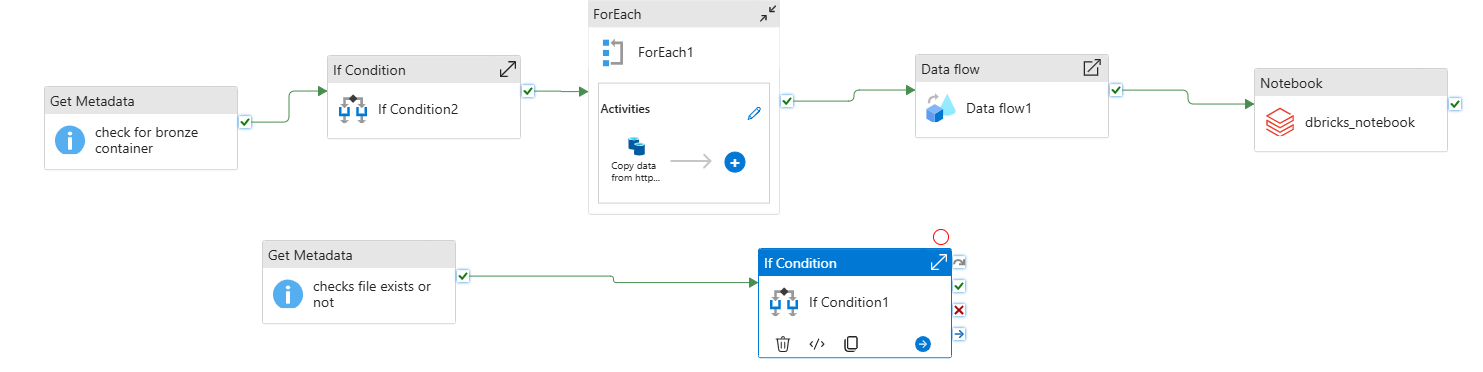
A screenshot of a computer screen

Description automatically generated

* In the Azure Data Factory Data Flow, data was sunk to the ADLS Gen2 storage in the silver container. Following this, a Notebook activity was added to the pipeline. A linked service was created to securely access the Azure Key Vault to retrieve the access token. The path for the notebook created in Azure Databricks was provided, enabling the execution of the notebook from within the Data Factory pipeline.

A screenshot of a computer

Description automatically generated



**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.
* **Bronze Container:** Raw data was uploaded into this container.
* **Silver Container**: orders and details data was joined and stored in silver container. Azure data bricks notebook accessed data from the silver container for cleaning and transformation using the Databricks Spark Cluster.
* **Gold Container:** The cleaned and transformed data was saved in the gold container for further analysis or reporting.
* Creating an Azure Databricks workspace from Azure portal and creating a Spark cluster within that Azure Databricks workspace.
* Opened a new notebook within Azure data bricks for data cleaning and transformation.
* A secure connection was established between the ADLS Gen2 storage account and the Azure Databricks workspace using a Service Principal. Service Principal was configured with the necessary permissions to allow access to storage containers from Databricks.

client\_secret=dbutils.secrets.get(scope='Demo-scope',key='secretID')

client\_id=dbutils.secrets.get(scope='Demo-scope',key='clientID')

tenant\_id=dbutils.secrets.get(scope='Demo-scope',key='tenantID1')

spark.conf.set("fs.azure.account.auth.type.kpprojejectadlsstorage.dfs.core.windows.net", "OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type.kpprojejectadlsstorage.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id.kpprojejectadlsstorage.dfs.core.windows.net",client\_id )

spark.conf.set("fs.azure.account.oauth2.client.secret.kpprojejectadlsstorage.dfs.core.windows.net", client\_secret)

spark.conf.set("fs.azure.account.oauth2.client.endpoint.kpprojejectadlsstorage.dfs.core.windows.net", f"https://login.microsoftonline.com/{tenant\_id}/oauth2/token")

* Importing required libraries, initialized a spark session and defined schema using StructType and StructField to structure the incoming data.
* Reading data from csv file and displaying the data.

