

Large Scale Data Processing with ETL Operations using PySpark

Table of Contents

- Project statement
- Project Overview
- Prerequisites
- Project Requirements
- Execution Overview
- Tools/Technology Used in the Project
- Source Data Files
- Implementation -Tasks performed
- Steps on practical implementation on the Azure portal
- Successful output generated
- Conclusion

Project statement:

Implement a project that leverages Azure Databricks and PySpark for large-scale data processing. Perform ETL operations on massive datasets using PySpark, and utilize Databricks clusters for scalability.

Project Overview:

This project implements a large-scale data processing solution by leveraging the power of Azure Databricks and PySpark. The core of this initiative is to perform robust ETL (Extract, Transform, Load) operations on massive datasets.

The process involves extracting raw data from various sources and then utilizing custom PySpark scripts within the Azure Databricks environment to execute complex transformations, data cleansing, and aggregations. The key to handling massive data volumes efficiently is the strategic use of scalable Databricks clusters, which ensures high-performance processing. Once transformed, the high-quality data is loaded into a final destination like Azure Data Lake or a SQL database, making it ready for advanced analytics and reporting.

Prerequisites:

1. Azure Subscription: Have an active Azure subscription for resource management.
2. Azure Databricks :Set up an Azure Databricks workspace for Spark processing.
3. Azure Storage Account: Create a storage account in order to store CSV file .
4. Data Source: Ensure availability of the CSV file.
5. Databricks Cluster: Set up a Databricks cluster for Spark jobs.
6. Monitoring and Logging: Set up monitoring in Databricks.

Project Requirements:

1. Technical Infrastructure:

- Azure Databricks: Access to an Azure Databricks workspace for executing PySpark scripts.
- Storage Accounts: Azure storage (Blob storage or Data Lake) for intermediate and final data storage.

2. Data Sources and Destinations:

- Source Data Specifications: Identification of data sources (e.g., SQL databases, CSV files in storage accounts).
- ETL Logic: Develop PySpark scripts to perform data cleansing, transformation, filtering, and aggregation.

- Destination: Load the transformed data into a structured format (e.g., Parquet) in a designated destination container.

3. Development & Security:

- PySpark Environment: Setup for PySpark in Azure Databricks for data transformation scripting.
- Security: Implement secure connections to storage accounts from Databricks.

4. Performance and Scalability:

- Performance: The solution must be capable of scaling resources up or down based on the processing demands.
- Databricks Clusters: Configure a cluster with sufficient worker nodes and memory to process the massive dataset efficiently.

5. Monitoring:

- Job Monitoring: Utilize the Databricks UI and Spark UI to monitor the progress, performance, and health of the ETL jobs.

Execution Overview

1. Setup and Configuration:

- Provision the necessary Azure resources: Azure Databricks Workspace and Azure Storage Account.
- Create a Databricks cluster with an appropriate configuration for large-scale data processing.

2. Data Ingestion:

- Upload the source dataset (Online Retail Dataset CSV) to a container in the Azure Storage Account.

3. Azure Databricks Setup:

- Workspace Preparation: Initialize and configure the Azure Databricks workspace.
- PySpark Environment: Establish the PySpark environment within Databricks for data transformation scripting.

4. ETL Development with PySpark:

- Create a new Databricks notebook.
- Write PySpark code to read the raw CSV data into a DataFrame.
- Implement a series of transformations, including but not limited to:
 - Filtering out null values in critical columns.
 - Renaming columns for clarity.
 - Performing aggregations to summarize the data.

5. Data Loading:

- Write the final, transformed DataFrame to a destination container in the Azure Storage Account, preferably in an optimized format like Parquet for efficient querying.

6. Execution and Monitoring:

- Run the PySpark notebook on the configured Databricks cluster.
- Monitor the job's execution through the Databricks interface, analyzing the Spark jobs, stages, and tasks to ensure optimal performance.

7. Validation:

- Verify that the transformed data has been successfully written to the destination storage container and that the output is accurate.

Tools/Technology Used in the Project

- **Cloud Platform:** Microsoft Azure
- **Processing Engine:** Azure Databricks
- **Programming Language/Framework:** PySpark
- **Data Storage:** Azure Blob Storage / Azure Data Lake Storage
- **Development Interface:** Databricks Notebooks

Source Date Files

We will be using the [Online Retail Dataset](#) dataset from Kaggle.

This Dataset has 5,41,910 rows.

```
ers > iamesaq > Downloads > JupyterNotebooks > # data.csv > data
1 InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
2 536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010 8:26,2.55,17850,United Kingdom
3 536365,71053,WHITE METAL LANTERN,6,12/1/2010 8:26,3.39,17850,United Kingdom
4 536365,84406B,CREAM CUPID HEARTS COAT HANGER,8,12/1/2010 8:26,2.75,17850,United Kingdom
5 536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6,12/1/2010 8:26,3.39,17850,United Kingdom
6 536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6,12/1/2010 8:26,3.39,17850,United Kingdom
7 536365,22752,SET 7 BABUSHKA NESTING BOXES,2,12/1/2010 8:26,7.65,17850,United Kingdom
8 536365,21730,GLASS STAR FROSTED T-LIGHT HOLDER,6,12/1/2010 8:26,4.25,17850,United Kingdom
9 536366,22633,HAND WARMER UNION JACK,6,12/1/2010 8:28,1.85,17850,United Kingdom
10 536366,22632,HAND WARMER RED POLKA DOT,6,12/1/2010 8:28,1.85,17850,United Kingdom
```

Implementation -Tasks performed:

- Create a Azure Storage Account
- Upload the dataset to a container
- Set Up Azure Databricks Workspace.
- Develop PySpark Scripts for ETL Operations.
- Add ETL Operation Activity in the Pipeline.

- Schedule a Job execution.
- Monitor Job Performance.

Steps on practical implementation on the Azure portal

Step 1: Set Up Azure Storage Account

1. Create a Data Factory Instance:

- Go to the Azure Portal and search for “Storage Account”.
- Click on "+ Create"
- Fill in the details like name, region, performance etc., and create the instance.

The screenshot shows the Microsoft Azure Storage accounts page. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, a breadcrumb trail shows 'Home > Storage accounts'. The main area displays a list of storage accounts with columns for Name, Type, Kind, Resource Group, and Location. Two accounts are listed:

Name	Type	Kind	Resource Group	Location
dbstoragetycc362sqj26w	Storage account	StorageV2	mrg-ETLProject	East Asia
hexaware1etlproject	Storage account	StorageV2	DEProject	East Asia

Microsoft Azure

Home > Storage accounts >

Create a storage account

[Basics](#) [Advanced](#) [Networking](#) [Data protection](#) [Encryption](#) [Tags](#) [Review + create](#)

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Resource group * [Create new](#)

Instance details

Storage account name *

Region * [Deploy to an Azure Extended Zone](#)

Primary service

Performance * Standard: Recommended for most scenarios (general-purpose v2 account)
 Premium: Recommended for scenarios that require low latency.

Redundancy * [View documentation](#)

Make read access to data available in the event of regional unavailability.

[Previous](#) [Next](#) [Review + create](#) [Give feedback](#)

Microsoft Azure

Home > Storage accounts >

Create a storage account

[Basics](#) [Advanced](#) [Networking](#) [Data protection](#) [Encryption](#) [Tags](#) [Review + create](#)

[View automation template](#)

Basics

Subscription	Azure for Students
Resource group	DEProject
Location	East Asia
Storage account name	hexaware1etlproject
Primary service	Azure Blob Storage or Azure Data Lake Storage Gen 2
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

Enable hierarchical namespace	Disabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

Security

Secure transfer	Enabled
Blob anonymous access	Disabled
Allow storage account key access	Enabled
Default to Microsoft Entra authorization in the Azure portal	Disabled
Minimum TLS version	Version 1.2

[Previous](#) [Next](#) [Create](#) [Give feedback](#)

Deployment is in progress

Deployment name: hexaware1etlproject_1756305782884
Subscription: Azure for Students
Resource group: DEProject

Start time: 8/27/2025, 8:13:57 PM
Correlation ID: a17b8563-baf6-457c-8a1b-63fabf87205

Your deployment is complete

Deployment name: hexaware1etlproject_1756305782884
Subscription: Azure for Students
Resource group: DEProject

Start time: 8/27/2025, 8:13:57 PM
Correlation ID: a17b8563-baf6-457c-8a1b-63fabf87205

2. Create Container in Azure Storage Account:

- After deployment, click on “Go to Resource”.
- Then click on “Data Storage” > “Containers” > “Add Container”
- Enter a container name and create 3 containers.

Name	Last modified	Anonymous access level	Lease state
Slogs	8/26/2025, 11:35:14 AM	Private	Available
destination	8/26/2025, 11:36:55 AM	Private	Available
raw	8/26/2025, 11:36:33 AM	Private	Available
source	8/26/2025, 11:36:49 AM	Private	Available

More events in the activity log → Dismiss all ▾

Successfully created storage container
Successfully created storage container 'destination'.
a few seconds ago

Successfully created storage container
Successfully created storage container 'source'.
a few seconds ago

Successfully created storage container
Successfully created storage container 'raw'.
a few seconds ago

Step 2: Upload the Dataset to a Storage Account.

