

Data Engineering Project -1

ETL Pipeline for Global Superstore Dataset

Team:

Aathirainathan P
Ambati Sesha Sai Sahithya
Eswara Venkata Sai Raja

Date: 16-12-2024

1. Project Overview:

The **Global Superstore Dataset** project leverages **Azure Data Engineering tools** to build a robust data pipeline for ingesting, transforming, and analyzing sales, profit, and operational data of a global retail store. This end-to-end solution demonstrates how to process large-scale retail data for actionable business insights by utilizing **Azure Data Factory (ADF)**, **Azure Databricks**, and **Delta Lake**.

2. Project Statement:

The objective of this project is to construct a three-tier data pipeline (**Bronze, Silver, and Gold zones**) that extracts data from Azure Blob Storage to Azure Delta Lake Storage, transforms it using **Azure Databricks**, and stores the results in a **Azure Databricks Delta Lake**. The project focuses on enabling seamless data processing, advanced analytics, and visualization to uncover trends in sales, profit margins, and regional performance.

Key deliverables include:

- Building scalable data pipelines for data ingestion, transformation, and loading (ETL).
- Cleaning and standardizing raw data to create a trusted dataset.
- Performing analytical transformations and aggregations to support data-driven decision-making.
- Visualizing key metrics such as top-performing regions, high-profit categories, and trends over time.

This project showcases the practical application of modern cloud data platforms to address challenges in retail data management and analytics, enabling business leaders to make informed decisions.

3. Data Overview:

The **Global Superstore Dataset** provides a comprehensive view of a global retail operation, covering various aspects of sales, profits, shipping, and customer segmentation. The dataset contains transactional data across multiple dimensions, offering rich insights into the performance of the business. Below are the key components of the dataset:

Data Source

The dataset originates from **Azure Blob Storage**, where the raw data is stored as a CSV file named **Global_Superstore2.csv**. This data is ingested and processed through the Azure data engineering pipeline.

Schema Details:

The dataset schema includes the following key attributes:

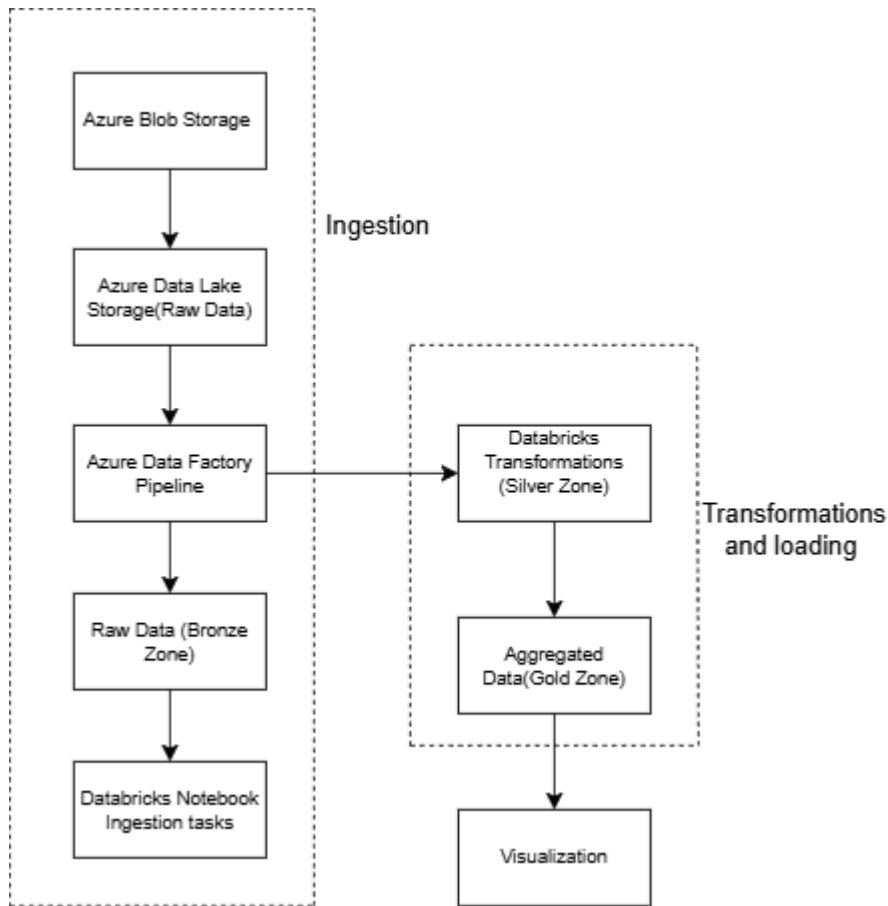
Column Name	Description
Row ID	Unique identifier for each record.
Order ID	Unique ID for each order placed.
Order Date	The date when the order was placed.
Ship Date	The date when the order was shipped.
Ship Mode	Shipping method chosen for the order (e.g., Standard Class, First Class).
Customer ID	Unique identifier for each customer.
Customer Name	Name of the customer placing the order.
Segment	Market segment (e.g., Consumer, Corporate, Home Office).
City	City of the customer.
State	State of the customer.
Country	Country of the customer.
Postal Code	Postal code of the customer.
Region	Region of the customer (e.g., East, West).
Product ID	Unique identifier for each product.
Category	Product category (e.g., Office Supplies, Furniture).
Sub-Category	Sub-category of the product (e.g., Binders, Chairs).
Product Name	Name of the product.
Sales	Sales amount for the product.
Quantity	Quantity of the product ordered.
Discount	Discount applied to the order.
Profit	Profit generated from the sale.
Shipping Cost	Cost of shipping the order.
Order Priority	Priority level of the order (e.g., High, Medium).

Data Zones

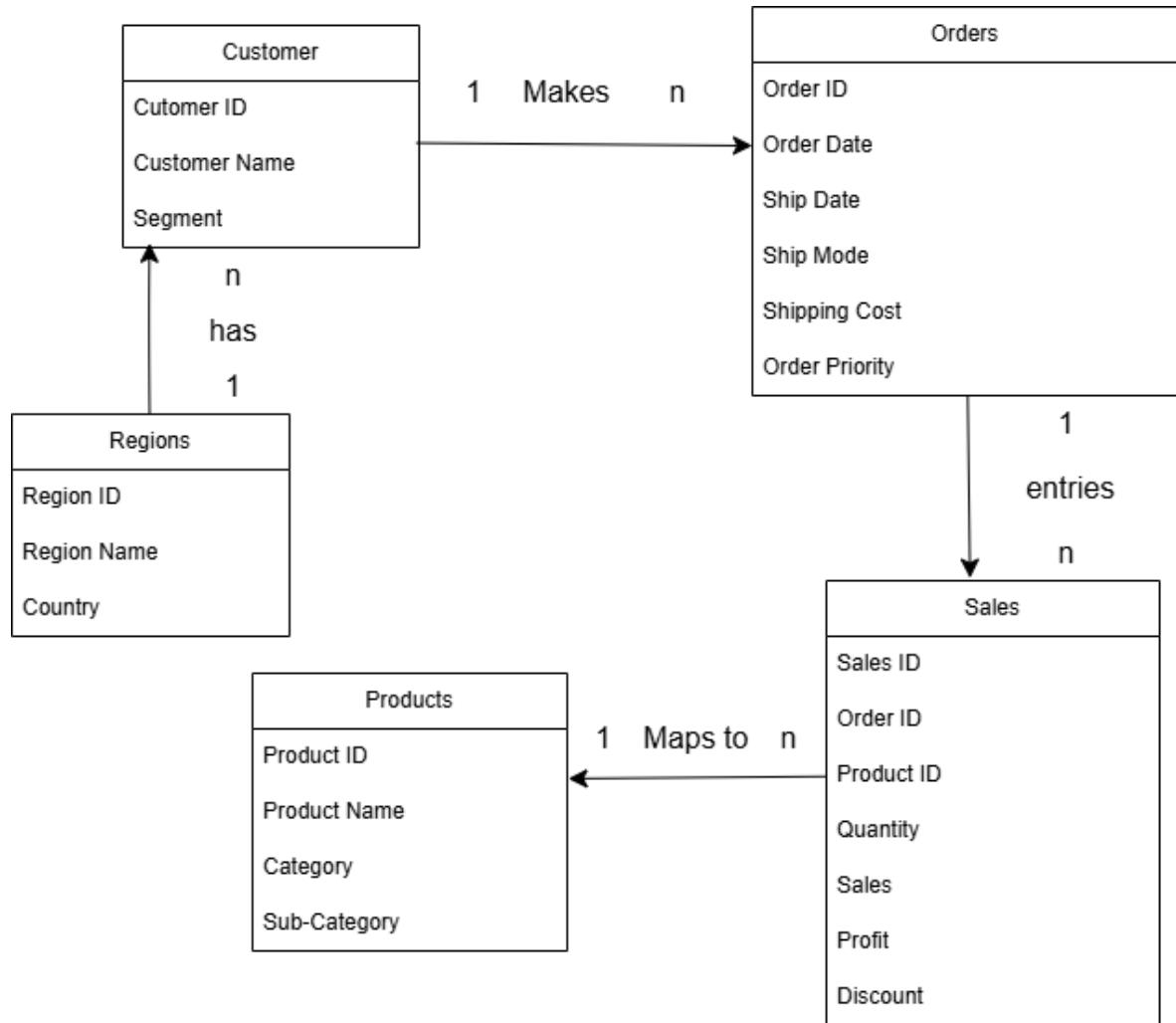
The dataset undergoes the following transformations through the pipeline:

1. **Bronze Zone:** Stores the raw data ingested from Azure Blob Storage.
2. **Silver Zone:** Contains cleaned and standardized data after transformations (e.g., type casting, null handling).
3. **Gold Zone:** Contains aggregated data optimized for analytics and visualization.

4. Architectural Diagram:



5. ER Diagram:



6. How it Works:

Source Data Files

We are using the **Global Superstore Dataset**, a publicly available dataset containing sales, orders, and customer data across regions. This data is ideal for ETL and analytics use cases.

File Details

File Name	File Type	Description
Global_Superstore2.csv	CSV	Contains data on orders, customers, products, and regions, including details like sales, profit, and discounts.

7. Execution Overview:

Step 1: Ingestion (Bronze Zone)

- The raw data file (Global_Superstore.csv) is uploaded to **Azure Blob Storage**.
- Azure Data Factory (ADF) pipelines copy the file into **Azure Data Lake Storage Gen2 (ADLS)**.
- The ingestion tasks involve mounting the data in **Databricks** for further processing.