from pyspark.sql import \*

from pyspark.sql.types import \*

from pyspark.sql.functions import \*

spark = SparkSession.builder.appName("ReadData").getOrCreate()

schema =  StructType([StructField('Order\_ID', StringType()),

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

                        StructField('CustomerName', StringType()),

                        StructField('State', StringType()),

                      StructField('City', StringType()),

                      StructField('Amount', IntegerType()),

                      StructField('Profit', IntegerType()),

                      StructField('Quantity', IntegerType()),

                      StructField('Category', StringType()),

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

                      StructField('PaymentMode', StringType())

    ])

df = spark.read.csv(

    "abfss://silver@kpprojejectadlsstorage.dfs.core.windows.net/join.csv",

    schema=schema,

    header=True  # Ensures the first row is treated as the header

);

df.show()

* Basic data cleaning and transformation steps using PySpark to prepare the data for further processing and ensure consistency and accuracy.:
* Remove Duplicates: Ensures unique entries by dropping rows with duplicate Order\_ID values.
* Handle Missing Values: Replaces null values in the Profit column with 0, and for City and State columns with "Unknown."
* Format Date Column: Converts the Order\_Date column into a consistent yyyy-MM-dd date format for easier analysis.

# Remove duplicate rows based on Order\_ID

df = df.dropDuplicates(["Order\_ID"])

# Replace null or missing values in specific columns

df = df.fillna({"Profit": 0, "City": "Unknown", "State": "Unknown"})

# Ensure Order\_Date is in a consistent format

from pyspark.sql.functions import to\_date

df\_transformed = df.withColumn("Order\_Date", to\_date(df["Order\_Date"], "yyyy-MM-dd"))

df\_transformed.show()

A screenshot of a computer

Description automatically generated

* Adding 3 derived columns to enhance dataset:
* **Profit\_Margin:** Calculates the profit percentage for each order by dividing Profit by Amount (multiplied by 100). If Amount is 0, the margin is set to 0.
* **High\_Value\_Order:** Identifies orders with an Amount greater than 1000 as "Yes"; otherwise, they are marked as "No."
* **Category\_Group:** Combines Category and Sub-Category into a single column, separated by " - " for better grouping and analysis.

#Adding Derived Columns like Profit\_Margin, High\_Value\_Order, Category\_Group

df\_transformed = df\_transformed.withColumn("Profit\_Margin", when(col("Amount") != 0, (col("Profit") / col("Amount")) \* 100).otherwise(0))

df\_transformed = df\_transformed.withColumn("High\_Value\_Order", when(col("Amount") > 1000, "Yes").otherwise("No"))

transformed = df\_transformed.withColumn("Category\_Group", concat\_ws(" - ", col("Category"), col("Sub-Category")))

transformed.show()

* Calculating the Total Sales and Total Profit for each state by grouping the dataset based on the State column.

#Total Sales and Profit by State

total\_sales\_by\_state = transformed.groupBy("State").agg(

    {"Amount": "sum", "Profit": "sum"}

).withColumnRenamed("sum(Amount)", "Total\_Sales").withColumnRenamed("sum(Profit)", "Total\_Profit")

total\_sales\_by\_state.show()

A screenshot of a computer

Description automatically generated

* Identifying the most popular payment modes by grouping the data based on the Payment Mode column.

#Popular Payment Modes

payment\_mode\_summary = transformed.groupBy("PaymentMode").count().orderBy("count", ascending=False)

payment\_mode\_summary.show()

A screenshot of a computer

Description automatically generated

* Identifying the top customers based on total sales. It groups the data by CustomerName, calculates the total Amount for each customer, and orders the results in descending order of sales. This analysis highlights the customers contributing the most to the revenue.

#Top Customers by Sales

top\_customers\_by\_sales = transformed.groupBy("CustomerName").agg({"Amount": "sum"}).orderBy("sum(Amount)", ascending=False)

top\_customers\_by\_sales.show()

A screenshot of a computer

Description automatically generated

* Writing the processed and transformed data into the "Gold" container in Azure Data Lake Storage (ADLS). The transformed dataset, along with summary datasets like total\_sales\_by\_state, payment\_mode\_summary, and top\_customers\_by\_sales, are saved as CSV files in overwrite mode.

