CAPSTONE PROJECT

Dataset Name – **Superstore Orders** (should already be on your laptop in csv format)

Tip: change the name of your file before you upload it to the database… make sure it does not contain any special characters like -,’; and so on.

PROJECT AIM: To build an end-to-end data engineering solution using the tools listed below

**BUSINESS REQUIREMENTS (KPIs)**

* Total Quantity, Total Sales, Total Profit
* Sales by Product Category
* Sales by City
* Sales by State
* Top 5 Purchasing Customers per year
* Average Sales per month

**DATA ENGINEERING TOOLS**

* SQL Server – On Prem Data Storage
* Azure Data Factory – Data Ingestion
* BLOB Storage – Data Storage
* Azure Databricks – Data Processing
* PYSPARK
* SPARK SQL
* Power BI – Analysis and Dashboarding

**CURRENT DATA STRUCTURE**

Data is already available on our On-prem SQL Server and it needs to be moved to the cloud and should be accessible to data analysts and end users for analytical and machine learning purposes.

**TABLES TO BE CREATED IN DATABRICKS**

* Customer Table
* Orders Table
* Sales Table -Facts
* Products Table
* Location Table

PROCESS

1. DATA ARCHITECHTURE
2. Upload the file (Superstore-Orders) to the SQL Server
3. Provision ADF, Azure Databricks and ADLS
4. Create Linked Service in ADF to move data from SQL Server to ADLS
5. Connect Databricks to storage – PySpark
6. Process data in Databricks (clean data and create new tables)
7. Create dashboard in PowerBI

SQL Server – UPLOADING YOUR FILE TO THE SQL SERVER

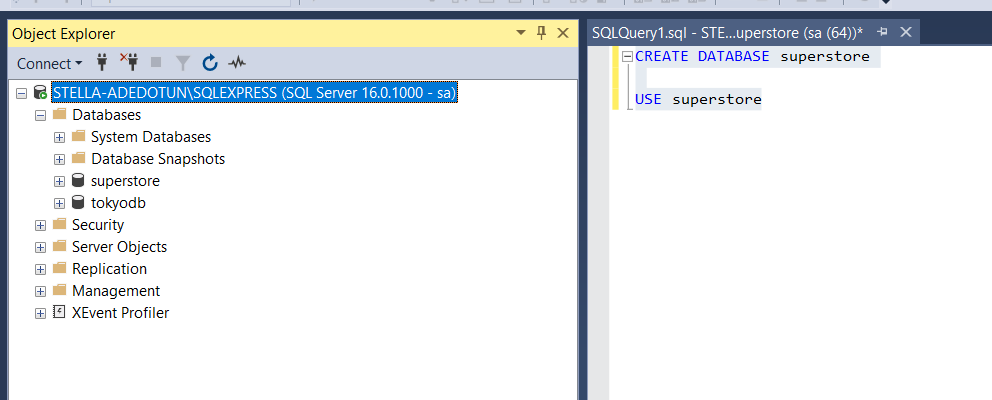
Create a database “superstore” (you can give it whatever name you want) by executing the query below in the query window:

CREATE DATABASE superstore

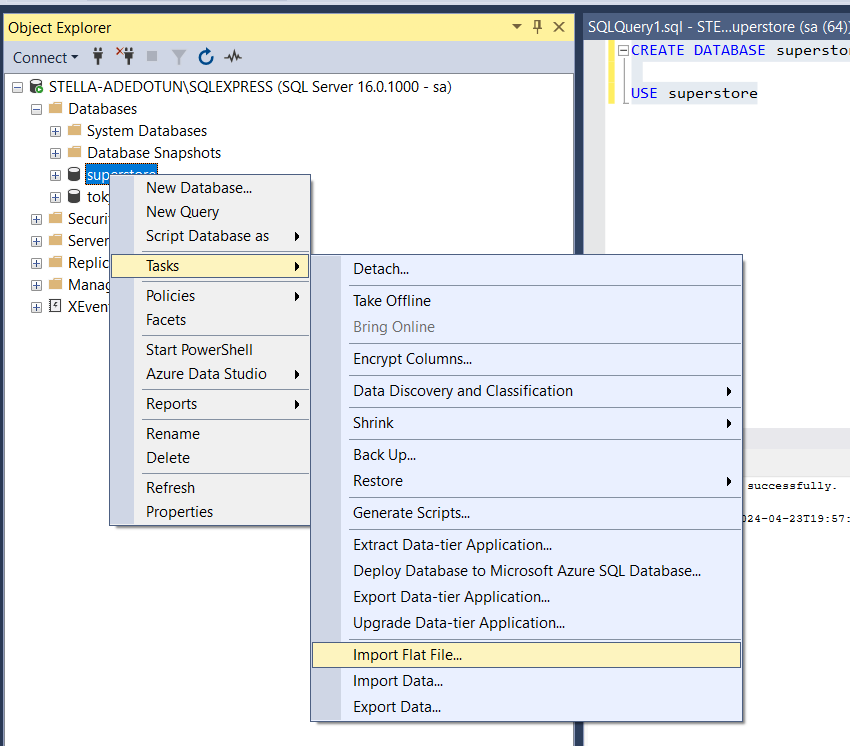
go

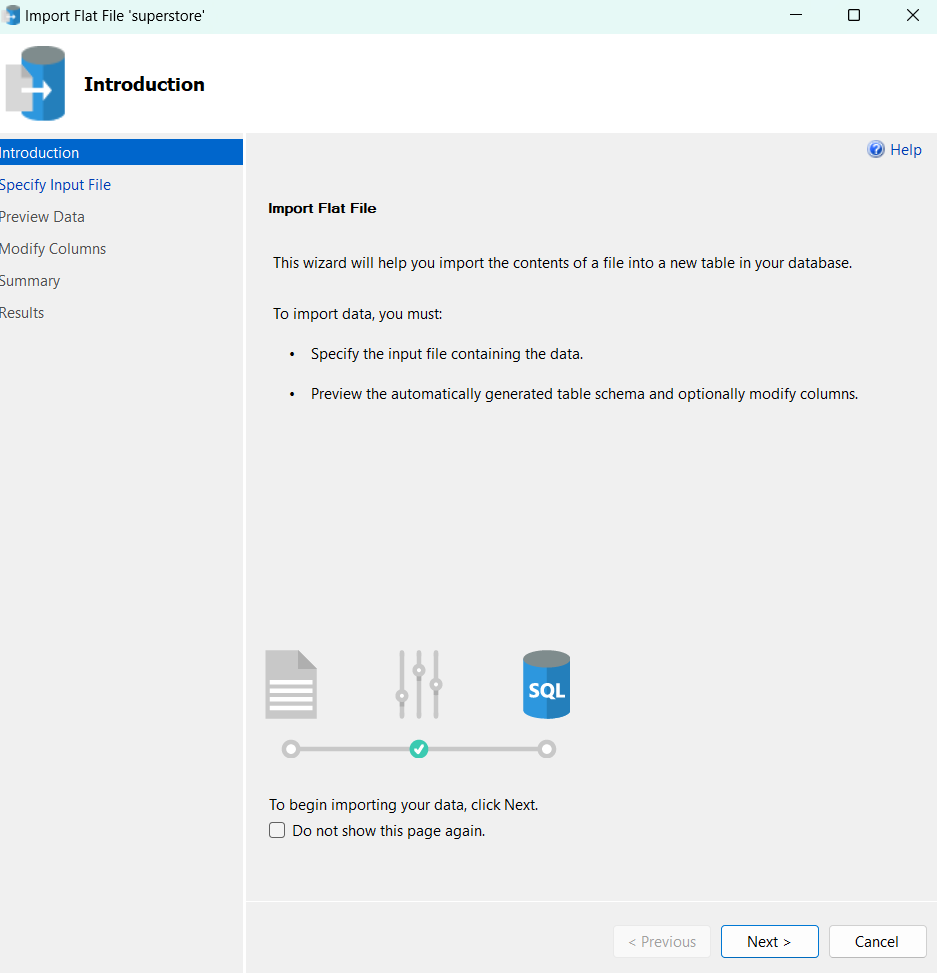
USE superstore

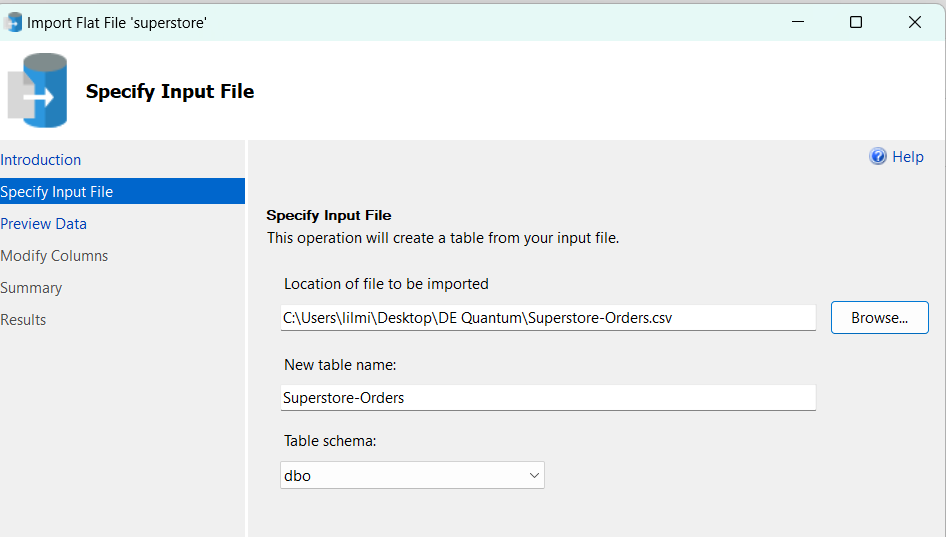
Refresh your databases and you should find your new database.

