Large Scale Data Processing with ETL Operations using PySpark

Table of Contents

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.

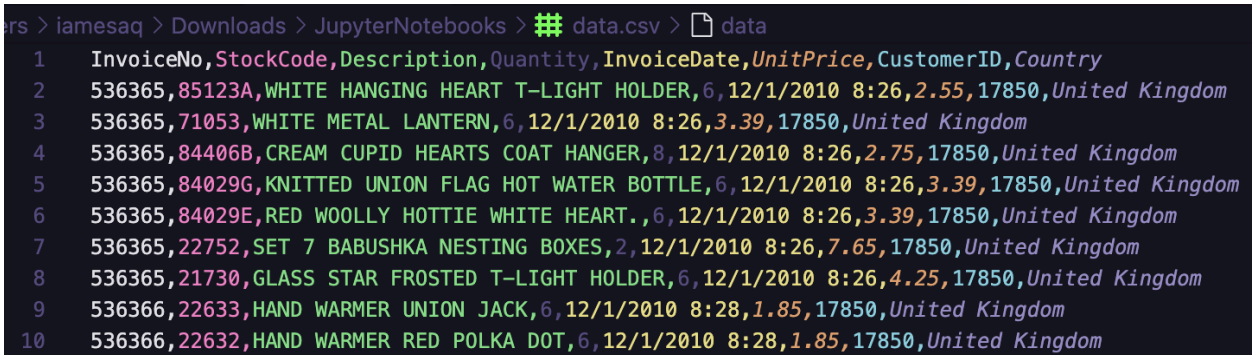
Tools/Technology Used in the Project

- **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.



Implementation -Tasks performed:

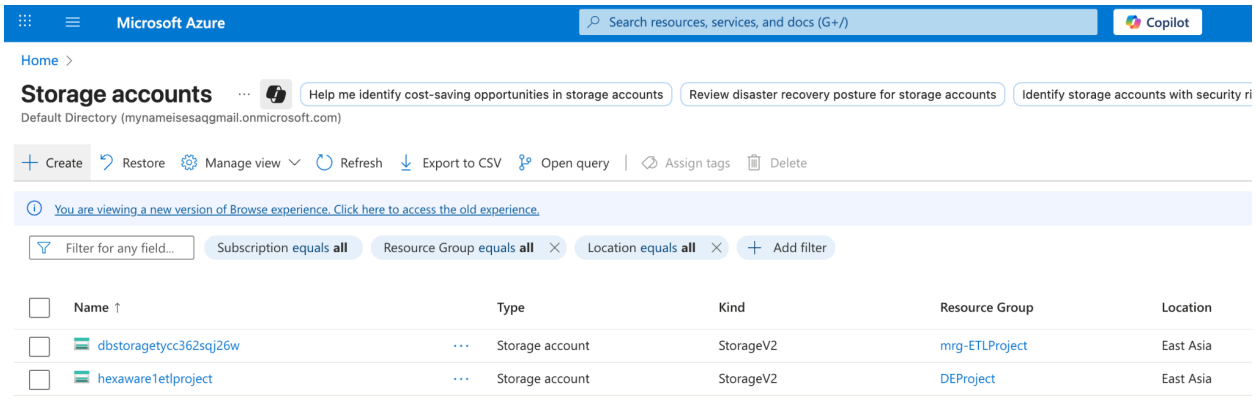
- Create a Azure Storage Account
- Upload the dataset to a container
- Set Up Azure Databricks Workspace.
- Develop PySpark Scripts for ETL Operations.
- Add ETL Operation Activity in the Pipeline.

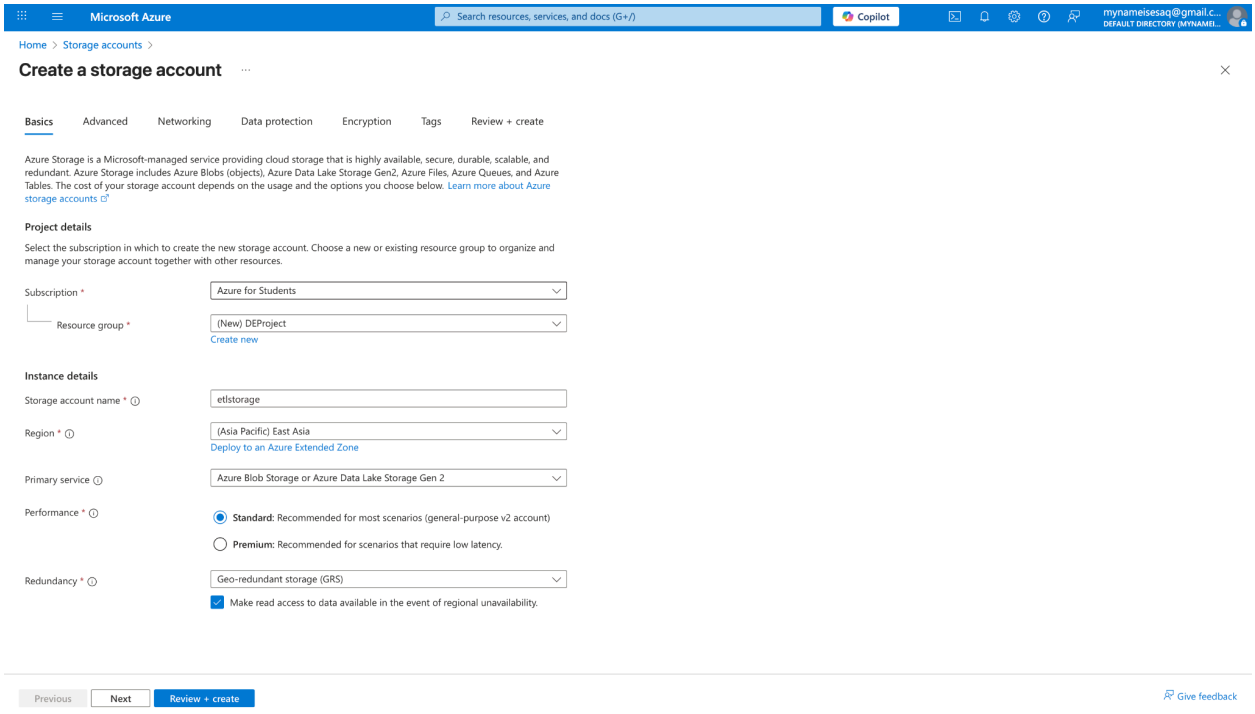
- Schedule a Job execution.
- Monitor Job Performance.