1. Open the created Container:

- Click on “Upload”
- Select the Dataset and select the created container and upload
- Then dataset is uploaded to the Storage Account

Home > etlproject02storage | Containers >

raw Container

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive)

Showing all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
data.csv	8/26/2025, 11:37:51 AM	Hot (Inferred)	Block blob	43.47 MB	Available

2. To Access Dataset from ADLS:

- In Storage Account, click on "Security+Networking" > "Access Keys"
- There you will find 2 access keys, copy any one of the "Key" for later.

Microsoft Azure

Home > Storage accounts > hexaware1etlproject

Storage accounts

You are viewing a new version of Browse experience. Click here to access the old experience.

+ Create ⚡ Restore ...

Default Directory (mynameiseisa@gmail.onmicrosoft.com)

Name: hexaware1etlproject

dbstoragetycc362sqj26w

Showing 1 - 2 of 2. Display auto count:

hexaware1etlproject | Access keys

Storage account

Search

File shares

Queues

Tables

Security + networking

Networking

Front Door and CDN

Access keys

Shared access signature

Encryption

Microsoft Defender for Cloud

Data management

Set rotation reminder Refresh Give feedback

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account. Learn more about managing storage account access keys

Storage account name: hexaware1etlproject

key1 Rotate key

Last rotated: 8/27/2025 (0 days ago)

Key: [REDACTED]

Show

Connection string: [REDACTED]

Show

Step 3: Set Up Azure Databricks

1. Create a Databricks workspace:

- Search for Azure Databricks in Azure Portal
- Click on "+ Create" and fill the name, region, etc details and create a new workspace
- Launch the workspace after deployment.

Create an Azure Databricks workspace

Project Details

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

Subscription *

Resource group *

Instance Details

Workspace name *

Region *

Pricing Tier *

We selected the recommended pricing tier for your workspace. You can change the tier based on your needs.

Managed Resource Group name

Review + create < Previous Next : Networking >

Welcome to Databricks

Search data, notebooks, recents, and more...

Untitled Notebook 2025-08-27 21:52:51 /Users/mynameisesaq@gmail.com 2 hours ago Notebook

Step 4: Create a new Compute:

1. Create a Compute:

- In Azure Databricks, go to Compute in the left-panel.
- Click on “Create compute”
- Fill the details according to your need and create.

The screenshot shows the Databricks Compute configuration interface. On the left, a sidebar lists various workspace sections like Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, and Data Engineering. The 'Compute' section is currently selected. The main area displays the configuration for 'Esaq A's Cluster' created on 2025-08-27 at 21:46:07. The 'Configuration' tab is active, showing the 'General' section with fields for 'Compute name' (set to 'Esaq A's Cluster 2025-08-27 21:46:07') and 'Policy' (set to 'Unrestricted'). Below this is the 'Performance' section, which includes options for 'Machine learning', 'Databricks runtime' (set to '16.4 LTS (includes Apache Spark 3.5.2, Scala 2.12)'), and 'Photon acceleration'. It also specifies 'Node type' as 'Standard_D4ds_v5' with '16 GB Memory, 4 Cores'. There are checkboxes for 'Single node' and 'Terminate after 20 minutes of inactivity'. The 'Tags' section indicates 'No custom tags' and 'Automatically added tags'. To the right, a summary box provides details: 'Summary' (16 GB Memory, 4 Cores), 'Data access' (Unity Catalog), and 'Price' (2 DBU/h). At the top right, there are 'Terminate' and 'Edit' buttons.

2. Create a Notebook in Databricks:

- Go to your Databricks workspace and create a new notebook.
- Choose PySpark as the language.

3. Write PySpark Script:

- Implement your PySpark script. For example:

```
storage_account_name = "mydatabricksstorage12"
container_name = "raw-data"
secret_name =
"d8CceViUDRX70J9LvideVqWIQccNb35nug80sVLPxoTcTjbewDuDhpC0Hxt4aQI343pzb6
hM/KvY+AStQYK2zg=="
```

Data_Ingestion/read_data.ipynb

The screenshot shows a Databricks workspace interface with two code cells. The top cell (cell 1) contains Python code for mounting an Azure Storage container and listing its contents. The bottom cell (cell 2) contains SQL code for creating a database and loading data into a Delta table.

Cell 1: Python Code (read_data)

```
# Define Azure Storage details
storage_account_name = "etlproject02storage"
container_name = "raw"
mount_point = f"/mnt/{container_name}"
storage_account_key = "HUNrqEOTIMF8y/u2wKtmOlIrwB3G0BDA8KhB5uBjKhdcLzCv6eUMIrW9luGrwLnQFV+8vtrKUT+1ASTjJXMuA=="

# First try to umount if it exists
try:
    dbutils.fs.unmount(mount_point)
    print(f"Unmounted existing mount: {mount_point}")
except:
    print(f"No existing mount found at: {mount_point}")

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

# Check mount
display(dbutils.fs.ls(mount_point))

```

(2) Spark Jobs
No existing mount found at: /mnt/raw

Table	+ path	name	size	modificationTime
1	dbfs:/mnt/raw/data.csv	data.csv	45580638	1756188471000

Cell 2: SQL Code (read_data)

```
%sql
create database if not exists online_retail;
```

(1) Spark Jobs
2

```
file_path = "/mnt/raw/data.csv"

raw_df = spark.read \
    .format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load(file_path)

# Create Delta Table
raw_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.sales_raw")

# Display schema and sample data
print("Raw Layer Schema:")
raw_df.printSchema()
print("Sample of Raw Data:")
raw_df.show(5)
```

(4) Spark Jobs
root
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: integer (nullable = true)
|-- InvoiceDate: string (nullable = true)
|-- UnitPrice: double (nullable = true)
|-- CustomerID: integer (nullable = true)
|-- Country: string (nullable = true)

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON +

Search data, notebooks, recents, and more... my-databricks-workspace

Correct way
spark.table("online_retail.sales_raw").count()
display(spark.table("online_retail.sales_raw").limit(5))

(3) Spark Jobs

Table

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Count
548311	22309	TEA COSY RED STRIPE	6	3/30/2011 12:05	2.55	13728	United King
548311	21874	GIN AND TONIC MUG	12	3/30/2011 12:05	1.25	13728	United King
548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	3/30/2011 12:05	1.25	13728	United King
548311	48173C	DOORMAT BLACK FLOCK	2	3/30/2011 12:05	7.95	13728	United King
548311	48187	DOORMAT NEW ENGLAND	2	3/30/2011 12:05	7.95	13728	United King

5 rows | 2.98s runtime Refreshed 55 minutes ago

Data_Cleaning/cleaning.ipynb

Microsoft Azure databricks

+ New

Workspace Recents Catalog Jobs & Pipelines * Compute Data Engineering Job Runs AI/ML Playground Experiments Features Models Serving

File Edit View Run Help Python Tabs: ON + Last edit was 14 minutes ago

a) Filter out cancellations and invalid quantities/prices

from pyspark.sql.functions import col
raw_df = spark.read.table("online_retail.sales_raw")
source_df = raw_df.filter(~col("InvoiceNo").startswith("C")) \ .filter(col("Quantity") > 0) \ .filter(col("UnitPrice") > 0)
source_df.limit(20).display()

(1) Spark Jobs

raw_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

Table

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Count
548311	22309	TEA COSY RED STRIPE	6	3/30/2011 12:05	2.55	13728	United Kir
548311	21874	GIN AND TONIC MUG	12	3/30/2011 12:05	1.25	13728	United Kir
548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	3/30/2011 12:05	1.25	13728	United Kir
548311	48173C	DOORMAT BLACK FLOCK	2	3/30/2011 12:05	7.95	13728	United Kir
548311	48187	DOORMAT NEW ENGLAND	2	3/30/2011 12:05	7.95	13728	United Kir
548311	20685	DOORMAT RED RETROSPOT	2	3/30/2011 12:05	7.95	13728	United Kir
548311	22691	DOORMAT WELCOME SUNRISE	2	3/30/2011 12:05	7.95	13728	United Kir
548311	48194	DOORMAT HEARTS	2	3/30/2011 12:05	7.95	13728	United Kir
548311	48188	DOORMAT WELCOME PUPPIES	2	3/30/2011 12:05	7.95	13728	United Kir
548311	48111	DOORMAT 3 SMILEY CATS	2	3/30/2011 12:05	7.95	13728	United Kir
548312	84836	ZINC METAL HEART DECORATION	6	3/30/2011 12:12	1.25	16059	United Kir
548312	22917	HERB MARKER ROSEMARY	2	3/30/2011 12:12	0.65	16059	United Kir
548312	22916	HERB MARKER THYME	2	3/30/2011 12:12	0.65	16059	United Kir

b) Handle null CustomerIDs: Replace with 0

```

from pyspark.sql.functions import lit,when
source_df = source_df.withColumn("CustomerID",
    when(col("CustomerID").isNull(), lit(0))
    .otherwise(col("CustomerID").cast("int"))
)
source_df.display()