#writing data to ADLS gold container

transformed.write.mode("overwrite").csv("abfs://gold@kpprojejectadlsstorage.dfs.core.windows.net/transformedfile")

total\_sales\_by\_state.write.mode("overwrite").csv("abfs://gold@kpprojejectadlsstorage.dfs.core.windows.net/total\_sales\_by\_state")

payment\_mode\_summary.write.mode("overwrite").csv("abfs://gold@kpprojejectadlsstorage.dfs.core.windows.net/payment\_mode\_summary")

top\_customers\_by\_sales.write.mode("overwrite").csv("abfs://gold@kpprojejectadlsstorage.dfs.core.windows.net/top\_customers\_by\_sales")

A screenshot of a computer

Description automatically generated

**Synapse:**

* Created Azure Synapse workspace, in there created dedicated SQL pool.

A screenshot of a computer

Description automatically generated

* Created External tables to pull data from ADLS Gold container, On which we can perform analysis .
* External table for transformedfile:

IF NOT EXISTS (SELECT \* FROM sys.external\_file\_formats WHERE name = 'SynapseDelimitedTextFormat')

    CREATE EXTERNAL FILE FORMAT [SynapseDelimitedTextFormat]

    WITH ( FORMAT\_TYPE = DELIMITEDTEXT ,

           FORMAT\_OPTIONS (

             FIELD\_TERMINATOR = ',',

             FIRST\_ROW = 2,

             USE\_TYPE\_DEFAULT = FALSE

            ))

GO

IF NOT EXISTS (SELECT \* FROM sys.external\_data\_sources WHERE name = 'gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net')

    CREATE EXTERNAL DATA SOURCE [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net]

    WITH (

        LOCATION = 'abfss://gold@kpprojejectadlsstorage.dfs.core.windows.net',

        TYPE = HADOOP

    )

GO

CREATE EXTERNAL TABLE dbo.transformedfile (

    [Order\_ID] nvarchar(4000),

    [Order\_Date] date,

    [CustomerName] nvarchar(4000),

    [State] nvarchar(4000),

    [City] nvarchar(4000),

    [Amount] bigint,

    [Profit] bigint,

    [Quantity] bigint,

    [Category] nvarchar(4000),

    [Sub-Category] nvarchar(4000),

    [PaymentMode] nvarchar(4000),

    [Profit\_Margin] float,

    [High\_Value\_Order] nvarchar(4000),

    [Category\_Group] nvarchar(4000)

    )

    WITH (

    LOCATION = 'transformedfile',

    DATA\_SOURCE = [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net],

    FILE\_FORMAT = [SynapseDelimitedTextFormat]

    )

GO

SELECT TOP 100 \* FROM dbo.transformedfile

GO

A screenshot of a computer

Description automatically generated

* External table for top\_customers\_by\_sales:

IF NOT EXISTS (SELECT \* FROM sys.external\_file\_formats WHERE name = 'SynapseDelimitedTextFormat')

    CREATE EXTERNAL FILE FORMAT [SynapseDelimitedTextFormat]

    WITH ( FORMAT\_TYPE = DELIMITEDTEXT ,

           FORMAT\_OPTIONS (

             FIELD\_TERMINATOR = ',',

             FIRST\_ROW = 2,

             USE\_TYPE\_DEFAULT = FALSE

            ))

GO

IF NOT EXISTS (SELECT \* FROM sys.external\_data\_sources WHERE name = 'gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net')

    CREATE EXTERNAL DATA SOURCE [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net]

    WITH (

        LOCATION = 'abfss://gold@kpprojejectadlsstorage.dfs.core.windows.net',

        TYPE = HADOOP

    )

GO

CREATE EXTERNAL TABLE dbo.top\_customers\_by\_sales (

    [CustomerName] nvarchar(4000),

    [sum(Amount)] bigint

    )

    WITH (

    LOCATION = 'top\_customers\_by\_sales',

    DATA\_SOURCE = [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net],

    FILE\_FORMAT = [SynapseDelimitedTextFormat]