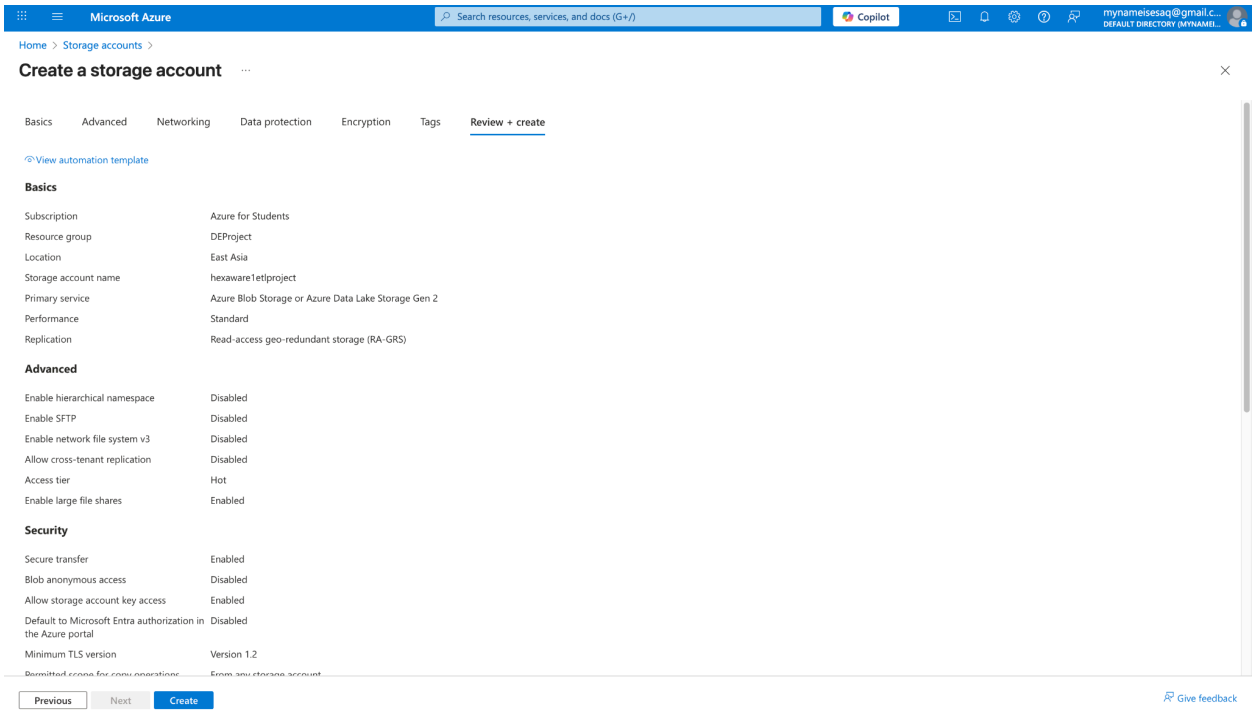
Steps on practical implementation on the Azure portal