Step 2: Transformation (Silver Zone)

- Using Databricks Notebooks, the raw data is cleaned and transformed into structured data:
 - Data types are cast appropriately (e.g., sales as float, quantity as int).
 - Columns like Profit Margin and High Discount are calculated.
 - Null values and invalid rows are removed.

Step 3: Analytical Zone (Gold Zone)

- The cleaned and aggregated data in the Silver Zone is used to:
 - Compute sales and profit by region, category, and year.
 - Identify high-profit orders and discount trends.
- The processed data is used in Delta format for visualizations and insights.

Pipeline Overview:

1. Copy Data Activity (Blob Storage to Data Lake)

- **Purpose:** The first step of the pipeline is to copy raw data (in CSV/JSON format) from Azure Blob Storage to Azure Data Lake Storage Gen2 (ADLS).
- **Source:** Azure Blob Storage (raw Global Superstore dataset).
- **Destination:** Azure Data Lake Storage Gen2 (Bronze Zone).

2. Ingestion Activity (Raw Data to Delta Format)

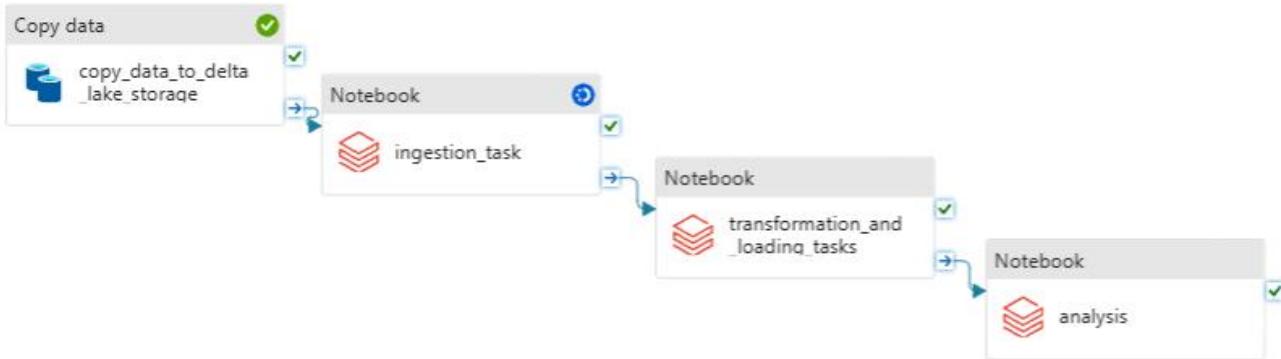
- **Purpose:** The second activity reads the raw data stored in the Bronze Zone and performs minimal transformations using a **Databricks Notebook**. These transformations include:
 - **Schema Application:** Ensures that the data has a consistent schema.
 - **Renaming Columns:** Columns are renamed to remove special characters.
 - **Dropping Unnecessary Columns:** Redundant or unnecessary data columns are removed.
- **Output:** Data is stored as Delta tables in the **Silver Zone** (standardized zone).

3. Transformation Activity (Silver to Gold Zone)

- **Purpose:** In this activity, a Databricks SQL notebook is executed to transform the preprocessed data into the final structured form. Key transformations include:
 - **Deduplication:** Remove duplicate rows.
 - **Aggregations:** Grouping data by category, region, and other key attributes, and applying aggregations like SUM, COUNT, etc.
 - **Computed Columns:** Additional columns (e.g., profit margin) are created.
- **Output:** The transformed data is saved in the **Gold Zone** (analytical zone) as Delta tables, making it ready for analysis and visualization.

4. Analysis Activity (Data Queries and Visualization)

- **Purpose:** The final activity executes SQL queries to analyze the transformed data stored in the Gold Zone. Examples of analysis could be:
 - **Top Regions by Total Sales:** Identify which regions contribute the most to the total sales.
 - **Profit Margin Analysis:** Calculate and analyze profit margins across different categories.
 - **Sales Trends:** Track sales over time, by category or region.
- **Output:** The results of the SQL queries are displayed in notebooks and can be visualized using Databricks' built-in visualization tools (e.g., bar charts, pie charts).



8. Resources Used for the project:

- Azure Data Lake Storage
- Azure Blob Storage
- Azure Data Factory
- Azure Databricks
- Azure Databricks Delta Lake

9. Project Requirements:

1. Ingestion Tasks Requirements:

- Mount Azure Blob Storage
- Verify the mount
- Load Data from Blob Storage
- Inspect and Verify Schema
- Persist Raw Data (Bronze Zone)
- Perform Data Validation Checks
- Deduplicate Raw Data
- Save Cleaned Raw Data in Bronze Zone

2. Transformation Tasks & Load Requirements:

- Add a Column
- Convert Sales and Profit to Float
- Filter Rows with Zero or Negative Profit
- Group by Category and Region and perform aggregation
- Add a Profit Margin Column
- Remove Duplicate Rows based on 'Category' and 'Region'
- Rename Columns for Clarity
- Add a Year Column
- Filter the Data for a Specific Year
- Sort Data by Total Sales
- Save Transformed Data in Silver Zone

3. Analysis Requirements:

- Top 5 Regions by Total Sales
- Top Categories by Profit Margin
- Year-over-Year Sales Growth
- Regions with Highest Discounts
- Most Profitable Category-Region Pair

10. Tasks Performed:

Creating an Azure Blob storage:

The screenshot shows the Microsoft Azure portal interface for creating a new storage account. The top navigation bar includes 'Create a storage account - Micr...', a search bar, and various icons. The main title is 'Create a storage account'. Below it, there are tabs for 'Basics', 'Advanced', 'Networking', 'Data protection', 'Encryption', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The 'Project details' section asks for a subscription ('MML Learners') and a resource group ('rg-azuser2356_mml.local-eylrK'). The 'Resource group' dropdown has a 'Create new' link. The 'Instance details' section requires a 'Storage account name' ('sourceaccount1hexa') and a 'Region' ('(Asia Pacific) Central India'). There is also a 'Deploy to an Azure Extended Zone' link. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons, along with a 'Give feedback' link. The status bar at the bottom shows system information like weather (84°F Haze), search, and file icons.

The screenshot shows the Microsoft Azure Deployment Overview page for a deployment named "sourceaccount1hexa_1734262842255". The status is "Your deployment is complete". Deployment details include a name, subscription ("MML Learners"), and resource group ("rg-azuser2356_mml.local-eylrK"). The deployment started at 12/15/2024, 5:11:09 PM. A Correlation ID is also provided. On the right, there are links for Cost Management, Microsoft Defender for Cloud, Free Microsoft tutorials, and Work with an expert. The bottom navigation bar shows the date as 15-12-2024.

Creating a delta lake storage account:

The screenshot shows the Microsoft Azure "Create a storage account" page. The account type is set to "destinationlakeacchexa" in the "Storage account name" field. The region is "(Asia Pacific) Central India". The primary service is "Azure Blob Storage or Azure Data Lake Storage Gen 2". The performance level is "Standard" (selected). Redundancy is set to "Locally-redundant storage (LRS)". The bottom navigation bar shows the date as 15-12-2024.

A Create a storage account - Microsoft Azure

portal.azure.com/#create/Microsoft.StorageAccount-ARM

Microsoft Azure

Home > Storage accounts >

Create a storage account

Permitted scope for copy operations (preview) From any storage account

Hierarchical Namespace

Hierarchical namespace, complemented by Data Lake Storage Gen2 endpoint, enables file and directory semantics, accelerates big data analytics workloads, and enables access control lists (ACLs) [Learn more](#)

Enable hierarchical namespace

Access protocols

Blob and Data Lake Gen2 endpoints are provisioned by default [Learn more](#)

Enable SFTP

Enable network file system v3

Blob storage

Allow cross-tenant replication
Cross-tenant replication and hierarchical namespace cannot be enabled simultaneously.

Access tier Hot: Optimized for frequently accessed data and everyday usage scenarios
 Cool: Optimized for infrequently accessed data and backup scenarios

Previous Next Review + create Give feedback

A destinationlakeacchexa_1734263037295 | Overview

portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2F2a3c6418-97b9-4d96-a24b-2c2d7633d375%2FresourceGroups%2Frg-azuser2356_mml.local@eylirk

Microsoft Azure

Home >

destinationlakeacchexa_1734263037295 | Overview

Deployment

Search Delete Cancel Redeploy Download Refresh

Overview Your deployment is complete

Deployment name: destinationlakeacchexa_1734263037295
Subscription: MML Learners
Resource group: rg-azuser2356_mml.local@eylirk

Start time: 12/15/2024, 5:14:09 PM
Correlation ID: f92b38a4-61b6-4ba0-aa5e-513341381b47

Deployment details

Next steps

Go to resource Give feedback Tell us about your experience with deployment

Cost Management
Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

Microsoft Defender for Cloud
Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

Free Microsoft tutorials
[Start learning today >](#)

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
[Find an Azure expert >](#)

84°F Haze Search

Upload the data into a container of the source account:

The screenshot shows the Microsoft Azure Storage account overview page for 'superstore-data'. The top navigation bar includes 'Copilot', 'Search resources, services, and docs', and a user profile. A success message box at the top right says 'Successfully uploaded blob(s)'. The main content area displays a table of blobs, with one entry: 'Global_Superstore2.xlsx' (Modified: 12/15/2024, 5:18:06 ...). The left sidebar has 'Overview' selected, along with 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The taskbar at the bottom shows various pinned icons.

Create an Azure Data Factory:

The screenshot shows the 'Create Data Factory' wizard on the 'Basics' step. The top navigation bar includes 'Copilot', 'Search resources, services, and docs', and a user profile. A warning message at the top says 'Changes on this step may reset later selections you have made. Review all options prior to deployment.' The 'Basics' tab is selected, with other tabs like 'Git configuration', 'Networking', 'Advanced', and 'Tags' available. The 'Review + create' button is at the bottom. The configuration fields include 'Subscription' (MML Learners), 'Resource group' (rg-azuser2356_mml.local-eylRk), 'Name' (SuperstoreADFFhexa), 'Region' (South India), and 'Version' (V2). The taskbar at the bottom shows various pinned icons.

A Microsoft.DataFactory-20241215172028 | Overview

Deployment succeeded

Deployment 'Microsoft.DataFactory-20241215172028' to resource group 'rg-azuser2356_mml.local-eylrk' was successful.