    )

GO

SELECT TOP 100 \* FROM dbo.top\_customers\_by\_sales

GO

A screenshot of a computer

Description automatically generated

* External table for total\_sales\_by\_state:

IF NOT EXISTS (SELECT \* FROM sys.external\_file\_formats WHERE name = 'SynapseDelimitedTextFormat')

    CREATE EXTERNAL FILE FORMAT [SynapseDelimitedTextFormat]

    WITH ( FORMAT\_TYPE = DELIMITEDTEXT ,

           FORMAT\_OPTIONS (

             FIELD\_TERMINATOR = ',',

             FIRST\_ROW = 2,

             USE\_TYPE\_DEFAULT = FALSE

            ))

GO

IF NOT EXISTS (SELECT \* FROM sys.external\_data\_sources WHERE name = 'gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net')

    CREATE EXTERNAL DATA SOURCE [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net]

    WITH (

        LOCATION = 'abfss://gold@kpprojejectadlsstorage.dfs.core.windows.net',

        TYPE = HADOOP

    )

GO

CREATE EXTERNAL TABLE dbo.total\_sales\_by\_state (

    [State] nvarchar(4000),

    [Total\_Sales] bigint,

    [Total\_Profit] bigint

    )

    WITH (

    LOCATION = 'total\_sales\_by\_state',

    DATA\_SOURCE = [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net],

    FILE\_FORMAT = [SynapseDelimitedTextFormat]

    )

GO

SELECT TOP 100 \* FROM dbo.total\_sales\_by\_state

GO

A screenshot of a graph

Description automatically generated

* External table for payment\_mode\_summary:

IF NOT EXISTS (SELECT \* FROM sys.external\_file\_formats WHERE name = 'SynapseDelimitedTextFormat')

    CREATE EXTERNAL FILE FORMAT [SynapseDelimitedTextFormat]

    WITH ( FORMAT\_TYPE = DELIMITEDTEXT ,

           FORMAT\_OPTIONS (

             FIELD\_TERMINATOR = ',',

             FIRST\_ROW = 2,

             USE\_TYPE\_DEFAULT = FALSE

            ))

GO

IF NOT EXISTS (SELECT \* FROM sys.external\_data\_sources WHERE name = 'gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net')

    CREATE EXTERNAL DATA SOURCE [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net]

    WITH (

        LOCATION = 'abfss://gold@kpprojejectadlsstorage.dfs.core.windows.net',

        TYPE = HADOOP

    )

GO

CREATE EXTERNAL TABLE dbo.payment\_mode\_summary (

    [PaymentMode] nvarchar(4000),

    [count] bigint

    )

    WITH (

    LOCATION = 'payment\_mode\_summary',

    DATA\_SOURCE = [gold\_kpprojejectadlsstorage\_dfs\_core\_windows\_net],

    FILE\_FORMAT = [SynapseDelimitedTextFormat]

    )

GO

SELECT TOP 100 \* FROM dbo.payment\_mode\_summary

GO

A screenshot of a computer

Description automatically generated

* All the external tables I have created:

A screenshot of a computer

Description automatically generated

**5.Visualizations:**

* In this project, I used Power BI to create interactive visualizations by connecting to Azure Data Lake Storage via an endpoint URL. The report includes bar charts, donut charts, and slicers to display key business trends such as profit/loss, product quantity by category and payment mode, top-performing states in sales, and top customers. Conditional formatting was applied to highlight profit/loss in bar charts, enhancing data insights. These visualizations offer insightful representations of the data.

A screenshot of a computer

Description automatically generated

The business shows fluctuating profits, with Cash on Delivery being the preferred payment method. "Furniture" is the most popular category, especially "Bookcases," while Maharashtra leads in sales. Harivansh is the highest-paying customer, contributing significantly to revenue.

**6.Conclusion:**

By implementing these steps, a powerful ETL pipeline is created using Azure Databricks, Azure Data Factory, and ADLS Gen2. This configuration enables efficient data processing, transformation, and storage, supporting advanced analytics and reporting through Power BI and Azure Synapse.