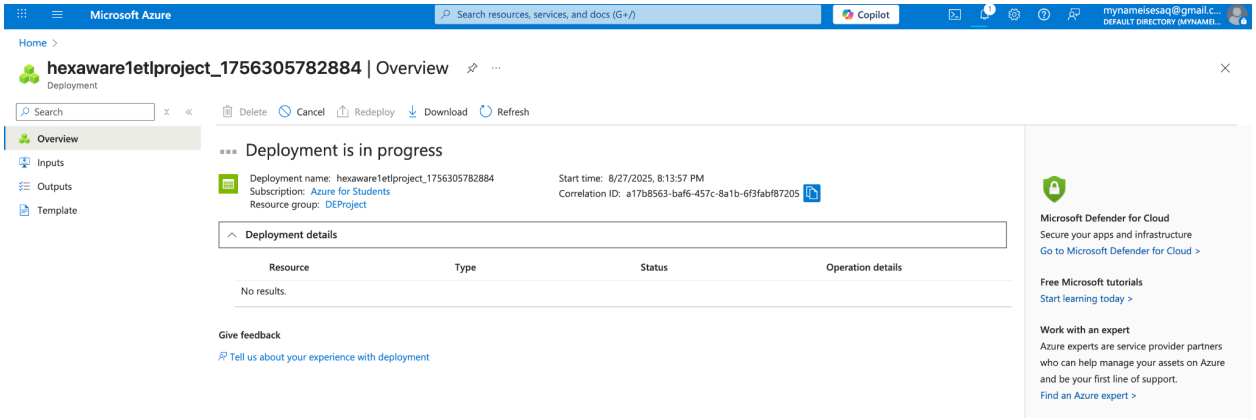
Step 1: Set Up Azure Storage Account

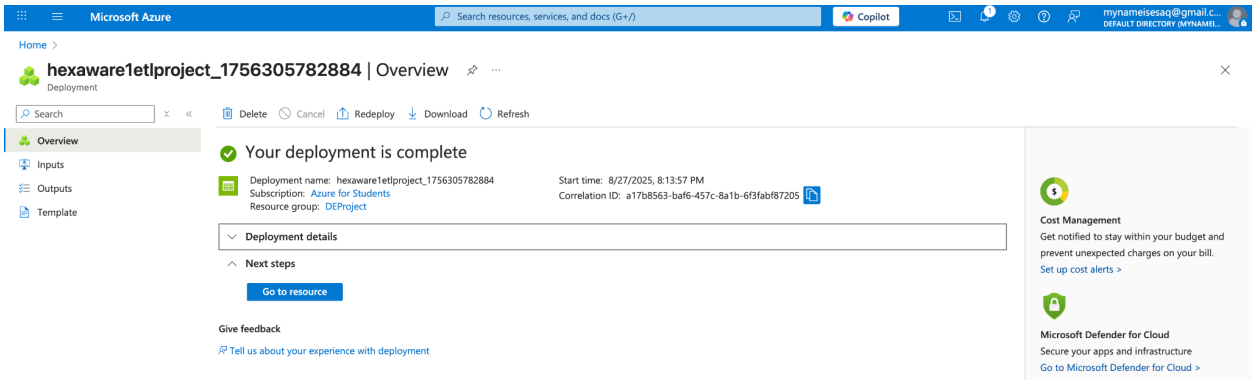
1. **Create a Data Factory Instance:**
    - Go to the Azure Portal and search for “Storage Account”.
    - Click on "+ Create"
    - Fill in the details like name, region, performance etc., and create the instance.



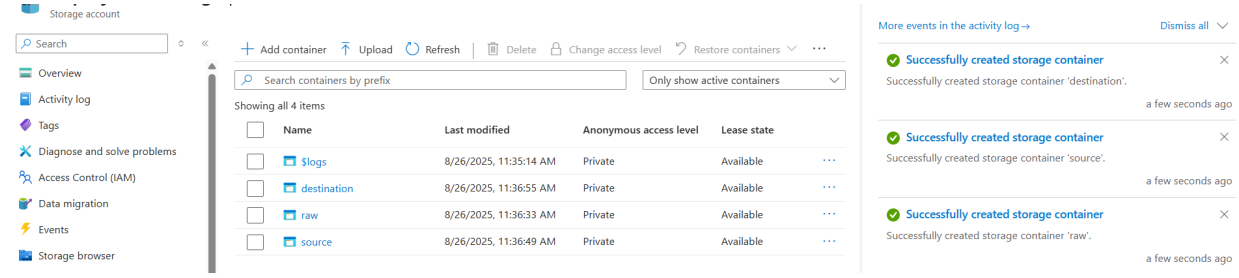

Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
mynameisesaq@gmail.c...
DEFAULT DIRECTORY (MYNAMEI...
Home > Storage accounts >
Create a storage account
Basics Advanced Networking Data protection Encryption Tags Review + create
Azure Storage is a Microsoft-managed service providing 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 *
Azure for Students
Resource group *
(New) DEProject
Create new
Instance details
Storage account name *
etlstorage
Region *
(Asia Pacific) East Asia
Deploy to an Azure Extended Zone
Primary service
Azure Blob Storage or Azure Data Lake Storage Gen 2
Performance *
Standard: Recommended for most scenarios (general-purpose v2 account)
Premium: Recommended for scenarios that require low latency.
Redundancy *
Geo-redundant storage (GRS)
Make read access to data available in the event of regional unavailability.
Previous
Next
Review + create
Give feedback


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
mynameisesaq@gmail.c...
DEFAULT DIRECTORY (MYNAMEI...
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



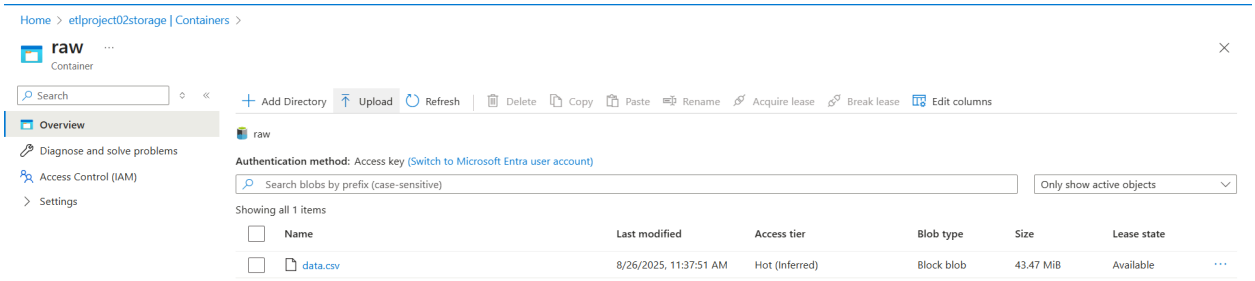


2. **Create Container in Azure Storage Account:**
    - After deployment, click on “Go to Resource”.
    - Then click on “Data Storage” > “Containers” > “Add Container”
    - Enter a container name and create 3 containers.