```

source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
548311	22309	TEA COSY RED STRIPE	6	3/30/2011 12:05	2.55	13728	United
548311	21874	GIN AND TONIC MUG	12	3/30/2011 12:05	1.25	13728	United
548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	3/30/2011 12:05	1.25	13728	United
548311	48173C	DOORMAT BLACK FLOCK	2	3/30/2011 12:05	7.95	13728	United
548311	48187	DOORMAT NEW ENGLAND	2	3/30/2011 12:05	7.95	13728	United
548311	20685	DOORMAT RED RETROSPOT	2	3/30/2011 12:05	7.95	13728	United
548311	22691	DOORMAT WELCOME SUNRISE	2	3/30/2011 12:05	7.95	13728	United
548311	48194	DOORMAT HEARTS	2	3/30/2011 12:05	7.95	13728	United
548311	48188	DOORMAT WELCOME PUPPIES	2	3/30/2011 12:05	7.95	13728	United
548311	48111	DOORMAT 3 SMILEY CATS	2	3/30/2011 12:05	7.95	13728	United
548312	84836	ZINC METAL HEART DECORATION	6	3/30/2011 12:12	1.25	16059	United
548312	22917	HERB MARKER ROSEMARY	2	3/30/2011 12:12	0.65	16059	United
548312	22916	HERB MARKER THYME	2	3/30/2011 12:12	0.65	16059	United
548312	22921	HERB MARKER CHIVES	2	3/30/2011 12:12	0.65	16059	United

c) Convert InvoiceDate to Timestamp and extract time parts

```

from pyspark.sql.functions import col, to_timestamp, year, month, dayofweek, hour
source_df = source_df.withColumn("InvoiceTimestamp",
    to_timestamp(col("InvoiceDate"), "M/d/yyyy H:mm")
) \
    .withColumn("Year", year(col("InvoiceTimestamp")) \
    .withColumn("Month", month(col("InvoiceTimestamp")) \
    .withColumn("DayOfWeek", dayofweek(col("InvoiceTimestamp")) \
    .withColumn("Hour", hour(col("InvoiceTimestamp")))
    .drop("InvoiceDate")
source_df.display()

```

source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 10 more fields]

InvoiceNo	StockCode	Description	Quantity	UnitPrice	CustomerID	Country	Invoice
548311	22309	TEA COSY RED STRIPE	6	2.55	13728	United Kingdom	2011-03-3
548311	21874	GIN AND TONIC MUG	12	1.25	13728	United Kingdom	2011-03-3
548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	1.25	13728	United Kingdom	2011-03-3
548311	48173C	DOORMAT BLACK FLOCK	2	7.95	13728	United Kingdom	2011-03-3
548311	20685	DOORMAT RED RETROSPOT	2	7.95	13728	United Kingdom	2011-03-3
548311	22691	DOORMAT WELCOME SUNRISE	2	7.95	13728	United Kingdom	2011-03-3
548311	48194	DOORMAT HEARTS	2	7.95	13728	United Kingdom	2011-03-3
548311	48188	DOORMAT WELCOME PUPPIES	2	7.95	13728	United Kingdom	2011-03-3
548311	48111	DOORMAT 3 SMILEY CATS	2	7.95	13728	United Kingdom	2011-03-3
548312	84836	ZINC METAL HEART DECORATION	6	1.25	16059	United Kingdom	2011-03-3
548312	22917	HERB MARKER ROSEMARY	2	0.65	16059	United Kingdom	2011-03-3

e) Add the TotalAmount/Revenue column

```

from pyspark.sql.functions import round
source_df = source_df.withColumn("TotalAmount",
                                 round(col("Quantity") * col("UnitPrice"), 2))
source_df.limit(20).display()

```

(1) Spark Jobs

source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 11 more fields]

#	InvoiceNo	StockCode	Description	Quantity	UnitPrice	CustomerID	Country	InvoiceTin
1	548311	22309	TEA COSY RED STRIPE	6	2.55	13728	United Kingdom	2011-03-30T
2	548311	21874	GIN AND TONIC MUG	12	1.25	13728	United Kingdom	2011-03-30T
3	548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	1.25	13728	United Kingdom	2011-03-30T
4	548311	48173C	DOORMAT BLACK FLOCK	2	7.95	13728	United Kingdom	2011-03-30T
5	548311	48187	DOORMAT NEW ENGLAND	2	7.95	13728	United Kingdom	2011-03-30T
6	548311	20685	DOORMAT RED RETROSPOT	2	7.95	13728	United Kingdom	2011-03-30T
7	548311	22691	DOORMAT WELCOME SUNRISE	2	7.95	13728	United Kingdom	2011-03-30T
8	548311	48194	DOORMAT HEARTS	2	7.95	13728	United Kingdom	2011-03-30T
9	548311	48188	DOORMAT WELCOME PUPPIES	2	7.95	13728	United Kingdom	2011-03-30T
10	548311	48111	DOORMAT 3 SMILEY CATS	2	7.95	13728	United Kingdom	2011-03-30T
11	548312	84836	ZINC METAL HEART DECORATION	6	1.25	16059	United Kingdom	2011-03-30T
12	548312	22917	HERB MARKER ROSEMARY	2	0.65	16059	United Kingdom	2011-03-30T
13	548312	22916	HERB MARKER THYME	2	0.65	16059	United Kingdom	2011-03-30T
14	548312	22921	HERB MARKER CHIVES	2	0.65	16059	United Kingdom	2011-03-30T

↓ 20 rows | 0.53s runtime

Refreshed 34 minutes ago

f) Cast InvoiceNo to integer

```

source_df = source_df.withColumn("InvoiceNoInt",
                                 when(col("InvoiceNo").rlike("[0-9]+$"),
                                      col("InvoiceNo").cast("int")
                                     ).otherwise(lit(None)))
source_df.display()

```

(1) Spark Jobs

source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 12 more fields]

#	InvoiceNo	StockCode	Description	Quantity	UnitPrice	CustomerID	Country	Invoice
1	548311	22309	TEA COSY RED STRIPE	6	2.55	13728	United Kingdom	2011-03-3
2	548311	21874	GIN AND TONIC MUG	12	1.25	13728	United Kingdom	2011-03-3
3	548311	21870	I CAN ONLY PLEASE ONE PERSON MUG	12	1.25	13728	United Kingdom	2011-03-3
4	548311	48173C	DOORMAT BLACK FLOCK	2	7.95	13728	United Kingdom	2011-03-3
5	548311	48187	DOORMAT NEW ENGLAND	2	7.95	13728	United Kingdom	2011-03-3
6	548311	20685	DOORMAT RED RETROSPOT	2	7.95	13728	United Kingdom	2011-03-3
7	548311	22691	DOORMAT WELCOME SUNRISE	2	7.95	13728	United Kingdom	2011-03-3
8	548311	48194	DOORMAT HEARTS	2	7.95	13728	United Kingdom	2011-03-3
9	548311	48188	DOORMAT WELCOME PUPPIES	2	7.95	13728	United Kingdom	2011-03-3
10	548311	48111	DOORMAT 3 SMILEY CATS	2	7.95	13728	United Kingdom	2011-03-3
11	548312	84836	ZINC METAL HEART DECORATION	6	1.25	16059	United Kingdom	2011-03-3
12	548312	22917	HERB MARKER ROSEMARY	2	0.65	16059	United Kingdom	2011-03-3
13	548312	22916	HERB MARKER THYME	2	0.65	16059	United Kingdom	2011-03-3
14	548312	22921	HERB MARKER CHIVES	2	0.65	16059	United Kingdom	2011-03-3

↓ 10,000 rows | Truncated data | 0.60s runtime

Refreshed 23 minutes ago

Microsoft Azure **cleaning** +

File Edit View Run Help Python Tabs: ON Last edit was 11 minutes ago

Create the Silver Delta Table

```

12
spark.conf.set(
    "fs.azure.account.key.etlproject02storage.dfs.core.windows.net",
    "HUhWrqEOTIMF0y/u2wKtm0IrwB3G0B0A8KhB5uBjKhdClzCv6eUMrW9luGrwLnQfV+8vtKUT+1AstjixXMu=")

13
# Create the Silver Delta Table
source_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.sales_cleaned")

(1) Spark Jobs

14
source_df.write.mode("overwrite").csv(f"abfss://source@etlproject02storage.dfs.core.windows.net/sales_cleaned")

(1) Spark Jobs

15
source_df.printSchema()

root
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: integer (nullable = true)
|-- UnitPrice: double (nullable = true)
|-- CustomerID: integer (nullable = true)
|-- Country: string (nullable = true)
|-- InvoiceTimestamp: timestamp (nullable = true)
|-- Year: integer (nullable = true)
|-- Month: integer (nullable = true)

```

Data_Transformation/transformation.ipynb

Microsoft Azure **transformation** +

File Edit View Run Help Python Tabs: ON

Customer Revenue

```

from pyspark.sql.functions import col, sum, countDistinct, when

dst_df = spark.read.table("online_retail.sales_cleaned")

customer_summary_df = dst_df.filter(col("CustomerID") != 0) \
    .groupby("CustomerID") \
    .agg(
        sum("TotalAmount").alias("TotalRevenue"),
        countDistinct("InvoiceNo").alias("NumberOfOrders"),
        sum("Quantity").alias("TotalItemsBought")
    ) \
    .withColumn("CustomerTier",
               when(col("TotalRevenue") > 10000, "Platinum") \
               .when(col("TotalRevenue") > 5000, "Gold") \
               .when(col("TotalRevenue") > 1000, "Silver") \
               .otherwise("Bronze"))
    ) \
    .orderBy(col("TotalRevenue").desc())

customer_summary_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.customer_summary")

customer_summary_df.write.mode("overwrite").parquet(f"abfss://destination@etlproject02storage.dfs.core.windows.net/customer_revenue")

display(customer_summary_df)

(13) Spark Jobs
customer_summary_df: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, TotalRevenue: double ... 3 more fields]
dst_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 12 more fields]

Table
CustomerID TotalRevenue NumberOfOrders TotalItemsBought CustomerTier