Pin to dashboard Go to resource group

Overview

Deployment name : Microsoft.DataFactory-20241215172028
Subscription : MML Learners
Resource group : rg-azuser2356_mml.local-eylrk

Start time : 12/15/2024, 5:22:02 PM
Correlation ID : ad503196-1f64-416a-a896-5d9c5c6058e4

Deployment details

Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

Rain showers Thursday

Search

Cloud icons

17:22 ENG IN 15-12-2024

In linked services add the source and destination services (Azure Blob Storage, Azure Delta Lake Storage):

The screenshot shows the Microsoft Azure Data Factory interface. On the left, a sidebar menu is open under the 'Connections' section, with 'Linked services' selected. The main area displays a grid of 'New linked service' options categorized into 'Data store' and 'Compute'. The 'Data store' category includes Azure Blob Storage, Azure Cosmos DB for MongoDB, and Azure Cosmos DB for NoSQL. The 'Compute' category includes Azure Data Explorer (Kusto), Azure Data Lake Storage Gen2, Azure Database for MariaDB (Legacy), Azure Database for MySQL, Azure Database for PostgreSQL, and Azure Databricks Delta Lake. A 'Create linked service' button is visible at the bottom of the grid.

This screenshot shows the configuration of a new Azure Blob Storage linked service. The 'Name' field is set to 'AzureBlobStorage1'. Under 'Connect via integration runtime', 'AutoResolveIntegrationRuntime' is selected. The 'Authentication type' is set to 'Account key'. In the 'Connection string' section, 'Azure Key Vault' is chosen. The 'Account selection method' is set to 'From Azure subscription', with 'Select all' chosen. The 'Azure subscription' dropdown shows 'sourceaccount1hexa'. The 'Storage account name' is set to 'sourceaccount1hexa'. At the bottom, there are 'Create', 'Back', 'Test connection', and 'Cancel' buttons.

Screenshot of the Microsoft Azure Data Factory 'Linked services' page.

The left sidebar shows the navigation path: Microsoft Azure | Data Factory > SuperstoreADHexa. The 'Connections' section is selected.

The main area displays the 'Linked services' list:

- Showing 1 - 1 of 1 items
- Name: AzureBlobStorage1 (Type: Azure Blob Storage)

A modal window titled 'New linked service' is open, showing a grid of data store and compute options:

Data store	Compute	
Azure Blob Storage	Azure Cosmos DB for MongoDB	Azure Cosmos DB for NoSQL
Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen2	Azure Database for MariaDB (Legacy)
Azure Database for MySQL	Azure Database for PostgreSQL	Azure Databricks Delta

Buttons at the bottom of the modal: Continue, Cancel.

Screenshot of the Microsoft Azure Data Factory 'Linked services' page, showing the configuration of a new linked service.

The left sidebar shows the navigation path: Microsoft Azure | Data Factory > SuperstoreADHexa. The 'Connections' section is selected.

The main area displays the 'Linked services' list:

- Showing 1 - 1 of 1 items
- Name: AzureBlobStorage1 (Type: Azure Blob Storage)

A modal window titled 'New linked service' is open, showing the configuration steps:

- Step 1: Set 'Azure Data Lake Storage Gen2' as the type.
- Step 2: 'Connect via integration runtime': Selected 'AutoResolveIntegrationRuntime'.
- Step 3: 'Authentication type': Selected 'Account key'.
- Step 4: 'Account selection method': Selected 'From Azure subscription'.
- Step 5: 'Azure subscription': Selected 'MML Learners (2a3c6418-97b9-4d96-a24b-2c2d7633d375)'.
- Step 6: 'Storage account name': Entered 'destinationlakeacchexa'.
- Step 7: 'Test connection': Selected 'To linked service'.
- Step 8: 'Annotations': New annotation is being added.

Buttons at the bottom of the modal: Create, Back, Test connection, Cancel.

Create a new pipeline and perform a Copy data tool to move data to Delta Lake Account:

The screenshot shows the Microsoft Azure Data Factory Copy Data tool wizard. The current step is 'Properties'. The left sidebar lists steps 1 through 5: Properties, Source, Destination, Settings, and Review and finish. The main area is titled 'Properties' and contains the following information:

Use Copy Data Tool to perform a one-time or scheduled data load from 90+ data sources. Follow the wizard experience to specify your data loading settings, and let the Copy Data Tool generate the artifacts for you, including pipelines, datasets, and linked services. [Learn more](#)

Properties

Select copy data task type and configure task schedule

Task type

Built-in copy task
You will get a single pipeline which is capable of smoothly copying data from over 100 different data sources.

Metadata-driven copy task
You will get parameterized pipelines which can read metadata from an external store to load data at a large scale.

You will get single pipeline to quickly copy objects from data source store to destination in a very intuitive manner.

Task cadence or task schedule *

Run once now Schedule Tumbling window

At the bottom are 'Previous' and 'Next >' buttons, and a 'Cancel' button in the top right corner.

Select source data:

The screenshot shows the Microsoft Azure Data Factory Copy Data tool wizard. The current step is 'Source'. The left sidebar lists steps 1 through 5: Properties, Source, Dataset, Configuration, Destination, Settings, and Review and finish. The main area is titled 'Source data store' and contains the following configuration:

Specify the source data store for the copy task. You can use an existing data store connection or specify a new one.

Source type: All

Connection *: AzureBlobStorage1

File or folder *: Global_Superstore2.xlsx

Options:

Binary copy ①
 Recursively ②
 Enable partitions discovery ③

Max concurrent connections: ④

Filter by last modified

Start time (UTC):
End time (UTC):

A 'Browse' panel on the right shows the file 'Global_Superstore2.xlsx' under the root folder 'superstore-data'. At the bottom are 'OK' and 'Cancel' buttons.

Select destination container:

The screenshot shows the Microsoft Azure Data Factory Copy Data tool wizard at the 'Destination' step. The left sidebar lists steps 1 through 5: Properties, Source, Destination, Dataset, Configuration, Settings, and Review and finish. Step 3, 'Destination', is selected and highlighted in blue. The main panel is titled 'Destination data store' and contains the following fields:

- Destination type:** All
- Connection:** AzureDataLakeStorage1
- Folder path:** (disabled)
- File name:** Filenames are defined by source
- Compression type:** None
- Copy behavior:** Select...
- Max concurrent connections:** (disabled)
- Block size (MB):** (disabled)

Below the main panel is a 'Browse' window titled 'destination-acc'. It displays the message 'Select a file or folder.' and shows the path 'Root folder > destination-acc'. A note indicates 'Showing 0 items'. At the bottom right of the main panel are 'OK' and 'Cancel' buttons.

The screenshot shows the Microsoft Azure Data Factory Copy Data tool wizard at the 'Settings' step. The left sidebar lists steps 1 through 5: Properties, Source, Destination, Dataset, Configuration, Settings, and Review and finish. Step 4, 'Settings', is selected and highlighted in blue. The main panel is titled 'Settings' and contains the following fields:

- Task name:** CopyPipeline_xgw
- Task description:** (empty text area)
- Data consistency verification:** (checkbox)
- Fault tolerance:** Skip missing files
- Enable logging:** (checkbox)
- Enable staging:** (checkbox)

At the bottom right of the main panel are 'Previous' and 'Next' buttons. The status bar at the bottom of the screen shows the date and time as 15-12-2024 and 17:33.

Screenshot of the Microsoft Azure Copy Data tool interface showing a successful deployment from Azure Blob Storage to Azure Data Lake Storage Gen2.

Copy Data tool

Deployment complete

Deployment step	Status
Validating copy runtime environment	Succeeded
> Creating datasets	Succeeded
> Creating pipelines	Succeeded
> Running pipelines	Succeeded

Datasets and pipelines have been created. You can now monitor and edit the copy pipelines or click finish to close Copy Data Tool.

Finish | Edit pipeline | Monitor

83°F Haze | Search | ENG IN | 15-12-2024

Screenshot of the Microsoft Azure Data Factory interface showing the Project 1 pipeline configuration.

Factory Resources

- Pipelines (1)
- Project 1 pipeline
- Change Data Capture (preview) (0)
- Datasets (2)
- Data flows (0)
- Power Query (0)

Project 1 pipeline

Activities

- Copy data

Source dataset: SourceDataset_xgw

File path type: Wildcard file path

Wildcard paths: superstore-data / Wildcard folder path / *

Start time (UTC): [empty]

End time (UTC): [empty]

Filter by last modified: [empty]

Recursively: checked

Watchlist Ideas | Search | ENG IN | 11:14 | 16-12-2024

Create Azure Databricks Workspace:

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * MML Learners

Resource group * rg-azuser2356_mml.local-eylrK

Workspace name * Myworkspacehexa

Region * Central India

Pricing Tier * Standard (Apache Spark, Secure with Microsoft Entra ID)

Managed Resource Group name Enter name for managed resource group

Review + create < Previous Next : Networking >

Your deployment is complete

Deployment name : rg-azuser2356_mml.local-eylrK_Myworkspacehexa
Subscription : MML Learners
Resource group : rg-azuser2356_mml.local-eylrK

Start time : 12/15/2024, 5:39:19 PM
Correlation ID : f5998383-4814-4369-ab98-277a1bd64e43

Deployment details

Next steps

Go to resource

Give feedback
Tell us about your experience with deployment

Cost management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >

Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials
Start learning today >

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
Find an Azure expert >

Create a cluster:

The screenshot shows the Microsoft Azure Databricks interface for creating a new cluster. The top navigation bar includes tabs for 'Myworkspacehexa - Microsoft' (active), 'Create Cluster - Databricks', and 'SuperstoreADHexa - Azure Data'. The URL in the address bar is `adb-1727980957508452.12.azuredata.databricks.net/compute/clusters/new?o=1727980957508452`. The main content area is titled 'azuser2356_mml.local's Cluster'.

Compute Options:

- Compute Type:** Single node
- Access mode:** Single user access (selected)
- Single user:** azuser2356_mml.local

Performance Settings:

- Databricks runtime version:** Runtime: 15.4 LTS (Scala 2.12, Spark 3.5.0)
- Node type:** Standard_D4ds_v5 (16 GB Memory, 4 Cores)
- Termination:** Checkmark for 'Terminate after 60 minutes of inactivity'

Tags: Add tags (Key: Value) and an 'Add' button.

Buttons: 'Create compute' (blue) and 'Cancel'.

Summary (UI | JSON):

Summary	
1 Driver	16 GB Memory, 4 Cores
Runtime	15.4.x-scala2.12
Standard_D4ds_v5	1 DBU/h

Create a notebook and copy the key of the storage account:

The screenshot shows the Microsoft Azure Storage accounts page. The left sidebar lists storage accounts: destinationlakeacchexa and sourceaccount1hexa. The main content area is titled "sourceaccount1hexa | Access keys" and shows two access keys: "key1" and "key2". Each key is associated with a rotation date of "12/15/2024 (0 days ago)". The "key1" section includes a "Show" button for the key and a "Show" button for the connection string. The "key2" section also includes similar "Show" buttons. At the bottom, there are navigation buttons for "Page 1 of 1" and a Windows taskbar with various pinned icons.