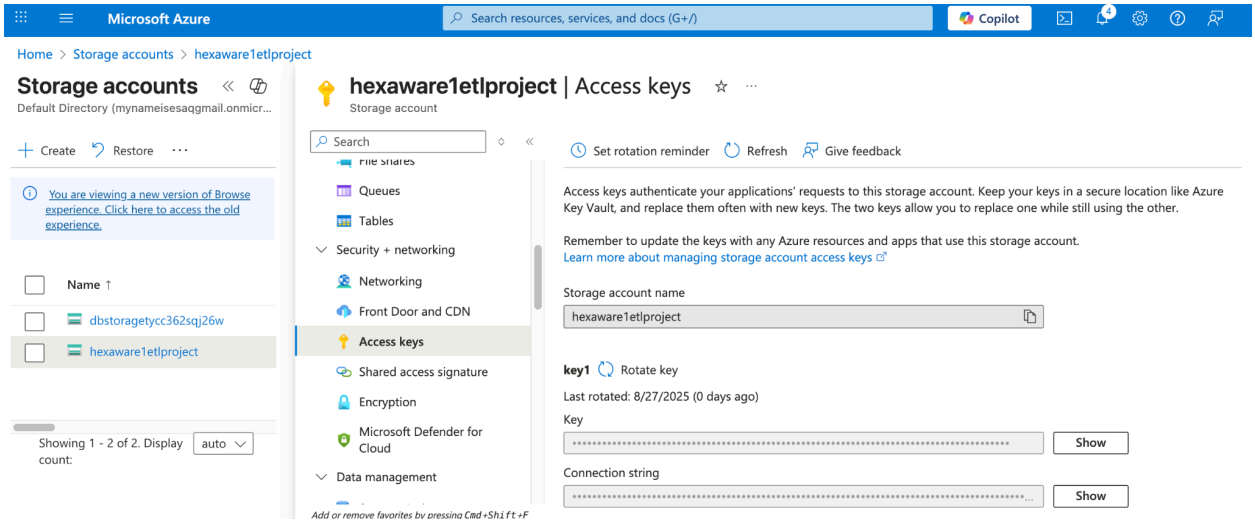


Step 2: Upload the Dataset to a Storage Account.

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

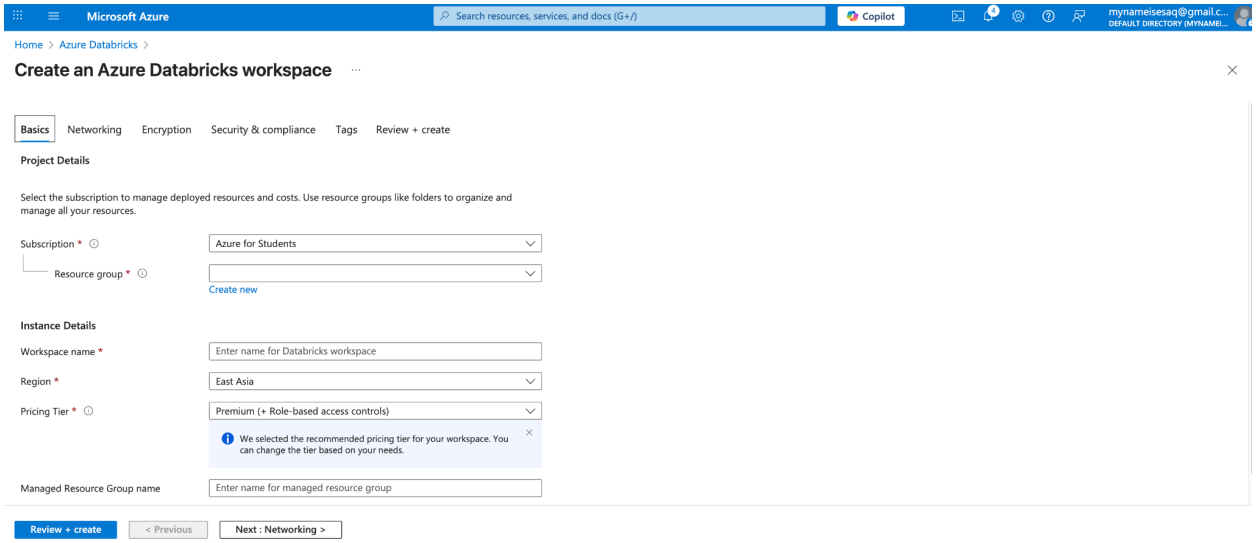


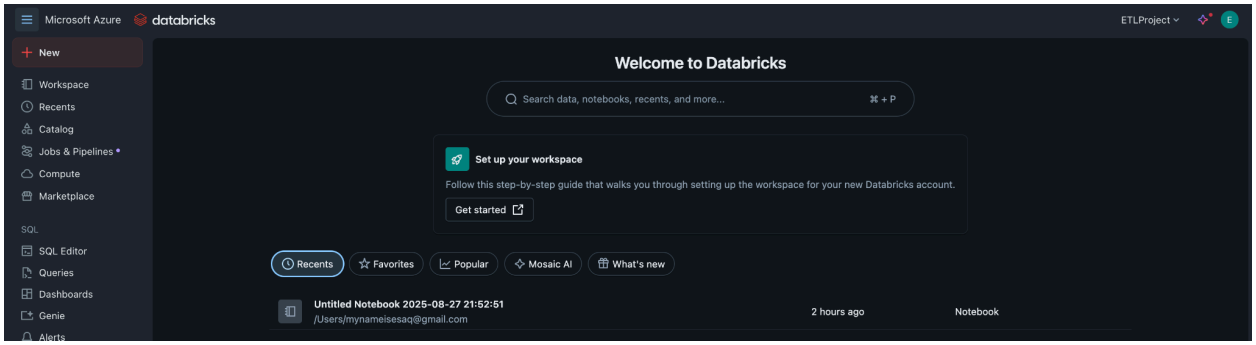
2. **To Access Dataset from ADLS:**
    - In Storage Account, click on "Security+Networking" > “Access Keys”
    - There you will find 2 access keys, copy any one of the “Key” for later.