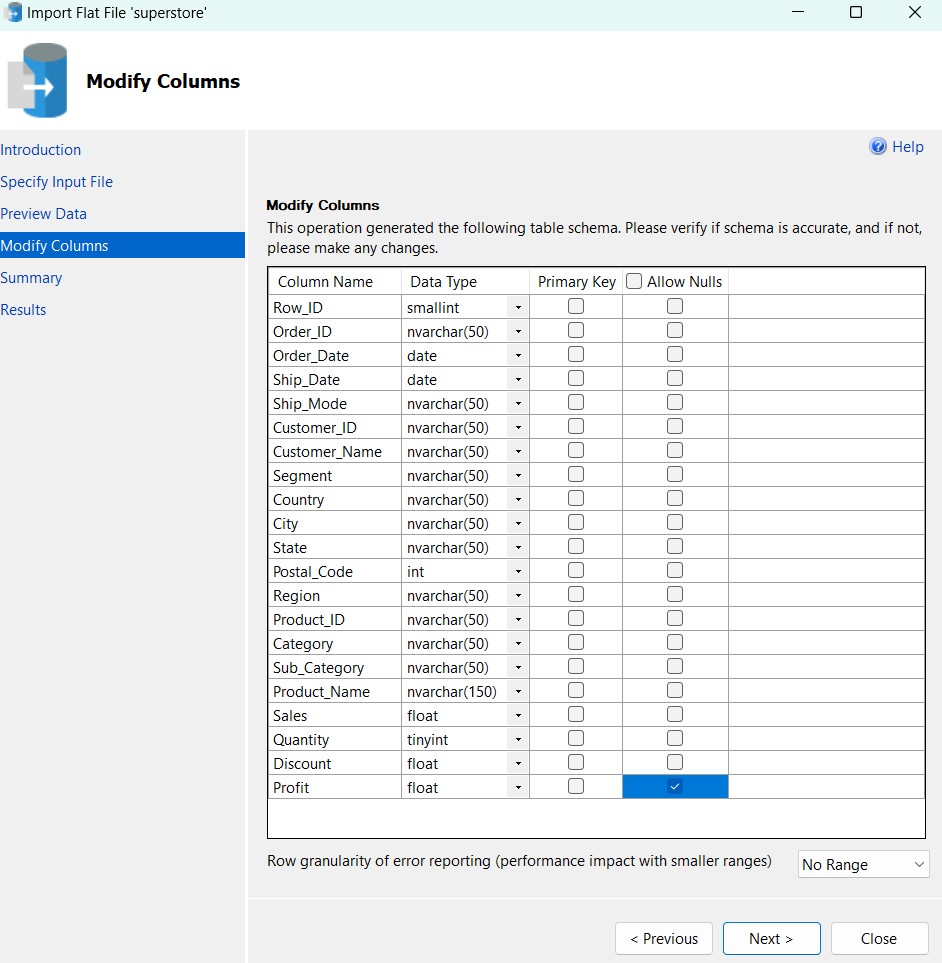


Superstore >>> Tasks >>> Import Flat file





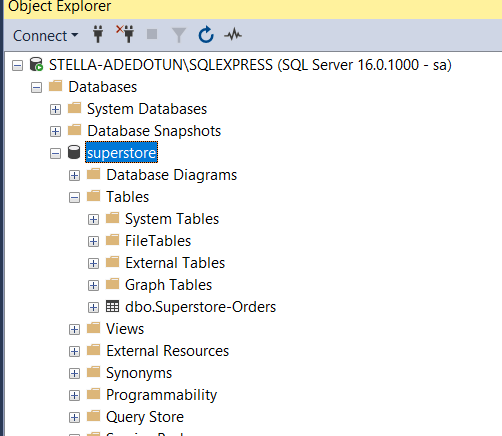




Click Next all through till you reach the Finish button and you see the message “Insert Data” comes up as “Success”.

Then, refresh your database to view your table:

Superstore >>> Tables >>> dbo.superstore-orders



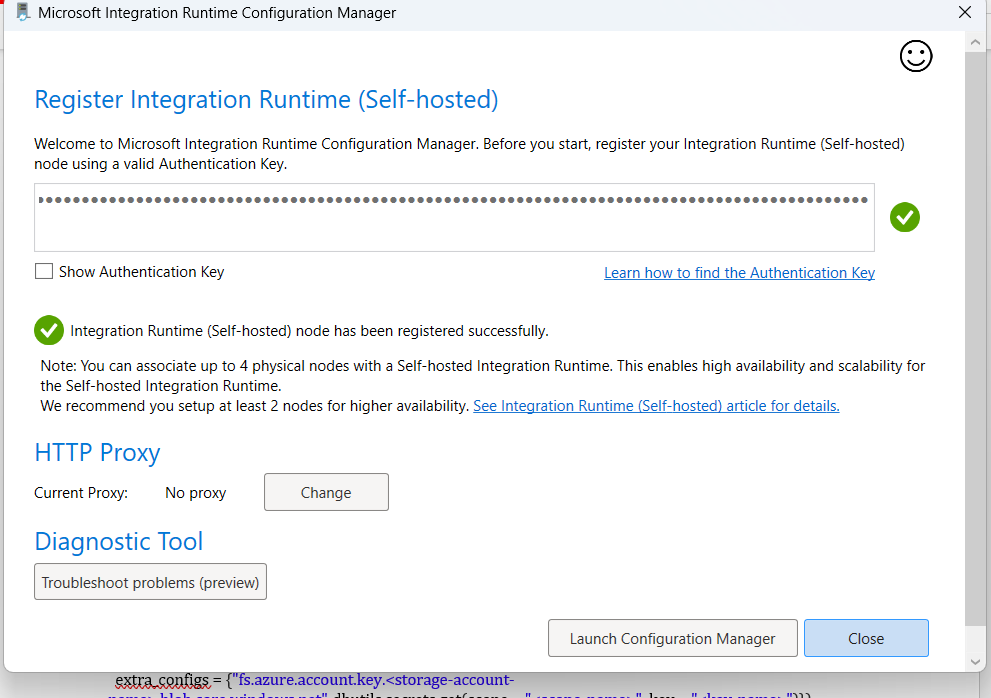
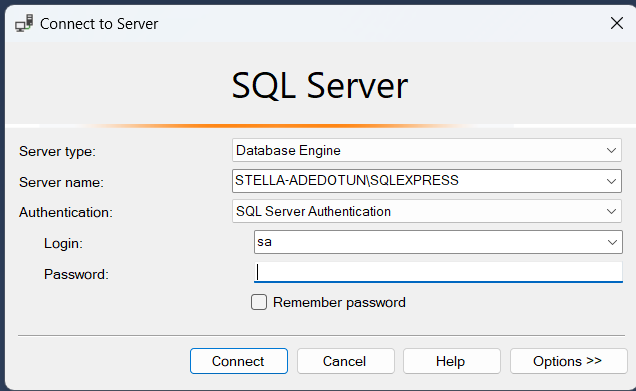
Run a query to view your file (I put the table name in double quotes because my file name contains a dash and may not be easily parsed with SQL)