```

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON +

```
%sql
select * from online_retail.customer_summary where CustomerTier = 'Silver'
```

`_sqldf: pyspark.sql.DataFrame` = [CustomerID: integer, TotalRevenue: double ... 3 more fields]

Table +

	CustomerID	TotalRevenue	NumberOfOrders	TotalItemsBought	CustomerTier
1	15032	4959.0999999999985	3	2968	Silver
2	15547	4954.839999999999	14	4032	Silver
3	12437	4951.41	18	2922	Silver
4	12700	4939.889999999998	4	2967	Silver
5	15786	4932.200000000035	3	3364	Silver
6	14709	4921.04000000001	13	2484	Silver
7	17133	4892.24000000002	7	2730	Silver
8	12664	4881.88	9	1203	Silver
9	12688	4873.81000000005	1	3028	Silver
10	14292	4871.929999999999	9	1663	Silver
11	17652	4867.72000000001	12	2509	Silver
12	15738	4812.42	16	2982	Silver
13	13013	4810.94000000005	13	2224	Silver
14	15555	4805.17	16	4439	Silver
15	17365	4801.56	8	4777	Silver

1,393 rows | 1.16s runtime Refreshed 37 minutes ago

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

Monthly Sales Trends

```
monthly_trends_df = dst_df.groupBy("Year", "Month") \
    .agg(sum("TotalAmount").alias("MonthlyRevenue"),
        sum("Quantity").alias("NumberOfItemsSold"),
        countDistinct("InvoiceNo").alias("TransactionCount"),
        countDistinct("CustomerID").alias("UniqueCustomers")) \
    .orderBy("Year", "Month")
```

`monthly_trends_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.monthly_trends")`
`monthly_trends_df.write.mode("overwrite").parquet(f"abfss://destination@etlproject02storage.dfs.core.windows.net/monthly_trends")`
`monthly_trends_df.display()`

`monthly_trends_df: pyspark.sql.DataFrame` = [Year: integer, Month: integer ... 4 more fields]

Table +

	Year	Month	MonthlyRevenue	NumberOfItemsSold	TransactionCount	UniqueCustomers
1	2010	12	823746.1399999646	359239	1559	886
2	2011	1	691364.5600000108	387785	1086	742
3	2011	2	523631.8900000278	283555	1100	759
4	2011	3	717639.3600000228	377526	1454	975
5	2011	4	537808.6200000242	308815	1246	857
6	2011	5	770536.0200000107	395738	1681	1057
7	2011	6	761739.9000000219	389213	1533	992
8	2011	7	719221.1900000235	401759	1475	950
9	2011	8	759138.3800000148	421770	1361	936
10	2011	9	1058590.1699999967	570820	1837	1267
11	2011	10	1154979.3000000077	623401	2040	1365
12	2011	11	1509496.329999743	754607	2769	1666

Top Selling Products

```

✓ 08:35 PM (6s)
top_products_df = dst_df.groupBy("StockCode", "Description") \
    .agg(sum("TotalAmount").alias("TotalRevenue"),
         sum("Quantity").alias("TotalNoOfItems"),
         countDistinct("InvoiceNo").alias("TotalNoOfTimesOrdered")) \
    .orderBy(col("TotalRevenue").desc())
top_products_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.top_products")
top_products_df.write.mode("overwrite").parquet("abfss://destination@etlproject02storage.dfs.core.windows.net/top_products")
top_products_df.display()

```

(13) Spark Jobs

top_products_df: pyspark.sql.dataframe.DataFrame = [StockCode: string, Description: string ... 3 more fields]

StockCode	Description	TotalRevenue	TotalNoOfItems	TotalNoOfTimesOrdered
DOT	DOTCOM POSTAGE	206248.7700000002	706	706
22423	REGENCY CAKESTAND 3 TIER	174484.7399999982	13879	1988
32843	PAPER CRAFT, LITTLE BIRDIE	168469.6	80995	1
85123A	WHITE HANGING HEART T-LIGHT HOLDER	104340.2900000033	37599	2189
47566	PARTY BUNTING	99504.3299999993	18295	1685
850998	JUMBO BAG RED RETROSPOT	94340.049999997	48474	2089
23166	MEDIUM CERAMIC TOP STORAGE JAR	81700.9200000003	78033	247
M	Manual	78110.2700000002	7224	289
POST	POSTAGE	78101.88	3150	1126
23084	RABBIT NIGHT LIGHT	66964.9899999987	30788	994
22086	PAPER CHAIN KIT 50'S CHRISTMAS	64952.29000000015	19355	1160

Sales by Country

```

✓ 08:35 PM (5s)
from pyspark.sql.functions import col, sum, countDistinct

sales_by_country_df = (
    dst_df.groupBy("Country")
    .agg(
        sum("TotalAmount").alias("TotalRevenue"),
        countDistinct("InvoiceNo").alias("NumberofOrders"),
        countDistinct("CustomerID").alias("UniqueCustomers")
    )
    .orderBy(col("TotalRevenue").desc())
)

# Save to Delta
sales_by_country_df.write.mode("overwrite").format("delta").saveAsTable("online_retail.sales_by_country")

# Save as Parquet in ADLS
sales_by_country_df.write.mode("overwrite").parquet(
    "abfss://destination@etlproject02storage.dfs.core.windows.net/sales_by_country"
)

display(sales_by_country_df)

```

(13) Spark Jobs

sales_by_country_df: pyspark.sql.dataframe.DataFrame = [Country: string, TotalRevenue: double ... 2 more fields]

Country	TotalRevenue	NumberofOrders	UniqueCustomers
United Kingdom	902522.07999794	18019	3921
Netherlands	285446.3399999997	94	9
EIRE	283453.9599999973	288	4
Germany	228867.140000004	457	94