Perform the ingestion tasks in the notebook:

1. Mount Azure Blob Storage

```
Just now (21s) 1 Python ::

%python
# Fetching the csv file from the blob storage
storage_account_name = "sourceaccount1hexa"
container_name = "superstore-data"
storage_account_key = "ooZRQBCJR9AwKPvv5YvKprjv3U1GndcovcyJcjT4dVN8DbxojxsSgLojGldlb400ZMJd1AqFqrY2+ASlj31gDQ=="

# Unmount the directory if it is already mounted
if any(mount.mountPoint == "/mnt/superstore" for mount in dbutils.fs.mounts()):
    dbutils.fs.unmount("/mnt/superstore")

# Mount Blob Storage
dbutils.fs.mount(
    source=f"wasbs://{{container_name}}@{{storage_account_name}}.blob.core.windows.net",
    mount_point="/mnt/superstore",
    extra_configs={
        f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": storage_account_key
    }
)

/mnt/superstore has been unmounted.

True
```

2. Verify the mount:

```
Just now (1s) 2 Python ::

#verifying the mount
display(dbutils.fs.ls("/mnt/superstore"))

▶ (2) Spark Jobs

Table + 🔍 🔍 🔍

path name size modificationTime
1 dbfs:/mnt/superstore/Global_Superstore2.c... Global_Superstore2.c... 12089916 1734265964000

↓ 1 row | 0.52 seconds runtime Refreshed now
```

3. Load Data from Blob Storage

06:04 PM (4s) 3 Python ⚡ ⏺ ⏹

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

# Load the dataset
file_path = "/mnt/superstore/Global_Superstore2.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)
df.show(5)

▶ (3) Spark Jobs
```

df: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 22 more fields]

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	Country		
Postal Code	Market	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority
32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	United States		
10024	US	East	TEC-AC-10003033	Technology	Accessories	Plantronics CS510...	2309.65	7	0	762.1845	933.57	Critical
26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	Australia		
NULL	APAC Oceania	FUR-CH-10003950	Furniture	Chairs	Novimex Executive...	3709.395	9	0.1	-288.765	923.63	Critical	
25220	TN-2012-71219	2012-10-17	2013-10-18	First Class	CR-12720	Carrie Reiter	Consumer	Brisbane	Queensland	Australia		

4. Inspect and Verify Schema

12:17 PM (<1s) 4

```
# Inspect and Verify Schema
df.printSchema()
```

```
-- Ship Mode: string (nullable = true)
-- Customer ID: string (nullable = true)
-- Customer Name: string (nullable = true)
-- Segment: string (nullable = true)
-- City: string (nullable = true)
-- State: string (nullable = true)
-- Country: string (nullable = true)
-- Postal Code: integer (nullable = true)
-- Market: string (nullable = true)
-- Region: string (nullable = true)
-- Product ID: string (nullable = true)
-- Category: string (nullable = true)
-- Sub-Category: string (nullable = true)
-- Product Name: string (nullable = true)
-- Sales: string (nullable = true)
-- Quantity: string (nullable = true)
-- Discount: string (nullable = true)
-- Profit: double (nullable = true)
-- Shipping Cost: double (nullable = true)
-- Order Priority: string (nullable = true)
```

5. Persist Raw Data (Bronze Zone)

```
▶ ✓ 12:17 PM (23s) 5
#Persist Raw Data (Bronze Zone)
bronze_zone_path = "/mnt/bronze/superstore"
df.write.format("delta").option("mergeSchema", "true").option("delta.columnMapping.mode", "name").mode("overwrite").save(bronze_zone_path)

▶ (6) Spark Jobs
```

6. Perform Data Validation Checks

```
▶ ✓ 12:17 PM (10s) 6
#Perform Data Validation Checks
# Check for null values in critical columns

from pyspark.sql.functions import col
df.filter((col("Order ID").isNull() | col("Customer ID").isNull())).show()

▶ (2) Spark Jobs
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Row ID|Order ID|Order Date|Ship Date|Ship Mode|Customer ID|Customer Name|Segment|City|State|Country|Postal Code|Market|Region|Product ID|Category|
Sub-Category|Product Name|Sales|Quantity|Discount|Profit|Shipping Cost|Order Priority|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

7. Deduplicate Raw Data

```
▶ ✓ 12:17 PM (<1s) 7
#Deduplicate Raw Data
df = df.dropDuplicates(["Order ID", "Row ID"])

▶ df: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 22 more fields]
```

8. Save Cleaned Raw Data in Bronze Zone

```
▶ ✓ 12:17 PM (11s) 8
#Save Cleaned Raw Data in Bronze Zone
df.write.format("delta").mode("overwrite").save(bronze_zone_path)

▶ (8) Spark Jobs
▶ ✓ 12:17 PM (1s) 9
dbutils.notebook.exit(bronze_zone_path)

Notebook exited: /mnt/bronze/superstore
```

Perform Transformation & Load tasks in the Notebook:

1. Add a Column

06:08 PM (1s) 4

```
#Add Total Cost Column
from pyspark.sql.functions import col

df_transformed = df.withColumn("TotalCost", col("Sales") - col("Profit"))
df.show(5)

▶ (1) Spark Jobs
```

df_transformed: pyspark.sql.dataframe.DataFrame = [Row ID:integer, Order ID:string ... 23 more fields]

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	Country			
Postal Code	Market	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority	TotalCost
32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	United States		Critical	10024
26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	Australia		NULL	
25330	ES-2013-1579342	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	Australia		NULL	
13524	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer	Berlin	Berlin	Germany		NULL	
47221	AF-2013-154746550000002	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office	Dakar	Dakar	Senegal		Africa	

only showing top 5 rows

2. Convert Sales and Profit to Float

06:09 PM (1s) 5 Python

```
#Convert Sales and Profit to Float

df_transformed = df_transformed.withColumn("Sales", col("Sales").cast("float"))
df_transformed = df_transformed.withColumn("Profit", col("Profit").cast("float"))

df_transformed.show(5)

▶ (1) Spark Jobs
```

df_transformed: pyspark.sql.dataframe.DataFrame = [Row ID:integer, Order ID:string ... 23 more fields]

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	Country			
Postal Code	Market	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Shipping Cost	Order Priority	TotalCost
32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	United States		Critical	10024
26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	Australia		NULL	
25330	ES-2013-1579342	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	Australia		NULL	
13524	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer	Berlin	Berlin	Germany		NULL	
47221	AF-2013-154746550000002	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office	Dakar	Dakar	Senegal		Africa	

only showing top 5 rows

3. Filter Rows with Zero or Negative Profit

06:10 PM (1s) 6 Python ⚙️ ⚡ ⋮

```
#Filter Rows with Zero or Negative Profit

df_transformed = df_transformed.filter(col("Profit") > 0)
df_transformed.show(5)

▶ (1) Spark Jobs
```

df_transformed: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]										
32298 CA-2012-124891 2012-07-31 2012-07-31 Same Day RH-19495 Rick Hansen Consumer New York City New York United States										
10024 US East TEC-AC-10003033 Technology Accessories Plantronics CS510... 2309.65 7 0 762.1845 933.57 Critic										
cal 1547.4655000000002										
25330 IN-2013-71249 2013-10-17 2013-10-18 First Class CR-12730 Craig Reiter Consumer Brisbane Queensland Australia										
NULL APAC Oceania TEC-PH-10004664 Technology Phones Nokia Smart Phone... 5175.171 9 0.1 919.971 915.49 Medi										
um 4255.20000000001										
47221 SG-2013-4320 2013-11-05 2013-11-06 Same Day RH-9495 Rick Hansen Consumer Dakar Dakar Senegal										
NULL Africa Africa TEC-SHA-10000501 Technology Copiers Sharp Wireless Fa... 2832.96 8 0 311.52 903.04 Critic										
al 2521.44										
22732 IN-2013-42360 2013-06-28 2013-07-01 Second Class JM-15655 Jim Mitchum Corporate Sydney New South Wales Australia										
NULL APAC Oceania TEC-PH-10000030 Technology Phones Samsung Smart Pho... 2862.675 5 0.1 763.275 897.35 Critic										
al 2099.4										
30570 IN-2011-81826 2011-11-07 2011-11-09 First Class TS-21340 Toby Swindell Consumer Porirua Wellington New Zealand										
NULL APAC Oceania FUR-CH-10004050 Furniture Chairs Novimex Executive... 1822.08 4 0 564.84 894.77 Critic										
al 1257.239999999998										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
-+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
--+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
only showing top 5 rows										

4. Group by Category and Region and perform aggregation

12:25 PM (1s) 5

```
from pyspark.sql.functions import col, sum
#Group by Category and Region and perform aggregation

# Step 1: Clean the data by filtering out rows with non-numeric values in Sales, Quantity, or Discount
df_clean = df_bronze.filter(
    (col("Sales").rlike("^[0-9]*\\.[0-9]+$")) & # Allow decimal numbers in Sales
    (col("Quantity").rlike("^[0-9]+$")) & # Allow only integers in Quantity
    (col("Discount").rlike("^[0-9]*\\.[0-9]+$")) # Allow decimal numbers in Discount
)

# Step 2: Cast the cleaned columns to appropriate numeric types
df_clean = df_clean.withColumn("Sales", col("Sales").cast("float")) \
    .withColumn("Quantity", col("Quantity").cast("int")) \
    .withColumn("Discount", col("Discount").cast("float"))

# Step 3: Drop rows with any null values in the numeric columns (after casting)
df_clean = df_clean.dropna(subset=["Sales", "Quantity", "Discount", "Profit"])

# Step 4: Group by Category and Region and perform aggregation
df_grouped = df_clean.groupBy("Category", "Region") \
    .agg(
        sum("Sales").alias("TotalSales"),
        sum("Profit").alias("TotalProfit"),
        sum("Quantity").alias("TotalQuantity"),
        sum("Discount").alias("TotalDiscount")
    )
```

```
# Step 5: Show the transformed data
df_grouped.show(5)
```

▶ (2) Spark Jobs

```
▶ df_clean: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 22 more fields]
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Category: string, Region: string ... 4 more fields]

+-----+-----+-----+-----+
| Category|Region| TotalSales| TotalProfit|TotalQuantity| TotalDiscount|
+-----+-----+-----+-----+
| Furniture|Canada|10595.279964447021|2613.240000000002| 78| 0.0|
| Technology| EMEA| 300854.583026886| 17494.443| 2259|189.1000056862831|
| Furniture| East| 205540.3473367691| 2501.816200000002| 2151| 90.6000018119812|
| Technology| Africa| 322367.0430994034|44129.492999999995| 2031|143.1999975964427|
| Technology| East| 264872.0816922188| 47439.5576| 1927|76.30000080168247|
+-----+-----+-----+-----+
only showing top 5 rows
```