Step 3: Set Up Azure Databricks

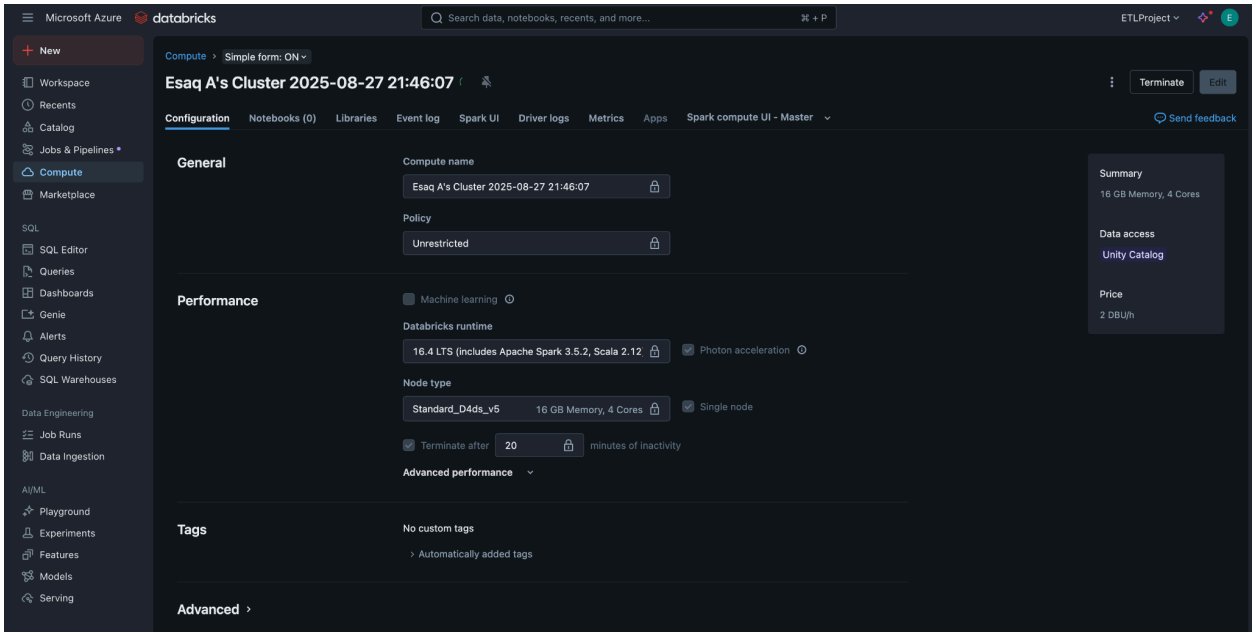
1. **Create a Databricks workspace:**
    - Search for Azure Databricks in Azure Portal
    - Click on “+ Create” and fill the name, region, etc details and create a new workspace
    - Launch the workspace after deployment.





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.