A screenshot of a computer

Description automatically generated

PROCESS 2: MOVING YOUR DATA FROM SQL SERVER TO CLOUD STORAGE

1. Create your storage accounts and containers – Blob
2. Create your data factory
3. Create a pipeline and drag the Copy Activity into the pipeline
4. Click on the ‘Source’ of the Copy Activity and the ‘+’ sign next to the Source Dataset to create a the Linked Service (this linked service will be used to transfer the file from the On-prem SQL Server to the Cloud storage). Configurations for your Linked Service is shown below:
   1. Change the Integration Runtime: Connect via integration runtime >>> +New >>> Self-Hosted >>> Continue >>> Create >>> Option2 (Click on Download and install integration runtime) >>> (new page pops up) Click on Download again and select the first option(integration runtime)
   2. Go to Downloads section or the location of the on your computer to install the application
   3. When prompted as seen below; copy the integration key from the Manual Set Up section and fill it into the checkbox to register the integration runtimeA screenshot of a computer

      Description automatically generated
   4. 
   5. Click on the **Launch Configuration manager and close**
   6. Then go back to the Linked Service to fill in the just created Linked Service
   7. Fill in your Server Name and Database name (Found in the SQL Server) 
   8. Fill in the remaining details
   9. Test the connection and make sure it is successful
   10. Then ‘Create’
   11. Select your table ‘Superstore-Orders’ on the next page
   12. Preview your data

Now, switch to the SINK

1. Create a dataset for the Blob container
2. Create a Linked Service to link the container in Blob (this will be used to deposit the data into the Blob Container)
3. Publish your changes
4. Debug your pipeline
5. Confirm that your data has been transferred from your SQL Server to your ADLS Gen 2 Container

PROCESS 3 : DATABRICKS

1. Create a cluster that will be used to run the operations
2. Open a new notebook and rename it
3. Make sure that your notebook is connected to your cluster
4. Change the notebook language to Python
5. Now, we need to connect our databricks to our storage account using the ‘dbutils mount’ function

dbutils.fs.mount(

source = "wasbs://<container-name>@<storage-account-name>.blob.core.windows.net",

mount\_point = "/mnt/iotdata",

extra\_configs = {"fs.azure.account.key.<storage-account-name>.blob.core.windows.net":dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>")})

A screenshot of a computer

Description automatically generated

Mount\_point : location where your files are stored in databricks

1. Now that we have connected to the blob storage, let’s confirm that our files has been mounted successfully to the mount point. The command will list all the files on the mount.

dbutils.fs.ls(f"/mnt/{mount\_name}"))