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON

transformation

Run all etl-processing-cluster Schedule Share

Printing schema for the table we have created

```

08:35 PM (c1a)
customer_summary_df.printSchema()
monthly_trends_df.printSchema()
sales_by_country_df.printSchema()
top_products_df.printSchema()

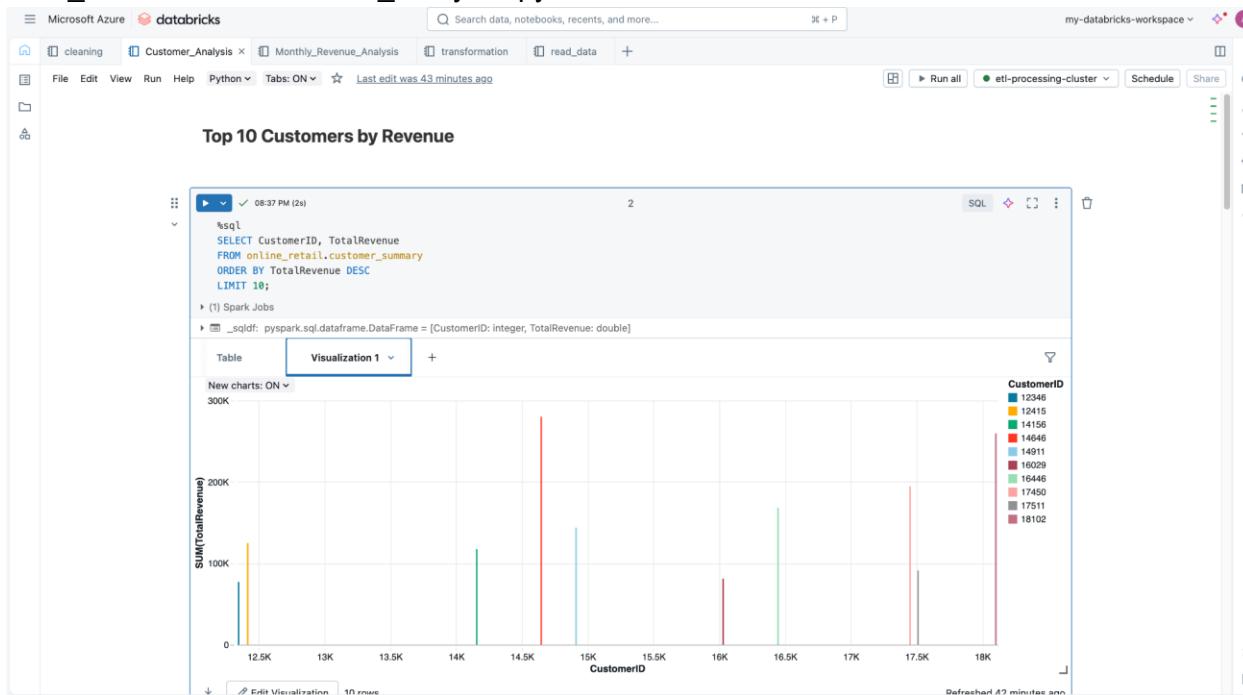
root
|-- Year: integer (nullable = true)
|-- Month: integer (nullable = true)
|-- MonthlyRevenue: double (nullable = true)
|-- NumberOfItemsSold: long (nullable = true)
|-- TransactionCount: long (nullable = false)
|-- UniqueCustomers: long (nullable = false)

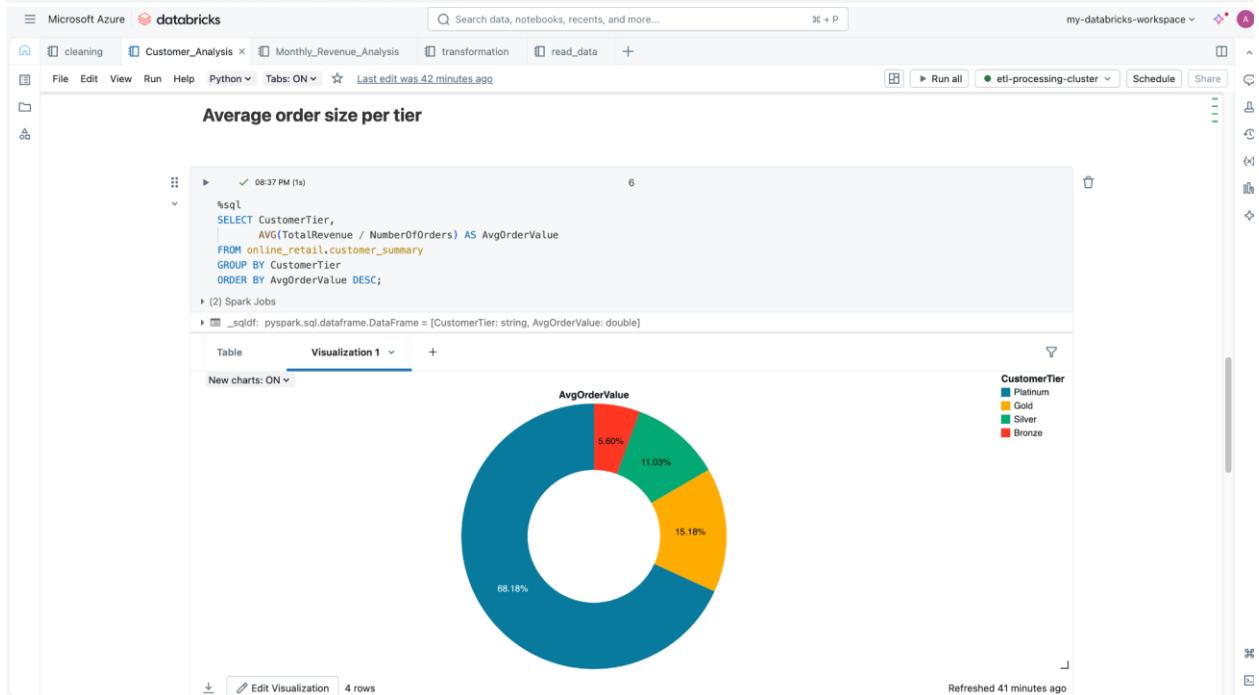
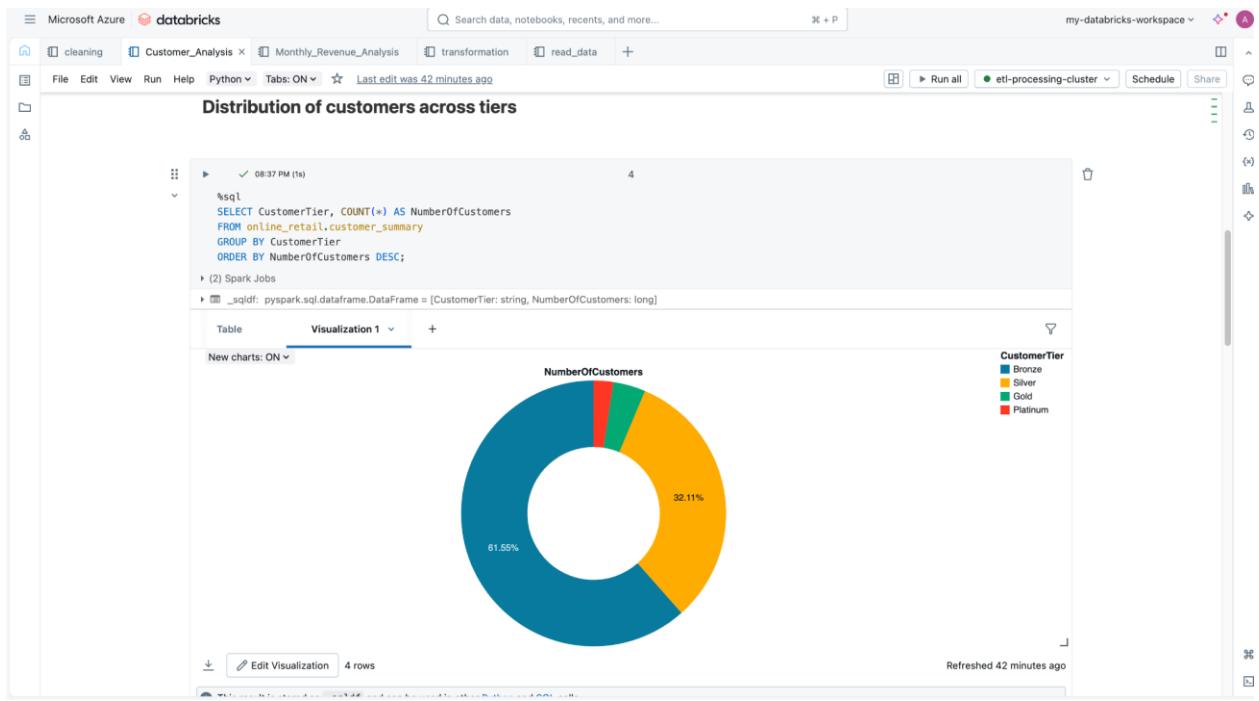
root
|-- Country: string (nullable = true)
|-- TotalRevenue: double (nullable = true)
|-- NumberOfOrders: long (nullable = false)
|-- UniqueCustomers: long (nullable = false)

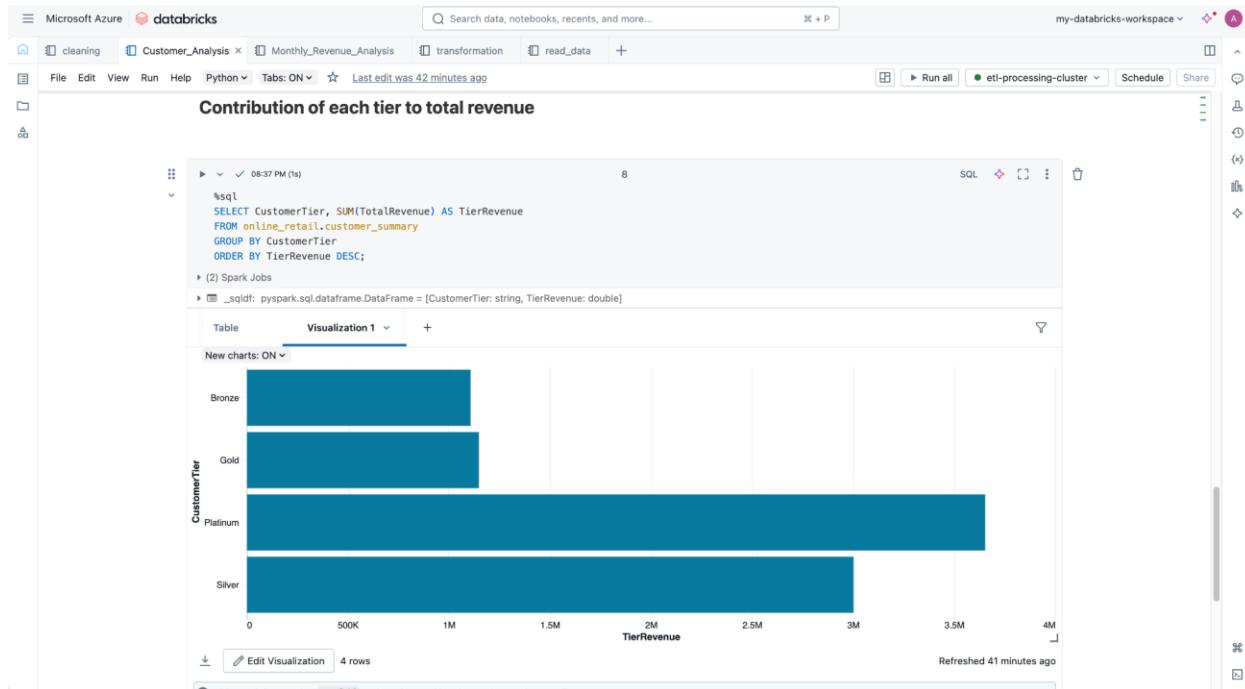
root
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- TotalRevenue: double (nullable = true)
|-- TotalNumberOfItems: long (nullable = true)
|-- TotalNumberOfTimesOrdered: long (nullable = false)