5. Add a Profit Margin Column

▶ ✓ 12:25 PM (<1s)

6

```
from pyspark.sql.functions import col

# Add a Profit Margin Column
df_grouped = df_transformed.withColumn("ProfitMargin", (col("Profit") / col("Sales")) * 100)

# Show the result
df_grouped.show(5)
```

▶ (1) Spark Jobs

```
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 24 more fields]

+-----+
| 42655|AG-2011-1390|2011-08-16|2011-08-21|Standard Class| DH-3675| Duane Huffman|Home Office| Bejaia| Bejaia|Algeria| NULL
|Africa|Africa|OFF-SAN-10001237|Office Supplies| Paper|SanDisk Cards & E...|581.04| 12| 0|220.68| 35.32| Medium|3
60.3599999999996| 37.98017365775751|
| 48096|AG-2011-1440|2011-12-15|2011-12-20|Standard Class| DB-3210| Dean Braden| Consumer|Constantine|Constantine|Algeria| NULL
|Africa|Africa|OFF-AVE-10004909|Office Supplies| Binders|Avery 3-Hole Punc...| 58.74| 2| 0| 20.52| 3.04| Medium|
38.22|34.933605501216725|
| 44980| AG-2011-380|2011-03-03|2011-03-09|Standard Class| CC-2370|Christopher Conant| Consumer| Algiers| Alger|Algeria| NULL
|Africa|Africa|FUR-ELD-10003131| Furniture| Furnishings|Eldon Frame, Durable|113.28| 1| 0| 29.43| 9.51| Medium|
83.85| 25.97987343071408|
| 47723|AG-2011-4410|2011-11-02|2011-11-04| First Class| JH-5985| Joseph Holt| Consumer| Bejaia| Bejaia|Algeria| NULL
|Africa|Africa|OFF-EAT-10003338|Office Supplies| Paper|Eaton Message Boo...| 24.72| 1| 0| 10.86| 4.29| High|
13.86| 43.93203866640217|
| 43718| AG-2011-450|2011-11-07|2011-11-11|Standard Class| BC-1125| Becky Castell|Home Office| Algiers| Alger|Algeria| NULL
|Africa|Africa|OFF-STO-10003604|Office Supplies| Fasteners|Stockwell Staples...| 10.92| 1| 0| 4.14| 1.05| Medium|
6.78| 37.912086424551|
```

6. Remove Duplicate Rows based on 'Category' and 'Region'

```
▶ ✓ 12:25 PM (1s) 7

# Remove Duplicate Rows based on 'Category' and 'Region'
df_grouped = df_transformed.dropDuplicates(["Category", "Region"])

df_grouped.show(5)

▶ (2) Spark Jobs
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]
+-----+
| 44980| AG-2011-380|2011-03-03|2011-03-09|Standard Class| CC-2370|Christopher Conant|Consumer| Algiers| Algiers| Algiers| Alg...
ria| NULL|Africa| Africa|FUR-ELD-10003131|Furniture| Furnishings|Eldon Frame, Durable|113.28| 1| 0| 29.43| 9.51|
Medium| 83.85|
| 46421| CA-2011-4370|2011-09-23|2011-09-30|Standard Class| DW-3540| Don Weiss|Consumer| Toronto| Ontario| Can...
ada| NULL|Canada| Canada|FUR-DEF-10000622|Furniture| Furnishings|Deflect-O Light B...| 18.99| 1| 0| 6.81| 1.41|
Medium| 12.18|
| 1351|MX-2011-104752|2011-03-11|2011-03-16|Standard Class| MM-18055| Michelle Moray|Consumer|Consolacion del Sur|Pinar del R...
Cuba| NULL| LATAM| Caribbean| FUR-FU-10000885|Furniture| Furnishings|Eldon Light Bulb,...| 44.46| 3| 0| 8.4| 3.41|
| Medium| 36.06|
| 33942|CA-2011-131002|2011-09-07|2011-09-12| Second Class| TB-21400| Tom Boeckenhauer|Consumer| Tulsa| Oklahoma|United Sta...
tes| 74133| US| Central| FUR-FU-10004270|Furniture| Furnishings|"Executive Impress...| 57.69| 3| 0| 23.6529| 4.22|
High| 34.037099999999995|
| 22308| ID-2014-20205|2014-10-22|2014-10-27|Standard Class| JC-15340| Jasper Cacioppo|Consumer| Lahore| Punjab| Pakis...
tan| NULL| APAC|Central Asia| FUR-FU-10000394|Furniture| Furnishings|Rubbermaid Photo ...|78.048| 2| 0.2| 26.328| 5.82|
Medium| 51.72|
+-----+
only showing top 5 rows
```

7. Rename Columns for Clarity

```
▶ ✓ 12:25 PM (<1s) 8

#Rename Columns for Clarity
df_grouped = df_transformed.withColumnRenamed("Sales", "TotalSales") \
    .withColumnRenamed("Profit", "TotalProfit")

df_grouped.show(5)

▶ (1) Spark Jobs
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]
+-----+
| 42655|AG-2011-1390|2011-08-16|2011-08-21|Standard Class| DH-3675| Duane Huffman|Home Office| Bejaia| Bejaia| Algeria| ...
|Africa|Africa|OFF-SAN-10001237|Office Supplies| Paper|SanDisk Cards & E...| 581.04| 12| 0| 220.68| 35.32| NULL
Medium|360.3599999999996|
| 48096|AG-2011-1440|2011-12-15|2011-12-20|Standard Class| DB-3210| Dean Braden| Consumer|Constantine|Constantine|Algeria| ...
|Africa|Africa|OFF-AVE-10004909|Office Supplies| Binders|Avery 3-Hole Punc...| 58.74| 2| 0| 20.52| 3.04| NULL
Medium| 38.22|
| 44980| AG-2011-380|2011-03-03|2011-03-09|Standard Class| CC-2370|Christopher Conant|Consumer| Algiers| Algiers| Algeria| ...
|Africa|Africa|FUR-ELD-10003131| Furniture| Furnishings|Eldon Frame, Durable| 113.28| 1| 0| 29.43| 9.51| NULL
Medium| 83.85|
| 47723|AG-2011-4410|2011-11-02|2011-11-04| First Class| JH-5985| Joseph Holt| Consumer| Bejaia| Bejaia| Algeria| ...
|Africa|Africa|OFF-EAT-10003338|Office Supplies| Paper|Eaton Message Boo...| 24.72| 1| 0| 10.86| 4.29| NULL
High| 13.86|
| 43718| AG-2011-450|2011-11-07|2011-11-11|Standard Class| BC-1125| Becky Castell|Home Office| Algiers| Algiers| Algeria| ...
|Africa|Africa|OFF-STO-10003604|Office Supplies| Fasteners|Stockwell Staples...| 10.92| 1| 0| 4.14| 1.05| NULL
Medium| 6.78|
+-----+
```

8. Add a Year Column

```
▶ ✓ 12:25 PM (<1s) 9
from pyspark.sql.functions import col, year, to_date

# Add a Year Column
df_grouped = df_bronze.withColumn("Year", year(to_date(col("Order Date"), "MM/dd/yyyy")))

# Show the result
df_grouped.show(5)

▶ (1) Spark Jobs
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]
+-----+-----+
| 48312|AE-2011-9160|2011-10-03|2011-10-07|Standard Class| PO-8865|Patrick O'Donnell| Consumer| Ajman| 'Ajman|United Arab E
mirates| NULL| EMEA| EMEA|TEC-EPS-10004171| Technology| Machines|Epson Calculator,...|78.408| 6| 0.7| -88.992|
3.87| Medium|2011|
| 47297|AE-2013-1130|2013-10-14|2013-10-14| Same Day| EB-4110| Eugene Barchas| Consumer|Ras al Khaymah|Ra's Al Khaymah|United Arab E
mirates| NULL| EMEA| EMEA|OFF-ACC-10004278|Office Supplies| Fasteners|Accos Paper Clips...| 4.248| 1| 0.7| -4.692|
0.1| High|2013|
| 43923|AE-2014-3830|2014-12-13|2014-12-19|Standard Class| GH-4665| Greg Hansen| Consumer|Ras al Khaymah|Ra's Al Khaymah|United Arab E
mirates| NULL| EMEA| EMEA|OFF-AVE-10000357|Office Supplies| Binders|Avery Binder Cove...| 3.159| 1| 0.7| -4.971|
0.25| Medium|2014|
| 43926|AE-2014-3830|2014-12-13|2014-12-19|Standard Class| GH-4665| Greg Hansen| Consumer|Ras al Khaymah|Ra's Al Khaymah|United Arab E
mirates| NULL| EMEA| EMEA|TEC-MOT-10001535| Technology| Phones|Motorola Headset,...|95.796| 4| 0.7|-156.564|
6.73| Medium|2014|
| 42656|AG-2011-1390|2011-08-16|2011-08-21|Standard Class| DH-3675| Duane Huffman|Home Office| Bejaia| Bejaia|
Algeria| NULL|Africa|Africa|OFF-IBI-10000440|Office Supplies| Binders|Ibico Binder Cove...| 26.22| 2| 0| 1.8|
2.41| Medium|2011|
```

9. Filter the Data for a Specific Year:

```
▶ ✓ 12:25 PM (1s) 10
from pyspark.sql.functions import col, year, to_date, sum
#Filter the Data for a Specific Year

# Step 1: Add the Year Column to the Original DataFrame
df = df_bronze.withColumn("Year", year(to_date(col("Order Date"), "MM/dd/yyyy")))

# Step 2: Perform the aggregation including 'Year'
df_transformed = df.groupBy("Category", "Region", "Year") \
    .agg(
        sum("Sales").alias("TotalSales"),
        sum("Profit").alias("TotalProfit"),
        sum("Quantity").alias("TotalQuantity"),
        sum("Discount").alias("TotalDiscount")
    )

# Step 3: Filter the Data for a Specific Year
df_grouped = df_transformed.filter(col("Year") == 2012)

# Show the result
df_grouped.show(5)

▶ (2) Spark Jobs
```

▶ (2) Spark Jobs

```
▶ df: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Category: string, Region: string ... 5 more fields]
▶ df_transformed: pyspark.sql.dataframe.DataFrame = [Category: string, Region: string ... 5 more fields]

+-----+-----+-----+-----+-----+
| Category| Region|Year|TotalSales|      TotalProfit|TotalQuantity|      TotalDiscount|
+-----+-----+-----+-----+-----+
| Furniture|Oceania|2012|100519.008| 8623.81800000001|       607.0|26.19999999999978|
| Technology| Africa|2012| 64734.582| 6320.74199999999|       404.0|36.6999999999996|
| Technology| North|2012|126353.563|25098.86300000005|       809.0| 7.37799999999993|
| Furniture| Canada|2012|    1600.68|        290.19|       16.0|        0.0|
| Technology|Oceania|2012| 89761.767| 14203.827|       688.0|23.49999999999986|
+-----+-----+-----+-----+-----+
only showing top 5 rows
```