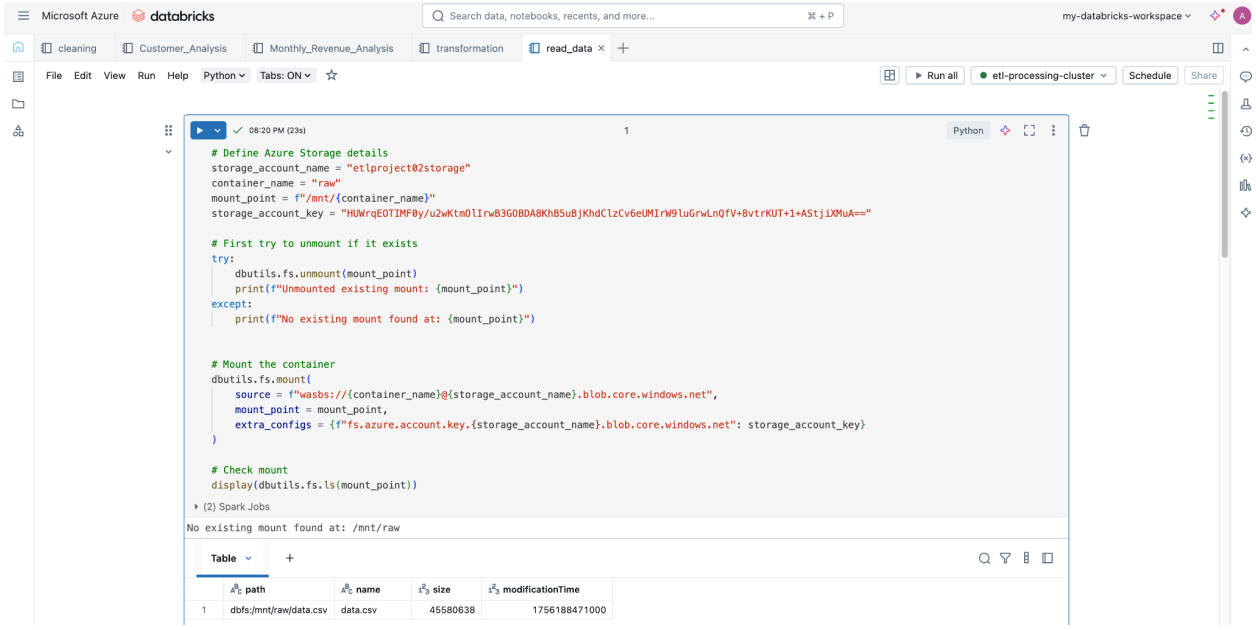


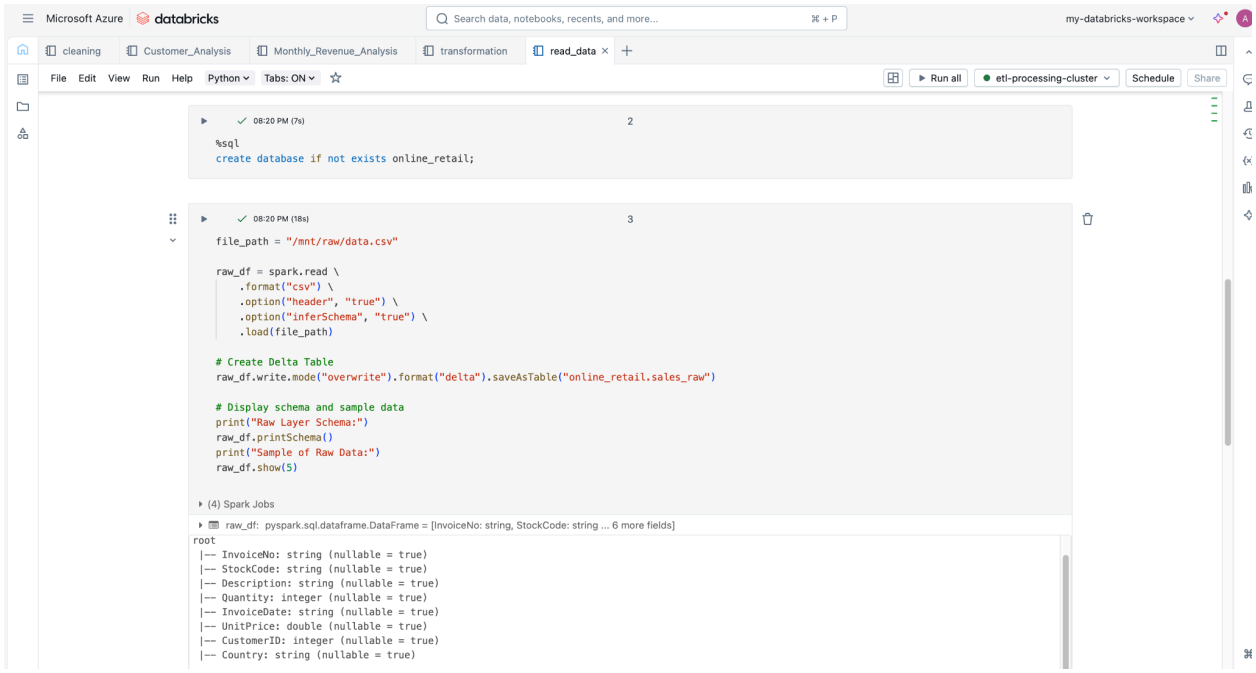
2. **Create a Notebook in Databricks:**
    - Go to your Databricks workspace and create a new notebook.
    - Choose PySpark as the language.

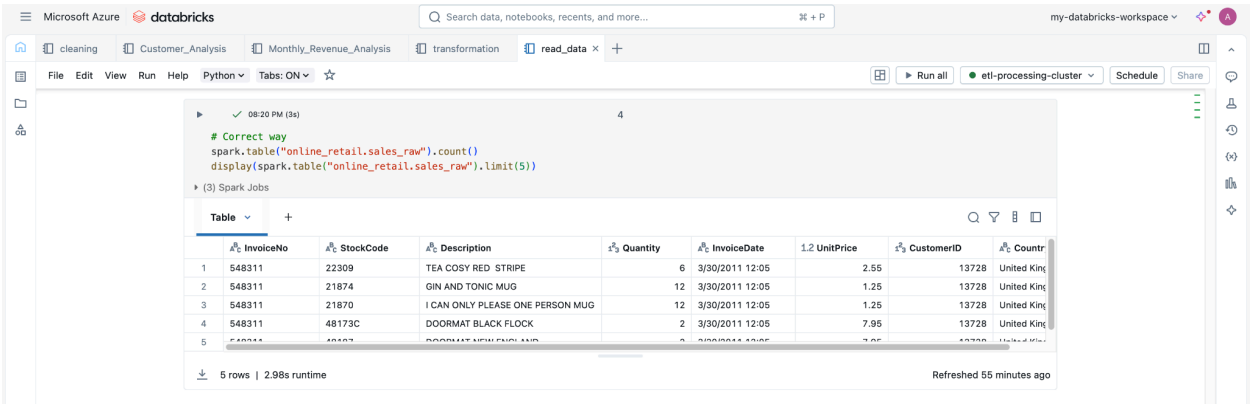
3. **Write PySpark Script:**
    - Implement your PySpark script. For example:

```
storage_account_name = "mydatabricksstorage12"
container_name = "raw-data"
secret_name =
"d8CceViUDRX70J9LvideVqWIQccNb35nug80sVLPxoTcTjbewDuDhpC0Hxt4aQI343pvb6hM
/KvY+AStQYK2zg==”
```