```

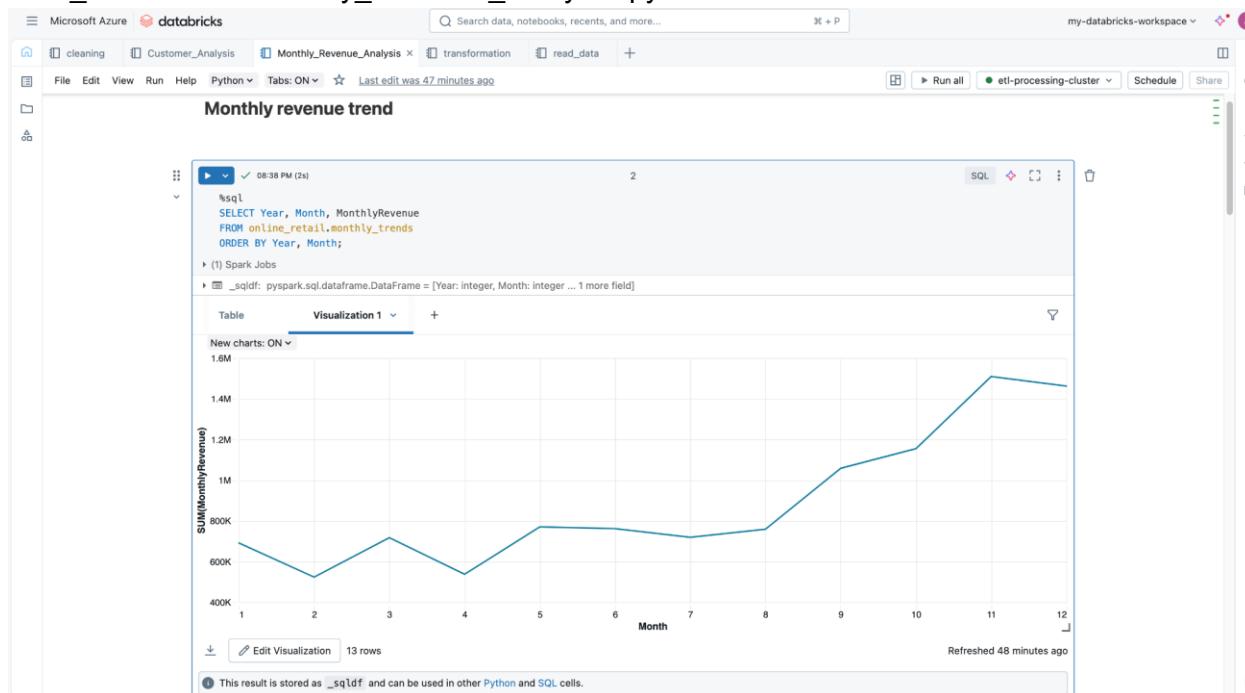
Data_Visualizaton/Customer_Analysis.ipynb

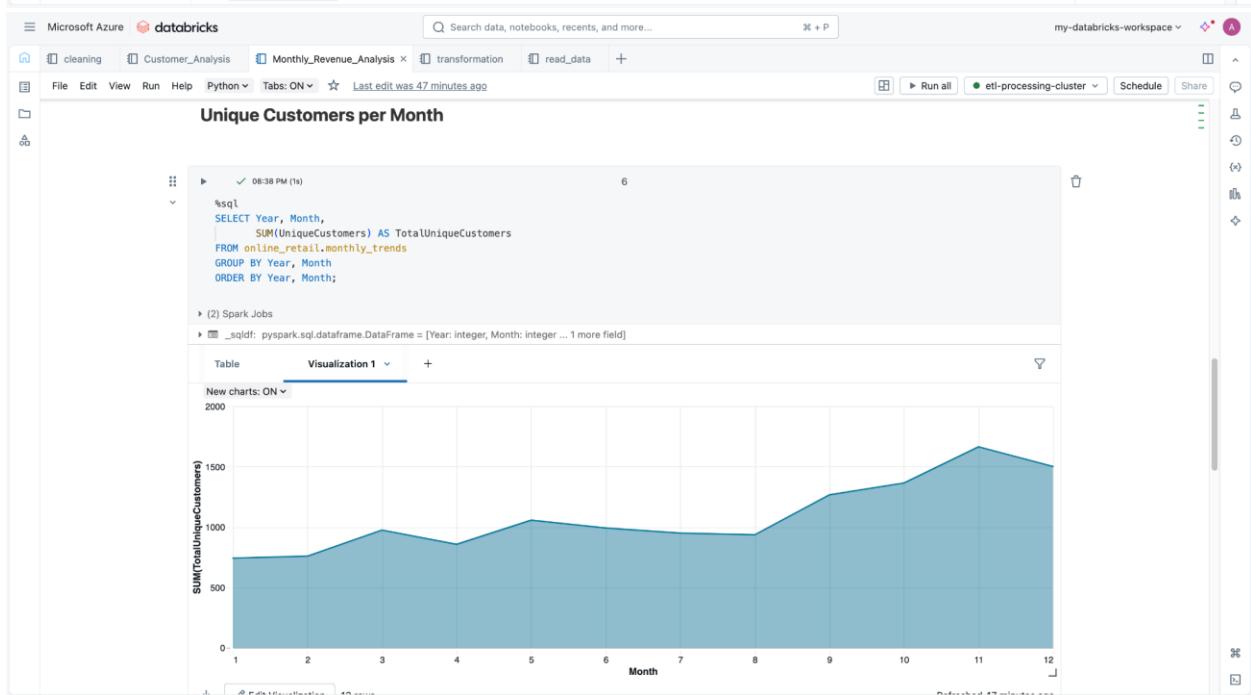
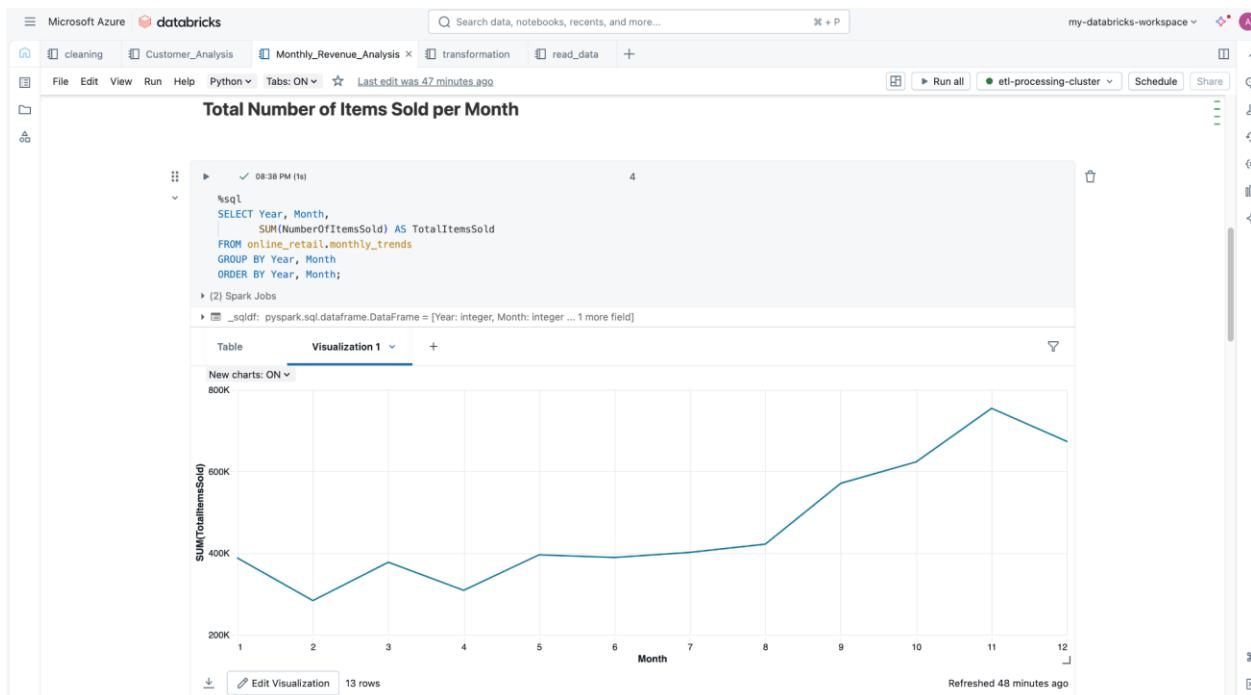






Data_Visualizaton/Monthly_Revenue_Analysis.ipynb





Data_Visualizaton/Product_Performance_Analysis.ipynb

Top 10 products by revenue

```
%sql
SELECT StockCode, Description, TotalRevenue
FROM online_retail.top_products
ORDER BY TotalRevenue DESC
LIMIT 10;
```

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

Product Performance Analysis

Which products were ordered most frequently?

```
%sql
SELECT StockCode, Description, TotalNoOfTimesOrdered
FROM online_retail.top_products
ORDER BY TotalNoOfTimesOrdered DESC
LIMIT 10;
```

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

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON +

Search data, notebooks, recents, and more...

my-databricks-workspace Schedule Share

Which products sold the highest number of units?

```
%sql
SELECT StockCode, Description, TotalNoOfItems
FROM online_retail.top_products
ORDER BY TotalNoOfItems DESC
LIMIT 10;
```

(1) Spark Jobs

_sqldf: pyspark.sql.DataFrame = [StockCode: string, Description: string ... 1 more field]

Table **Visualization 1** +

New charts: ON 100K

StockCode	SUM(TotalNoOfItems)
21212	~35K
22197	~35K
22492	~25K
23084	~30K
23166	~75K
23843	~80K
84077	~55K
84879	~35K
85099B	~45K
85123A	~38K

Refreshed 52 minutes ago

This result is stored as etl1ff and can be used in other Databricks and SQL calls.