10. Sort Data by Total Sales:

2025 PM (1s)

11

Python

```
#Sort Data by Total Sales
df_grouped = df_transformed.orderBy(col("TotalSales").desc())
df_grouped.show()
```

▶ (2) Spark Jobs

```
▶ df_grouped: pyspark.sql.dataframe.DataFrame = [Category: string, Region: string ... 5 more fields]

+-----+-----+-----+-----+-----+
| Category| Region|Year|TotalSales|      TotalProfit|TotalQuantity|      TotalDiscount|
+-----+-----+-----+-----+-----+
| Technology|Central|2014|343261.7189199996| 46266.4532199999|       2600.0|93.5599999999997|
| Office Supplies|Central|2014| 313521.116999999|36457.12840000016|       9564.338|406.100000000015|
| Technology|Central|2013| 281501.66124| 39571.08374000002|2230.779999999997| 76.5000000000003|
| Furniture|Central|2014|281108.8896999994|15576.57130000007|       2778.252|129.519999999995|
| Office Supplies|Central|2013|249453.5420000004|36849.0073999995|       7735.108|285.8000000000035|
| Technology|Central|2012|237291.8116200002| 31373.85432|       1791.0|61.1420000000001|
| Furniture|Central|2013| 232113.698| 20804.504|       2187.344|103.069999999987|
| Technology| South|2014| 208820.03832|16919.97712000003|       1617.0| 61.7320000000001|
| Furniture|Central|2012|183778.9994999998| 5231.08480000003|       1876.94| 85.139999999999|
| Office Supplies|Central|2012| 180068.292|26988.42259999998| 6003.08399999999|198.4000000000006|
| Office Supplies|Central|2011|177982.1069999996| 21178.898|       5032.1|215.5880000000008|
| Technology|Central|2011|176379.6888599997| 18323.03876|       1370.0|47.11800000000016|
| Furniture| South|2014| 168929.8605| 8636.21280000001|2001.168000000001| 78.6999999999996|
| Office Supplies| South|2014|166334.9320000003| 25971.4154|       5738.816|235.0939999999994|
| Technology| North|2014| 164019.24052| 31386.99052|       1225.0|17.2039999999986|
| Furniture|Central|2011|162401.9760999997|13228.07560000002|       1773.484|        77.86|
| Technology| South|2013| 154303.72908| 17870.0951799999|       1288.56|        54.104|
+-----+-----+-----+-----+-----+
```

11. Save Transformed Data in Silver Zone:

12

Python   

```
#Add a business logic
df_final = df_grouped.withColumn("HighDiscount", col("TotalDiscount") > 50)
df_final.show(5)

▶ (2) Spark Jobs
```

```
▶ df_final: pyspark.sql.dataframe.DataFrame = [Category: string, Region: string ... 6 more fields]
```

Category	Region	Year	TotalSales	TotalProfit	TotalQuantity	TotalDiscount	HighDiscount
Technology	Central	2014	343261.7189199996	46266.4532199999	2600.0	93.55999999999997	true
Office Supplies	Central	2014	313521.1169999999	36457.12840000016	9564.338	406.1000000000015	true
Technology	Central	2013	281501.66124	39571.0837400002	2230.7799999999997	76.5000000000003	true
Furniture	Central	2014	281108.8896999994	15576.57130000007	2778.252	129.5199999999995	true
Office Supplies	Central	2013	249453.5420000004	36849.00739999995	7735.108	285.80000000000035	true

only showing top 5 rows

13

```
#Save Transformed Data in Silver Zone
silver_zone_path = "/mnt/silver/superstore"
df_final.write.format("delta").mode("overwrite").save(silver_zone_path)

# Save transformed data as a table in the catalog (Silver Zone)
df_final.write.format("delta").mode("overwrite").saveAsTable("transformed_data")

▶ (20) Spark Jobs
```

14

```
▶ dbutils.notebook.exit(silver_zone_path)
```

```
Notebook exited: /mnt/silver/superstore
```

Perform Analysis:

1. Top 5 Regions by Total Sales:

12:25 PM (3s) 1 SQL

```
%sql
--Top 5 Regions by Total Sales
SELECT Region, SUM(TotalSales) AS Sales
FROM transformed_data
GROUP BY Region
ORDER BY Sales DESC;
```

(3) Spark Jobs

_sqldf: pyspark.sql.DataFrame = [Region: string, Sales: double]

Table Visualization 1 +

	Region	Sales
1	Central	2818863.50194
2	South	1598168.72088
3	North	1248165.60252
4	Oceania	1100184.61200000...
5	Southeast Asia	884423.1689999999
6	North Asia	848309.7810000001
7	EMEA	806161.311
8	Africa	783773.211
9	Central Asia	752826.567
...

2. Top Categories by Profit Margin:

12:25 PM (1s) 2 SQL

```
%sql
--Top Categories by Profit Margin
SELECT Category, AVG(TotalProfit / TotalSales * 100) AS AvgProfitMargin
FROM transformed_data
GROUP BY Category
ORDER BY AvgProfitMargin DESC;
```

(2) Spark Jobs

_sqldf: pyspark.sql.DataFrame = [Category: string, AvgProfitMargin: double]

Table Visualization 1 +

	Category	AvgProfitMargin
1	Office Supplies	15.09414189794695
2	Technology	14.947268833662447
3	Furniture	8.49766003762561

3 rows | 0.69 seconds runtime Refreshed 49 minutes ago

This result is stored as _sqldf and can be used in other Python and SQL cells.

3. Year-over-Year Sales Growth:

```
▶ ✓ 12:25 PM (1s) 3  
  
%sql  
--Year-over-Year Sales Growth  
SELECT Region, Year, SUM(TotalSales) AS Sales  
FROM transformed_data  
GROUP BY Region, Year  
ORDER BY Region, Year;
```

▶ (2) Spark Jobs

```
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Region: string, Year: integer ... 1 more field]
```

	Region	Year	Sales
1	Africa	2011	127187.273999999...
2	Africa	2012	144480.705
3	Africa	2013	229068.792000000...
4	Africa	2014	283036.440000000...
5	Canada	2011	8509.11
6	Canada	2012	16096.800000000...
7	Canada	2013	19161.149999999...
8	Canada	2014	23161.11
9	Caribbean	2011	57043.428960000...

4. Regions with Highest Discounts:

```
▶ ✓ 12:25 PM (1s) 4  
  
%sql  
--Regions with Highest Discounts  
SELECT Region, SUM(TotalDiscount) AS TotalDiscount  
FROM transformed_data  
GROUP BY Region  
ORDER BY TotalDiscount DESC;
```

▶ (2) Spark Jobs

```
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Region: string, TotalDiscount: double]
```

	Region	TotalDiscount
1	Central	1779.7980000000018
2	West	1268.2180000000003
3	South	1115.8319999999997
4	EMEA	986.1000000000007
5	Southeast Asia	851.33
6	East	723.8940000000001
7	Africa	718.7999999999998
8	Oceania	534.1000000000003
9	North	459.6299999999998
10	Caribbean	229.4200000000002

5. Most Profitable Category-Region Pairs:

The screenshot shows the Databricks SQL interface. At the top, there's a header with a back arrow, a green checkmark, and the text "12:25 PM (1s)". On the right, there are icons for "SQL", a red star, a copy button, and three dots. Below the header is a code editor window containing the following SQL query:

```
%sql  
--Most Profitable Category-Region Pair  
SELECT Category, Region, SUM(TotalProfit) AS TotalProfit  
FROM transformed_data  
GROUP BY Category, Region  
ORDER BY TotalProfit DESC;
```

Below the code editor is a section titled "(2) Spark Jobs". Under this, there's a table titled "Table" with a single row labeled "Visualization 1". The table displays the results of the query:

	Category	Region	TotalProfit
1	Technology	Central	135534.43004
2	Office Supplies	Central	121473.45640000001
3	Technology	North	99272.49152
4	Technology	North Asia	72471.015
5	Office Supplies	South	67105.35949999999
6	Office Supplies	North	64403.39300000004
7	Technology	Central Asia	56439.975
8	Furniture	Central	54840.23570000002
9	Technology	Oceania	54734.02200000004

Link Azure Databricks to Azure Data Factory:

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has a "Linked services" section selected. The main area shows a list of existing linked services:

Name	Type
AzureBlobStorage1	Azure Blob Storage
AzureDataLakeStorage1	Azure Data Lake Storage Gen2

To the right, there's a "New linked service" dialog box. The "Compute" tab is selected. It lists several options with icons:

All	Azure	Compute
Azure Batch	Azure Data Factory	Azure Data Lake Analytics
Azure Databricks	Azure Function	Azure HDInsight

At the bottom of the dialog, there are "Continue" and "Cancel" buttons.

Generate New Databricks Token:

The screenshot shows the Microsoft Azure Databricks Settings page. A modal window titled "Generate New Token" is displayed, stating "Your token has been created successfully." Below the message, the token value "dapi6868a344d4ef379e88fb260a6ba81fac" is shown in a text input field with a checked "Copied" button. A note below the input field says, "⚠ Make sure to copy the token now. You won't be able to see it again." At the bottom right of the modal is a "Done" button. The background shows the "User" section of the settings, which includes "Identity and access", "Compute", "Development", "Notifications", and "Advanced". On the left sidebar, there are sections for "Workspace", "Recents", "Catalog", "Workflows", "Compute", "Data Engineering", "Job Runs", "Machine Learning", "Playground", "Experiments", "Features", "Models", "Serving", and "Partner Connect".

The screenshot shows the Microsoft Azure Data Factory Linked Services page. The left sidebar lists various factory settings like General, Factory settings, Connections, Source control, Author, Security, and others. The "Connections" section is currently selected. The main area displays a table of linked services, showing two items: "AzureBlobStorage1" (Type: Azure Blob Storage) and "AzureDataLakeStorage1" (Type: Azure Data Lake Storage Gen2). To the right of the table, a form is open for creating a new linked service to "Azure Databricks". The form fields include:

- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Account selection method:** From Azure subscription (selected)
- Azure subscription:** MML Learners (2a3c6418-97b9-4d96-a24b-2c2d7633d375)
- Databricks workspace:** Myworkspacehexa
- Select cluster:** New job cluster (selected)
- Databrick Workspace URL:** https://adb-1727980957508452.12.azuredatabricks.net
- Authentication type:** Access Token (selected)
- Access token:** (Input field containing a redacted token value)

At the bottom of the form are "Create" and "Back" buttons, along with "Test connection" and "Cancel" buttons at the very bottom right.