A white rectangular box with text

Description automatically generated with medium confidence

1. Now, let’s create a dataframe (we need dataframes to work in python)

superstore\_csv = spark.read.csv(f"dbfs:/mnt/superstore/dbo.Superstore-Orders.txt", header=True, inferSchema=True, sep=",")

\*\*replace ‘superstore’ with the name of your dataframe\*\*

A screenshot of a computer

Description automatically generated

1. Now let’s view our data:

display(superstore\_csv)

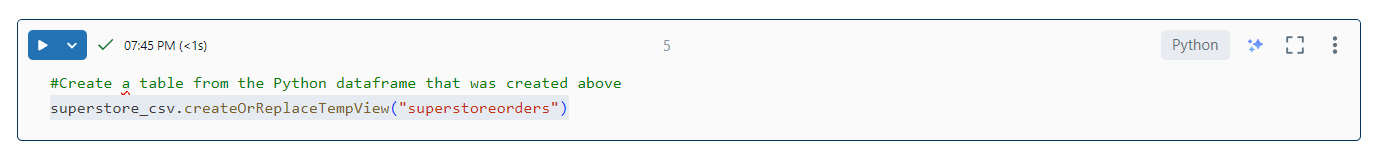
A screenshot of a computer

Description automatically generated

PROCESS 5: CREATING TABLES IN DATABRICKS

1. To create a table from our dataframe, we use the code below:

superstore\_csv.createOrReplaceTempView("table\_name")



1. Change the next cell to SQL and run the query below to view your table

Select \* from table\_name

A screenshot of a computer

Description automatically generated

1. Creating the first table or aggregate table

|  |
| --- |
| %sql  SELECT    COUNT(DISTINCT Order\_ID) AS OrderID,    EXTRACT(YEAR FROM Order\_Date) AS OrderYear,    date\_format(Order\_Date, 'MMM') AS OrderMonth,    date\_format(Order\_Date, 'EEEE') AS OrderDay,    SUM(Quantity) AS TotalQuantity,    SUM(Profit) AS TotalProfit,    SUM(Sales) AS TotalSales  FROM superstoreorders  GROUP BY EXTRACT(YEAR FROM Order\_Date), date\_format(Order\_Date, 'MMM'), date\_format(Order\_Date, 'EEEE') |

A screenshot of a computer

Description automatically generated

Download your table:



Rename it:



**Now let’s Create the smaller tables**

**Customers Table**

|  |
| --- |
| SELECT  DISTINCT (‘Customer ID’) Cust\_ID,  ‘Customer Name’,  Segment,  ‘Postal Code’,  City  FROM Superstore\_Orders |

**Products Table**

|  |
| --- |
| \*Products Table  SELECT  DISTINCT (‘Product ID’) Prod\_ID,  ‘Product Name’,  Category,  ‘Sub-Category’  FROM Superstore\_Orders |

**LOCATION TABLE**

|  |
| --- |
| SELECT  DISTINCT (‘Postcode’) Postalcode,  City,  State,  Region,  Country  FROM Superstore\_Orders |

**SALES TABLE**

|  |
| --- |
| SELECT  ‘Row\_ID’,  ‘Order\_ID’,  ‘Order Date’,  Cust\_ID,  ‘Prod\_ID’,  Quantity,  Sales,  Discount,  Profit  FROM Superstore\_Orders |

**DATE TABLE**

|  |
| --- |
| SELECT  Distinct(‘Order\_Date’),  date\_format(‘Order Date’, ‘YYY’) OrderYear,  date\_format(‘Order Date’, ‘MMM’) OrderMonth,  date\_format(‘Order Date’, ‘EEEE’) OrderDay,  FROM Superstore\_Orders; |

**Download your tables and name them appropriately**:

**\*\*\*MODEL YOUR TABLES IN POWERBI (Should either be a star or a snowflake model)**

RESOURCES

Databricks resource connection links: <https://docs.databricks.com/en/_extras/notebooks/source/data-sources/mount-azure-blob-storage.html>

Transform Data Using Apache Spark Dataframes

<https://docs.databricks.com/en/getting-started/dataframes.html>