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON +

Search data, notebooks, recents, and more...

my-databricks-workspace Schedule Share

What is the average revenue per order for each product?

```
%sql
SELECT StockCode, Description,
       (TotalRevenue / TotalNoOfTimesOrdered) AS AvgRevenuePerOrder
FROM online_retail.top_products
ORDER BY AvgRevenuePerOrder DESC
LIMIT 10;
```

(1) Spark Jobs

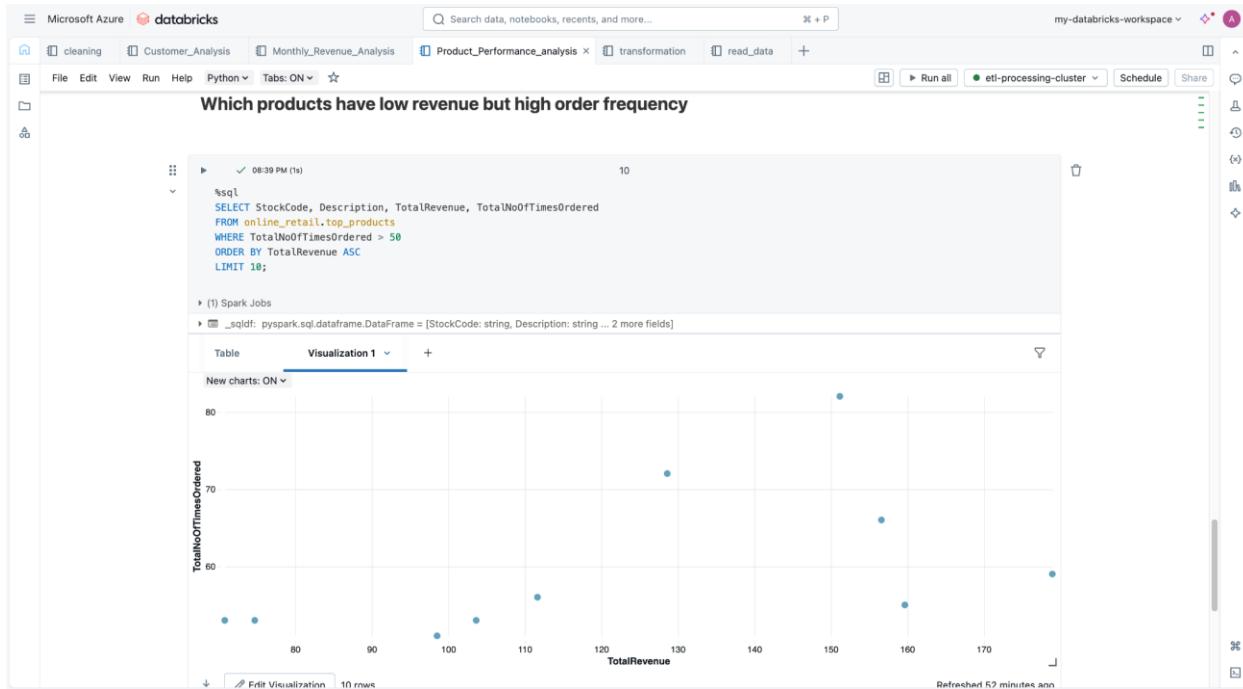
_sqldf: pyspark.sql.DataFrame = [StockCode: string, Description: string ... 1 more field]

Table **Visualization 1** +

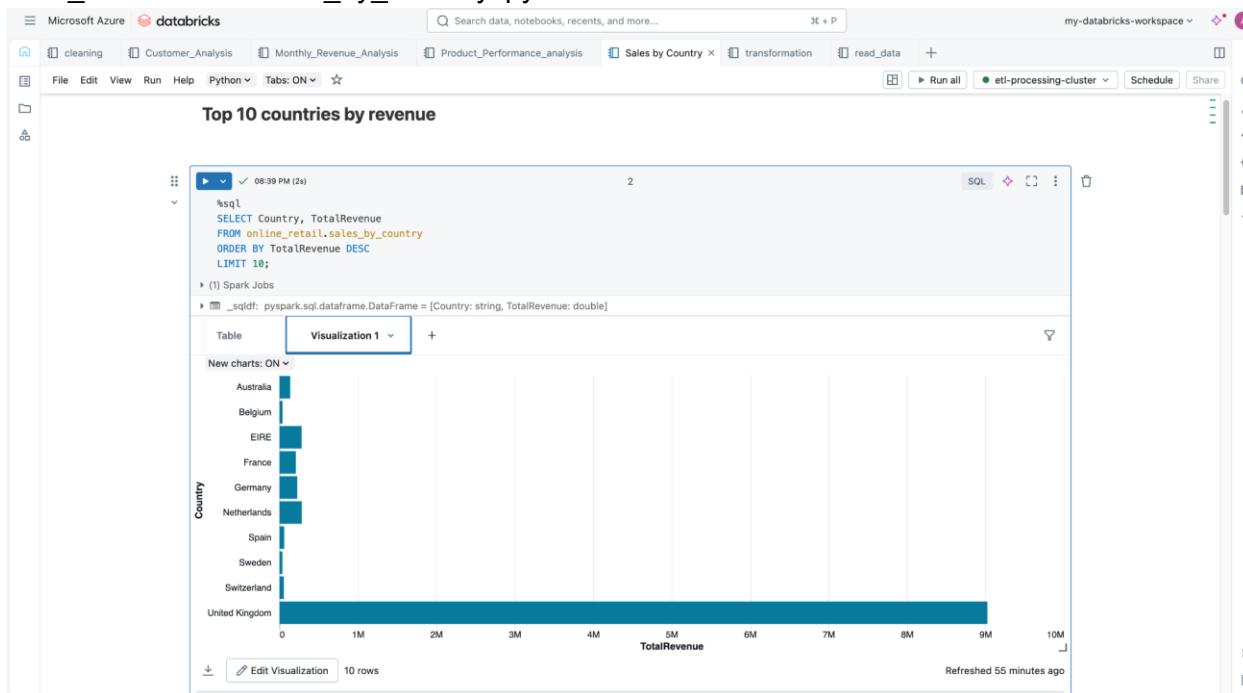
New charts: ON 200K

StockCode	AvgRevenuePerOrder
2014	~1K
22275	~1K
22222	~10K
22030	~1K
22833	~1K
23131	~1K
28843	~160K
899647	~1K
AMAZONFEE	~10K
B	~10K

Refreshed 52 minutes ago



Data_Visualizaton/Sales_by_Country.ipynb



Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON

Search data, notebooks, recents, and more...

my-databricks-workspace

Which countries have the highest revenue per customer

```
%sql
SELECT Country,
       (TotalRevenue / UniqueCustomers) AS RevenuePerCustomer
FROM online_retail.sales_by_country
ORDER BY RevenuePerCustomer DESC
LIMIT 10;
```

(1) Spark Jobs

_sqldf: pyspark.sql.DataFrame = [Country: string, RevenuePerCustomer: double]

Table Visualization 1 +

New charts: ON

Country	RevenuePerCustomer
Australia	~15K
EIRE	~70K
Hong Kong	~15K
Iceland	~5K
Japan	~5K
Netherlands	~32K
Norway	~3K
Singapore	~22K
Sweden	~5K
Switzerland	~2K

Refreshed 55 minutes ago

Microsoft Azure databricks

File Edit View Run Help Python Tabs: ON

Search data, notebooks, recents, and more...

my-databricks-workspace

What is the average order size per country

```
%sql
SELECT Country,
       (TotalRevenue / NumberOfOrders) AS AvgOrderValue
FROM online_retail.sales_by_country
ORDER BY AvgOrderValue DESC;
```

(1) Spark Jobs

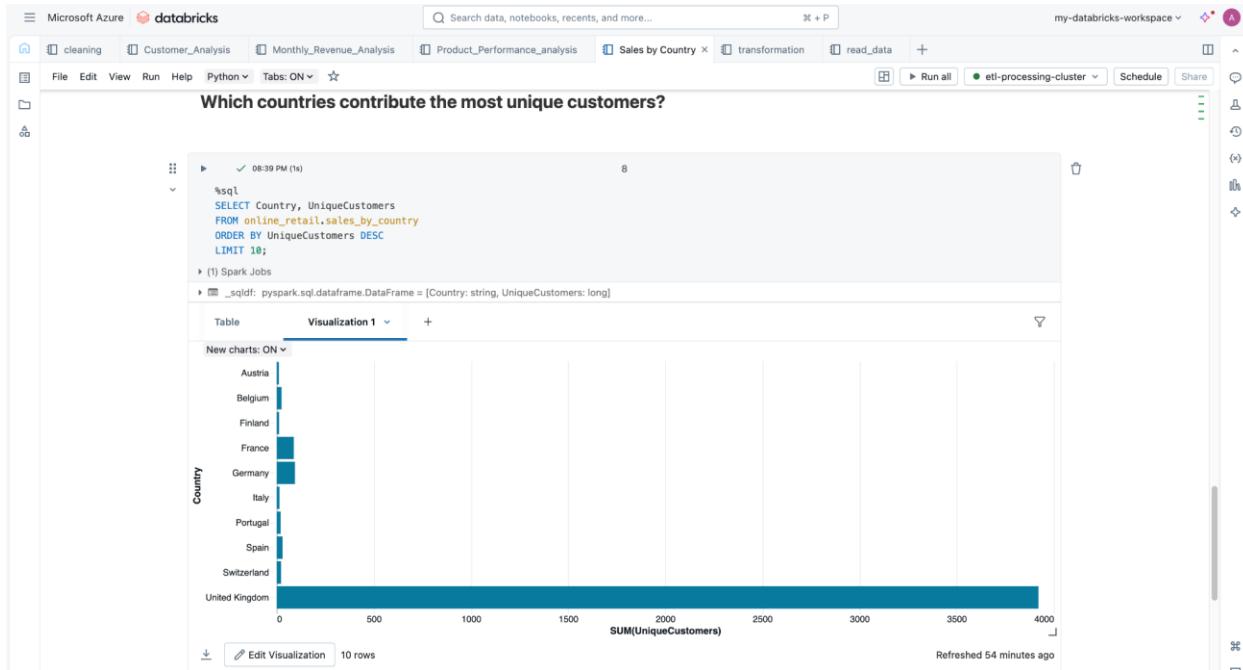
_sqldf: pyspark.sql.DataFrame = [Country: string, AvgOrderValue: double]