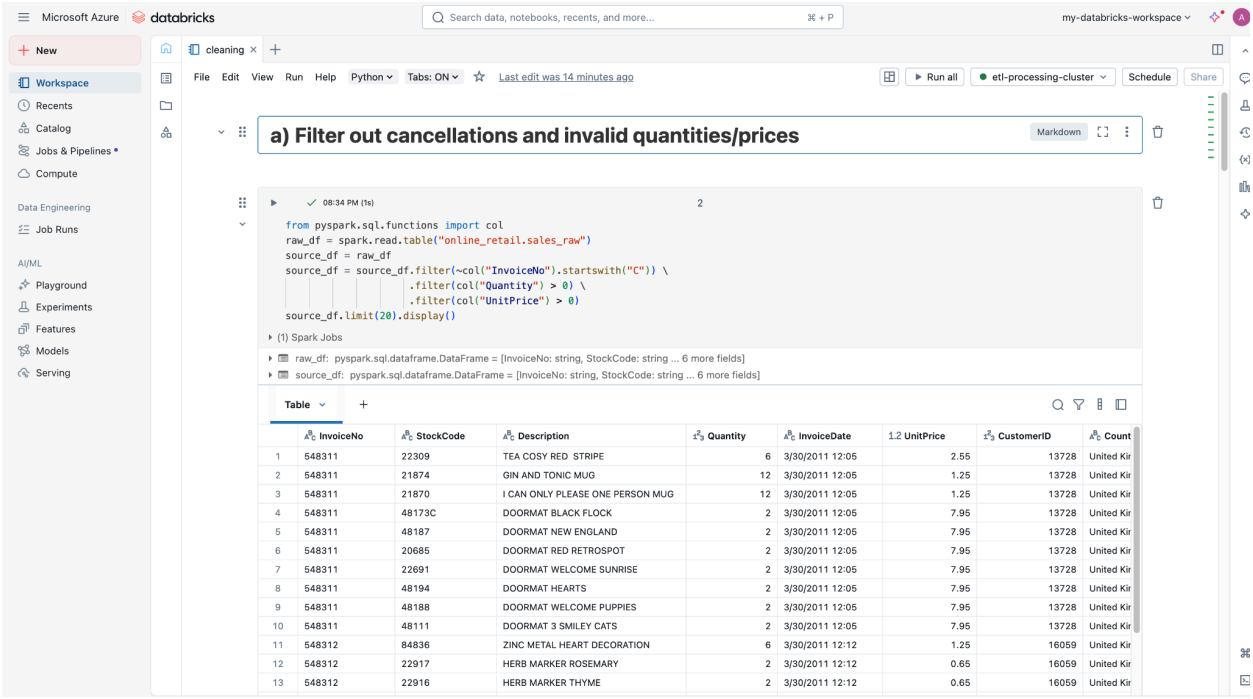
Data_Ingestion/read_data.ipynb

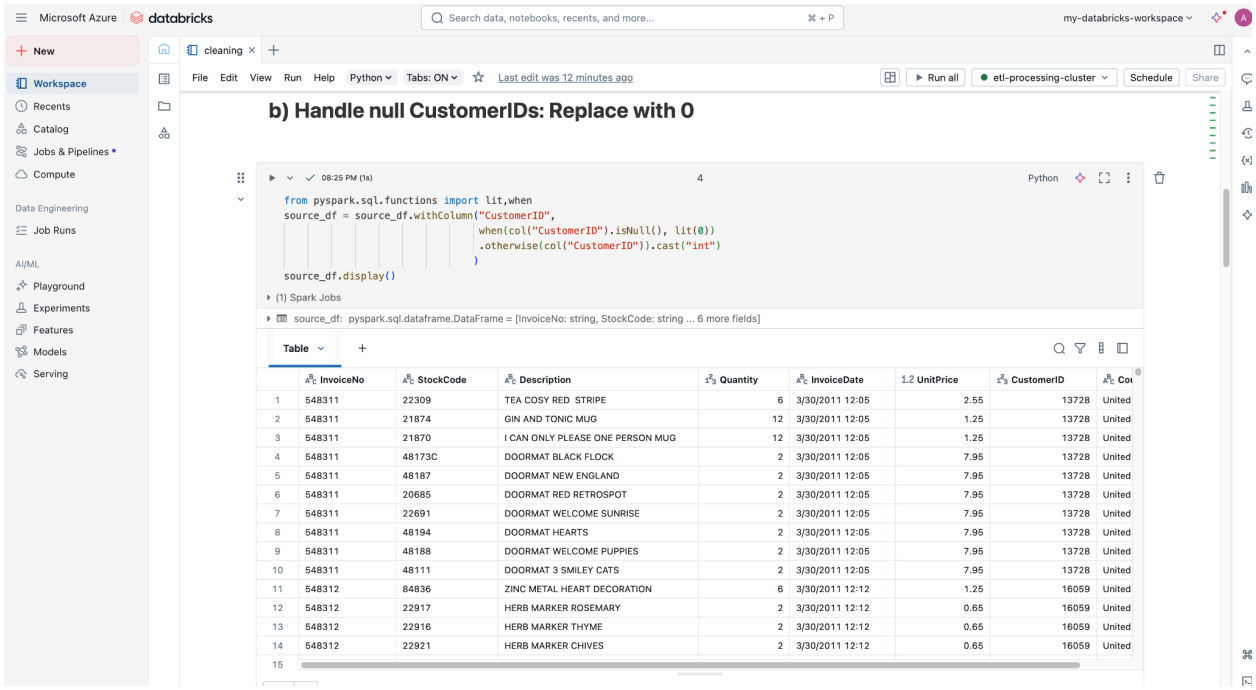