Add all the notebooks to the existing pipeline and run the pipeline:

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. On the left, the 'Factory Resources' sidebar is open, showing 'Pipelines' and 'Project 1 pipeline' selected. The main area displays the 'Project 1 pipeline' with the following activities connected sequentially:

- Copy data to Delta Lake Storage
- Notebook (Ingestion Tasks)
- Notebook (Transformation and Loading)
- Notebook (Analysis)

The pipeline has a status of 'Validated'.

The screenshot shows the Databricks Job Runs interface. The left sidebar shows 'Job Runs' selected. The main area displays the details for a specific task run:

Task run

Details

- Job ID: 805842802221705
- Task run ID: 335072940285952
- Run as: azuser2356_mml.local
- Started: 12/16/2024, 12:30:41 PM
- Ended: 12/16/2024, 12:37:34 PM
- Duration: 6m 53s
- Status: Succeeded

Output

```
%python
#Mount Azure Blob Storage

# Fetching the csv file from the blob storage
storage_account_name = "sourceaccount1hexa"
container_name = "superstore-data"
storage_account_key = "ooZRQBCJ9RAwKPvv5VvKprjv3UlGndcovcyJcjT4dVN8DbxojxsSgLojGldlb400ZMJD1AqFqrY2
+ASlj3lgD0=="

# Unmount the directory if it is already mounted
if any(mount.mountPoint == "/mnt/superstore" for mount in dbutils.fs.mounts()):
    dbutils.fs.unmount("/mnt/superstore")

# Mount Blob Storage
dbutils.fs.mount(
    source=f"wasbs://{(container_name)}@{(storage_account_name)}.blob.core.windows.net",
    mount_point="/mnt/superstore",
    extra_configs={
        f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": storage_account_key
    }
)
```

Notebook

/Users/azuser2356_mml.local@techademy.com/Project 1

Screenshot of the Microsoft Azure Databricks interface showing a job run details page.

The left sidebar shows the navigation menu with "Job Runs" selected. The main content area displays the output of a job run named "ADF_SuperstoreADHexa_CopyPipeline_xgw_transformation_and_loading_tasks_2837af72-bff1-450c-b107-af7b1c2ffef0 run".

The output pane shows two code snippets:

```
✓ 5.01 seconds 1
bronze_path = "/mnt/bronze/superstore"
df_bronze = spark.read.format("delta").load(bronze_path)

df_bronze: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 22 more fields]
```

```
✓ 18.52 seconds 2
#Add Total Cost Column
from pyspark.sql.functions import col

df_transformed = df_bronze.withColumn("TotalCost", col("Sales") - col("Profit"))
df_transformed.show(5)

df_transformed: pyspark.sql.dataframe.DataFrame = [Row ID: integer, Order ID: string ... 23 more fields]
```

The right pane displays the "Task run" details, including:

- Job ID: 837006909362069
- Task run ID: 114678943446306
- Run as: azuser2356_mml.local
- Started: 12/16/2024, 12:38:00 PM
- Ended: 12/16/2024, 12:40:10 PM
- Duration: 2m 9s
- Status: Succeeded

The status bar at the bottom indicates the date and time as 16-12-2024 12:40.

Screenshot of the Microsoft Azure Databricks interface showing a job run details page.

The left sidebar shows the navigation menu with "Job Runs" selected. The main content area displays the output of a job run named "ADF_SuperstoreADHexa_CopyPipeline_xgw_analysis_216d7184-aa67-4a47-96d8-88f579881b53 run".

The output pane shows a SQL query and its results:

```
%sql
--Top 5 Regions by Total Sales
SELECT Region, SUM(TotalSales) AS Sales
FROM transformed_data
GROUP BY Region
ORDER BY Sales DESC;
```

The results are displayed in a table:

Region	Sales
Central	2818863.50194
South	1598168.72088
North	1248165.60252
Oceania	1100184.61200000...
Southeast Asia	884423.1689999999
North Asia	848309.7810000001
EMEA	806161.311

The right pane displays the "Task run" details, including:

- Job ID: 584203222029249
- Task run ID: 764515447691552
- Run as: azuser2356_mml.local
- Started: 12/16/2024, 12:40:17 PM
- Ended: 12/16/2024, 12:42:12 PM
- Duration: 1m 54s
- Status: Succeeded

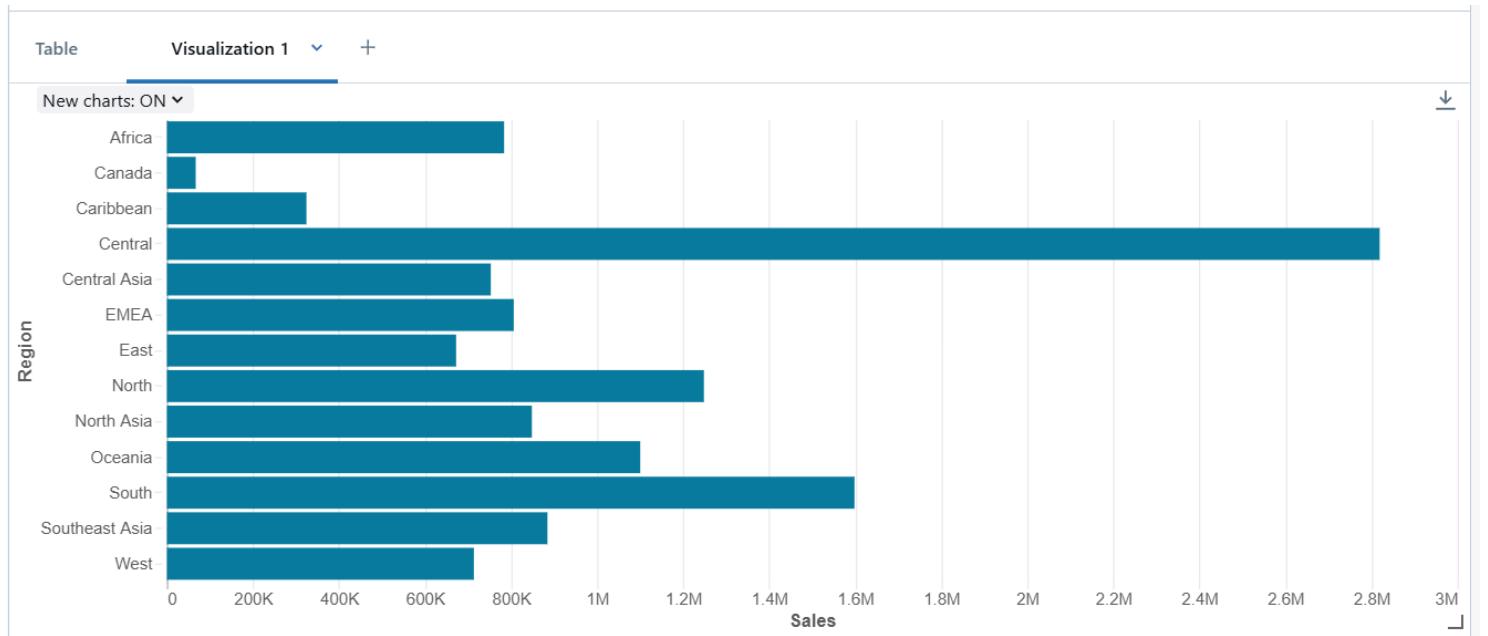
The status bar at the bottom indicates the date and time as 16-12-2024 12:42.

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. On the left, the 'Factory Resources' sidebar is open, showing a single pipeline named 'CopyPipeline_xgw'. The main workspace displays a pipeline diagram with four activities: 'Copy data' (copy_data_to_delta_lake_storage), 'Notebook' (ingestion_task), 'Notebook' (transformation_and_loading_tasks), and 'Notebook' (analysis). The 'Copy data' activity is connected to the 'ingestion_task' notebook. The 'ingestion_task' notebook is connected to the 'transformation_and_loading_tasks' notebook, which in turn connects to the 'analysis' notebook. All activities show a green checkmark indicating success. Below the diagram, the 'Output' tab is selected, showing a table of pipeline run details:

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
analysis	Succeeded	Notebook	12/16/2024, 12:40:13 P	2m 16s	AutoResolveIntegration
transformation_and_loading_t...	Succeeded	Notebook	12/16/2024, 12:37:58 P	2m 15s	AutoResolveIntegration
ingestion_task	Succeeded	Notebook	12/16/2024, 12:30:37 P	7m 20s	AutoResolveIntegration
copy_data_to_delta_lake_stor...	Succeeded	Copy data	12/16/2024, 12:30:22 P	14s	AutoResolveIntegration

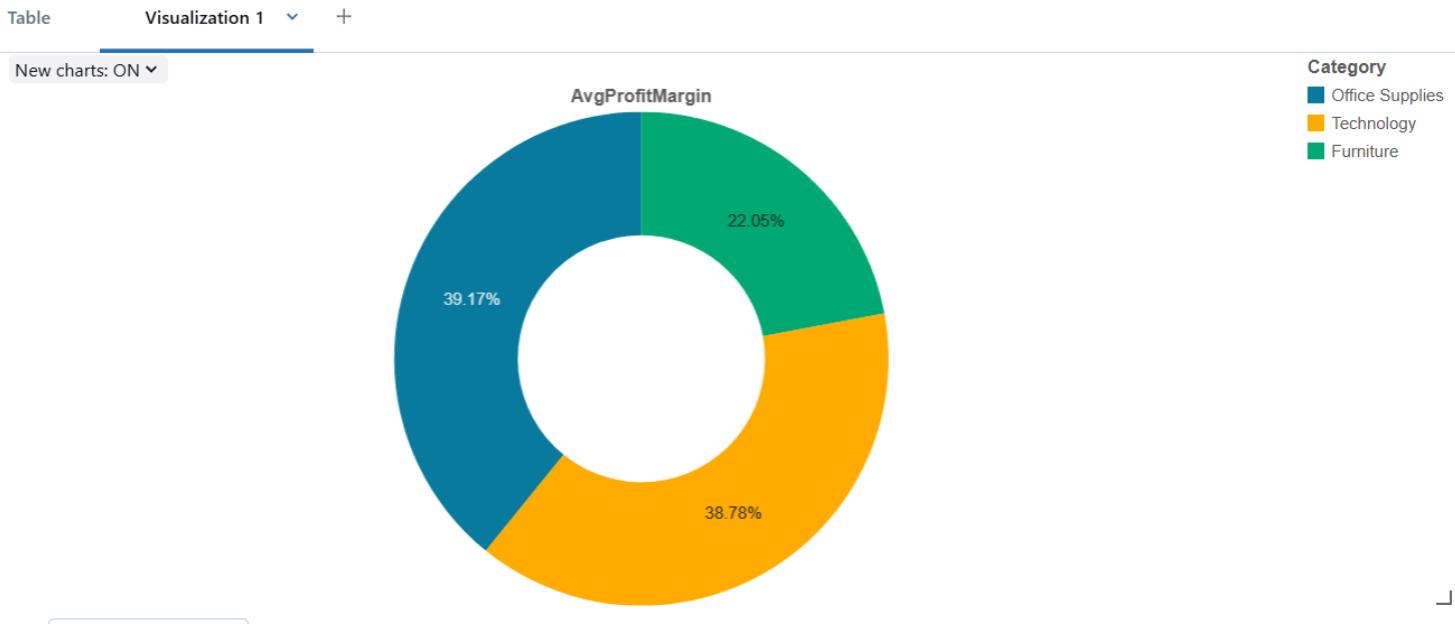
11. Analysis & Visualization:

1. Sales vs Region:

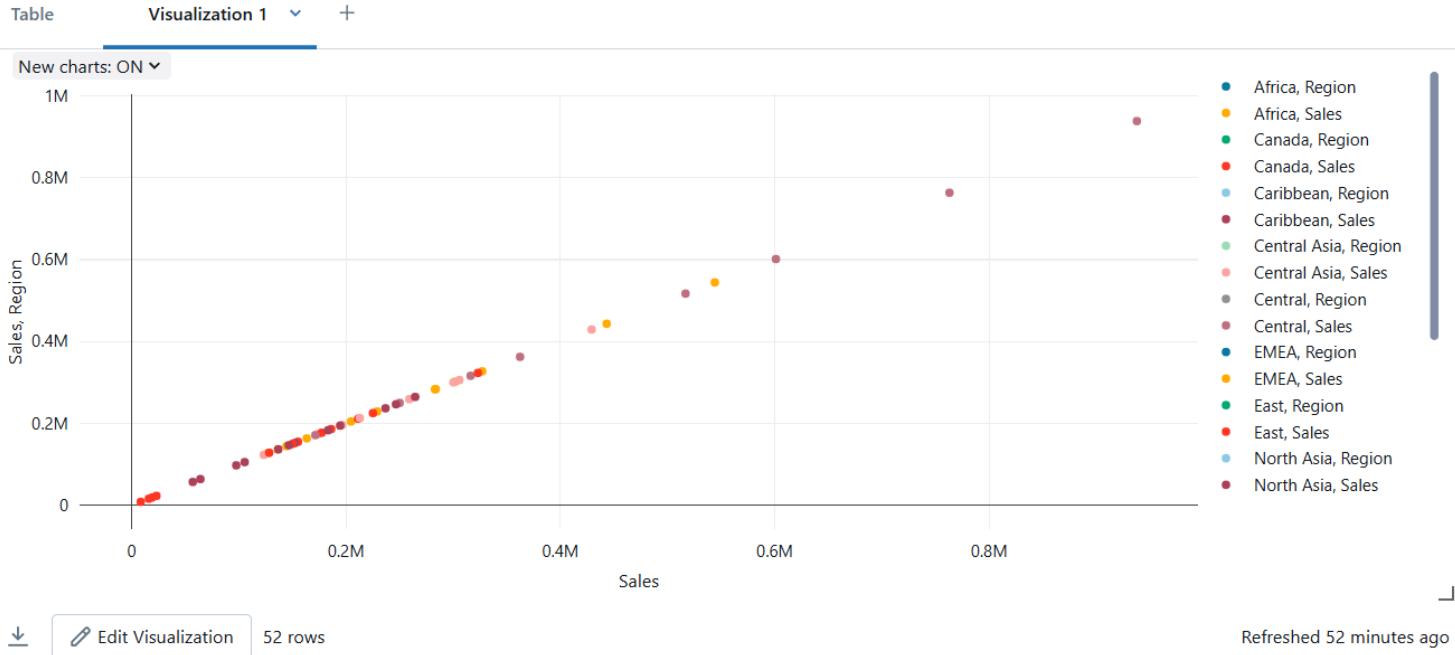


2. Average Profit margin:

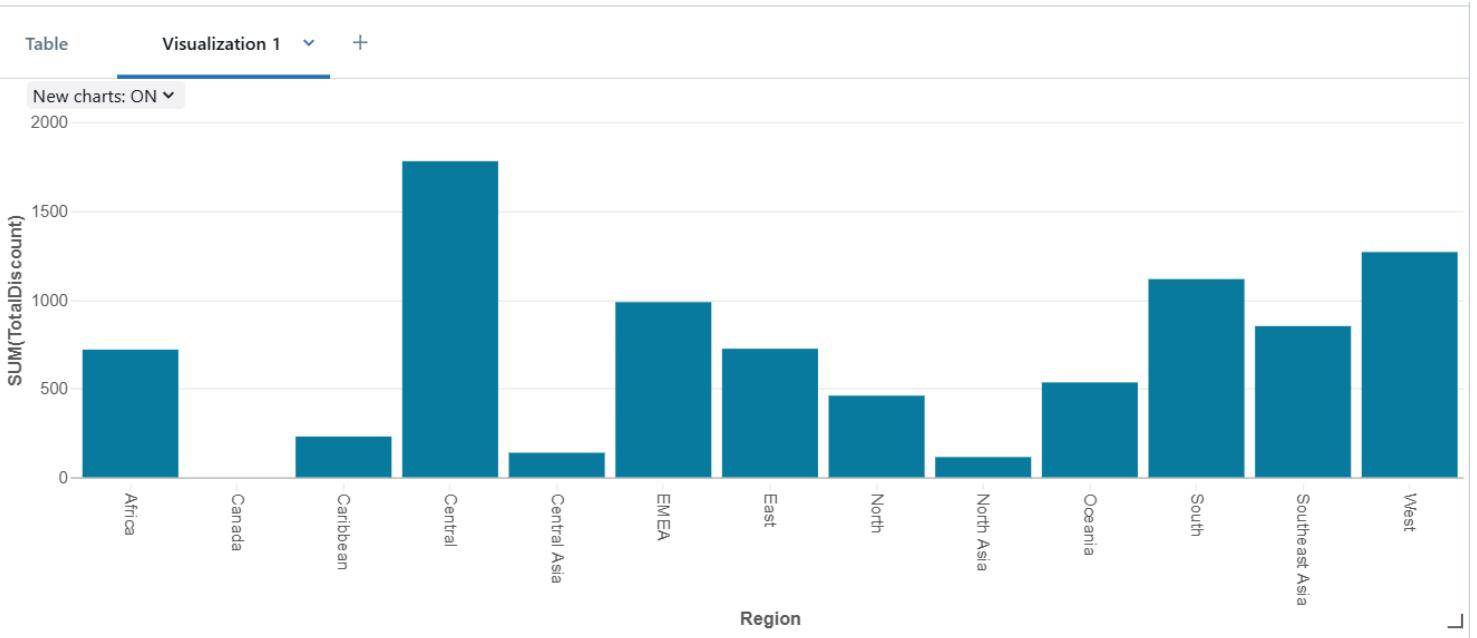
▶ `_sqldf: pyspark.sql.dataframe.DataFrame = [Category: string, AvgProfitMargin: double]`



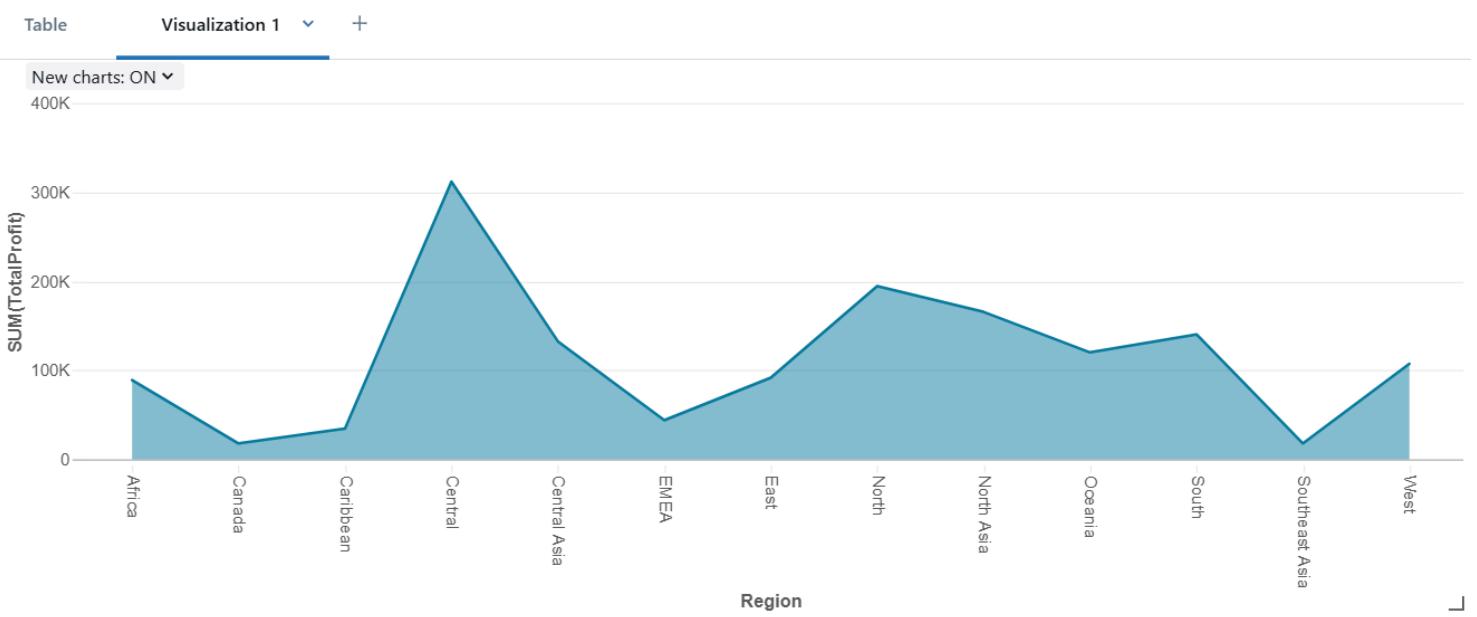
3. Sales vs Sales, region:



4. Region vs Sum(TotalDiscount):



5. Region vs SUM(TotalProfit):



Submitted by:
Eswara Venkata Sai Raja