Table Visualization 1 +

New charts: ON

Country	AvgOrderValue
Australia	~2200
Austria	~800
Bahrain	~200
Belgium	~150
Brazil	~1000
Canada	~800
Croatia	~800
Czech Republic	~400
Denmark	~1000
Egypt	~200
France	~500
Germany	~500
Greece	~1000
Hong Kong	~1500
Ireland	~1000
Italy	~500
Japan	~2000
Lithuania	~1500
Luxembourg	~1500
Malta	~500
Netherlands	~3000
Norway	~1000
Poland	~300
Portugal	~500
RSA	~1000
Saudi Arabia	~100
Spain	~800
Sweden	~1000
Switzerland	~1000
United Arab Emirates	~500
United Kingdom	~800
Unspecified	~500
USA	~1000

Refreshed 54 minutes ago



Here is the [Notebooks link](#) used for the transformations performed for the dataset used.

Step 5: Schedule a job:

1. Create a New Pipeline:

- In Databricks notebook, click on the “Schedule” button on top panel
- Select “Advanced”, set the time of the day when you want to schedule.

The scheduled job configuration is as follows:

- Job name***: Customer_Analysis
- Simple** (selected) vs **Advanced**
- Schedule**: Every Day at 08 : 30
- Show cron syntax**: Available but not selected
- Timezone**: (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
- Cluster***: etl-processing-cluster (14 GB - 4 Cores - DBR 16.4 LTS - Photon - Spark 3.5.2 - Scala 2...)
- Notes**: Jobs running on all-purpose clusters are considered all-purpose compute. Learn more

2. Monitoring the Scheduled Jobs

- In Databricks, click on Job runs in the left panel.
- Here you can monitor and manage the scheduled job runs.

The screenshot shows the Databricks interface under the 'Jobs & Pipelines' section. On the left sidebar, 'Job Runs' is selected. The main area is titled 'Jobs & Pipelines' and shows three categories: 'Ingestion pipeline', 'ETL pipeline', and 'Job'. Below this, the 'Job runs' tab is active, showing a timeline from 27 Aug, 12 AM to 28 Aug, 12 PM. A single run for 'ETL_Pipeline' is listed, starting at 06:00 AM on Aug 28, 2025, and completed successfully by the scheduler. The status bar indicates a duration of 5m 8s.

Successful output generated

The screenshot shows the Azure Storage Blob Explorer interface. The current container is 'top_products'. It displays a list of blobs with their names, last modified times, access tiers, blob types, sizes, and lease states. The blobs include '_SUCCESS', '_committed' files for different transaction steps, and log files like '_started' and '_vacuum'.

Name	Last modified	Access tier	Blob type	Size	Lease state
[...]	8/28/2025, 8:35:49 PM	Hot (Inferred)	Block blob	0	Available
_SUCCESS	8/28/2025, 8:35:49 PM	Hot (Inferred)	Block blob	123 B	Available
_committed_302660837931308330	8/26/2025, 3:28:09 PM	Hot (Inferred)	Block blob	233 B	Available
_committed_4985984993560014411	8/28/2025, 8:35:48 PM	Hot (Inferred)	Block blob	94 B	Available
_committed_vacuum786145933395621226	8/28/2025, 8:35:49 PM	Hot (Inferred)	Block blob	0	Available
_started_4985984993560014411	8/28/2025, 8:35:48 PM	Hot (Inferred)	Block blob	124.17 KiB	Available
part-00000-tid-4985984993560014411-114da079-c124-40ab-...	8/28/2025, 8:35:48 PM	Hot (Inferred)	Block blob		

destination > sales_by_country

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

<input type="checkbox"/>	Name	Last modified	Access tier	Blob type	Size	Lease state	...
<input type="checkbox"/>	[...]						...
<input type="checkbox"/>	_committed_4664427409557547105	8/26/2025, 3:19:12 PM	Hot (Inferred)	Block blob	124 B	Available	...
<input type="checkbox"/>	_committed_4977158028422595508	8/26/2025, 3:22:56 PM	Hot (Inferred)	Block blob	234 B	Available	...
<input type="checkbox"/>	_committed_5229165469448032315	8/28/2025, 8:35:54 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_vacuum2812209875930735912	8/28/2025, 8:35:54 PM	Hot (Inferred)	Block blob	129 B	Available	...
<input type="checkbox"/>	_started_5229165469448032315	8/28/2025, 8:35:54 PM	Hot (Inferred)	Block blob	0	Available	...
<input type="checkbox"/>	part-00000-tid-5229165469448032315-79f260a7-3b7f-4c2d-a...	8/28/2025, 8:35:54 PM	Hot (Inferred)	Block blob	2.3 KiB	Available	...

destination > monthly_trends

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

<input type="checkbox"/>	Name	Last modified	Access tier	Blob type	Size	Lease state	...
<input type="checkbox"/>	[...]						...
<input type="checkbox"/>	_committed_1214210885200035753	8/26/2025, 2:55:54 PM	Hot (Inferred)	Block blob	234 B	Available	...
<input type="checkbox"/>	_committed_6837125831028881171	8/26/2025, 2:55:32 PM	Hot (Inferred)	Block blob	124 B	Available	...
<input type="checkbox"/>	_committed_7685052019472324131	8/26/2025, 2:59:52 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_8507473497068785604	8/28/2025, 8:35:43 PM	Hot (Inferred)	Block blob	222 B	Available	...
<input type="checkbox"/>	_committed_8555231052868279132	8/26/2025, 3:13:51 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_vacuum2948108828637494095	8/28/2025, 8:35:43 PM	Hot (Inferred)	Block blob	195 B	Available	...
<input type="checkbox"/>	_started_8507473497068785604	8/28/2025, 8:35:43 PM	Hot (Inferred)	Block blob	0	Available	...
<input type="checkbox"/>	part-00000-tid-8507473497068785604-02deb404-a5cb-400f...	8/28/2025, 8:35:43 PM	Hot (Inferred)	Block blob	2.04 KiB	Available	...

destination > customer_revenue

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

<input type="checkbox"/>	Name	Last modified	Access tier	Blob type	Size	Lease state	...
<input type="checkbox"/>	[...]						...
<input type="checkbox"/>	_committed_1911470872272266835	8/28/2025, 8:35:37 PM	Hot (Inferred)	Block blob	222 B	Available	...
<input type="checkbox"/>	_committed_2155556316547594000	8/26/2025, 2:55:05 PM	Hot (Inferred)	Block blob	124 B	Available	...
<input type="checkbox"/>	_committed_4155717268048946532	8/26/2025, 3:14:05 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_5305855028892238027	8/26/2025, 3:13:05 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_6002914860824731174	8/26/2025, 3:08:33 PM	Hot (Inferred)	Block blob	223 B	Available	...
<input type="checkbox"/>	_committed_6608820219819215315	8/26/2025, 3:07:39 PM	Hot (Inferred)	Block blob	234 B	Available	...
<input type="checkbox"/>	_committed_vacuum3149720735347050524	8/28/2025, 8:35:37 PM	Hot (Inferred)	Block blob	228 B	Available	...
<input type="checkbox"/>	_started_1911470872272266835	8/28/2025, 8:35:37 PM	Hot (Inferred)	Block blob	0	Available	...
<input type="checkbox"/>	part-00000-tid-1911470872272266835-cc401b64-83eb-46b2...	8/28/2025, 8:35:37 PM	Hot (Inferred)	Block blob	56.01 KiB	Available	...

Here is the [Notebooks link](#) used for the transformations performed for the dataset used.

Conclusion

In conclusion, this project successfully showcases a powerful, scalable, and robust solution for large-scale data processing. By leveraging Azure Databricks clusters, we can distribute the computational load, and with PySpark, we can implement complex data transformations with ease.

The solution highlights:

- Scalable data processing with Databricks clusters.
- Efficient storage using Parquet/Delta formats.
- Reusable, automated ETL pipelines.
- Improved query speed and resource optimization.

This architecture serves as a foundational blueprint for building enterprise-grade ETL pipelines capable of handling the demands of big data.