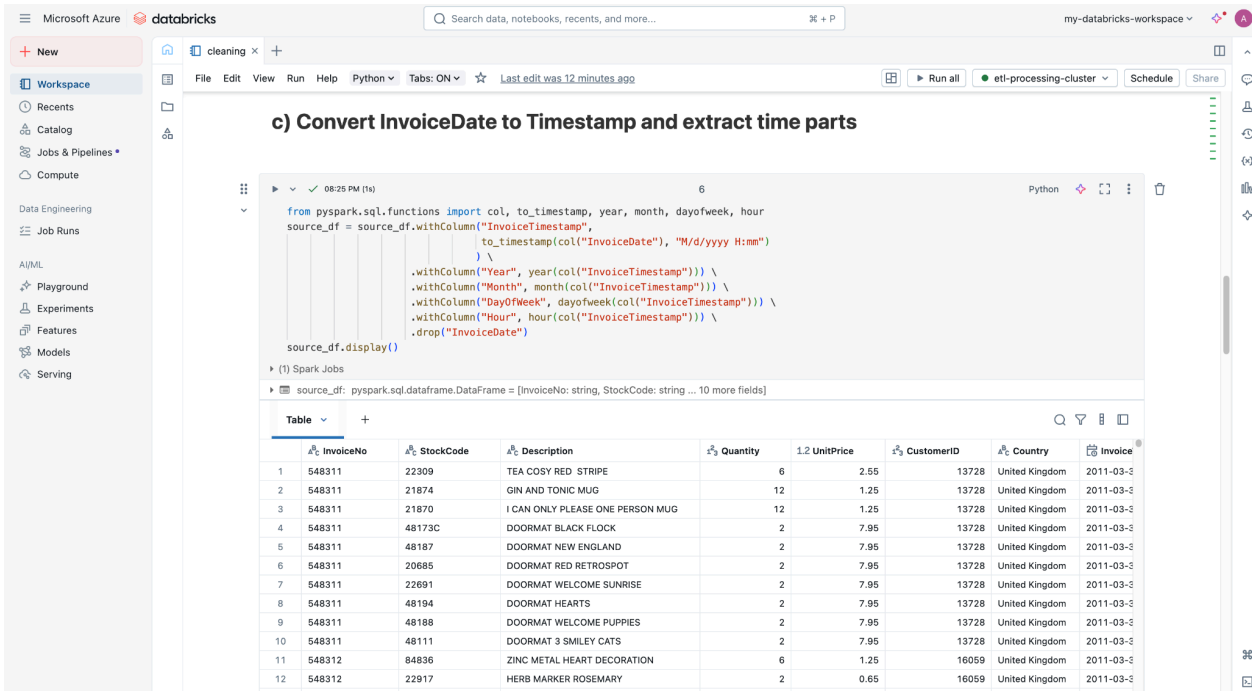


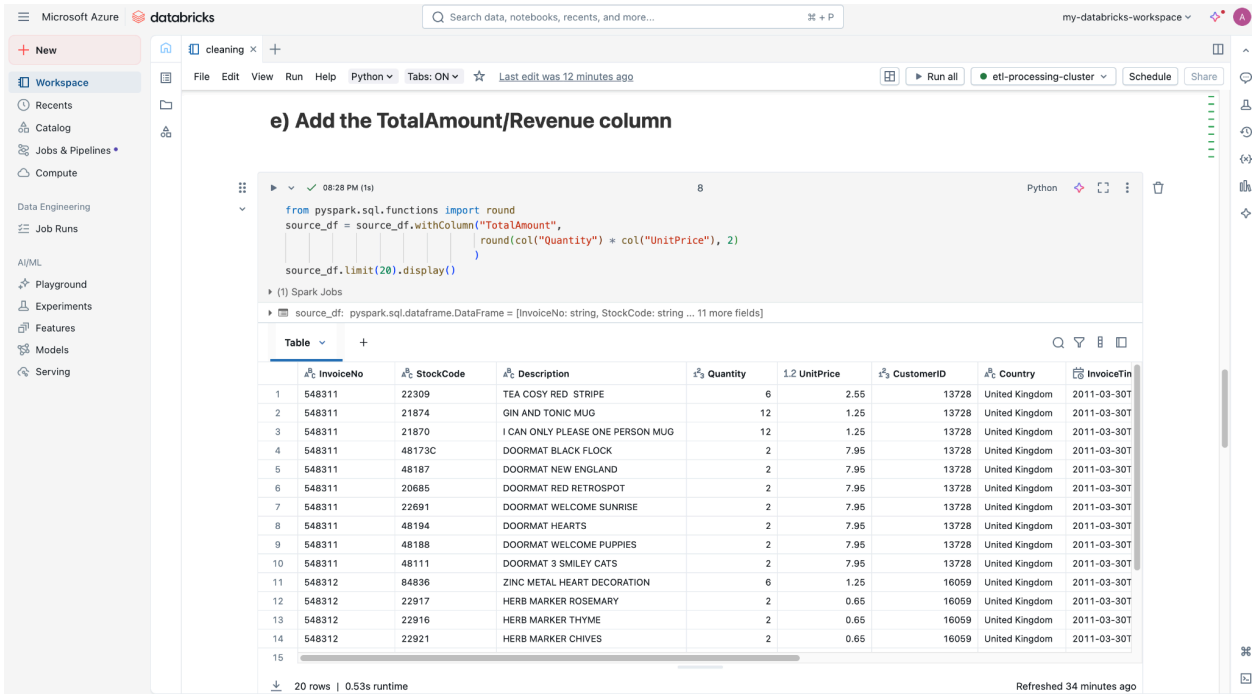


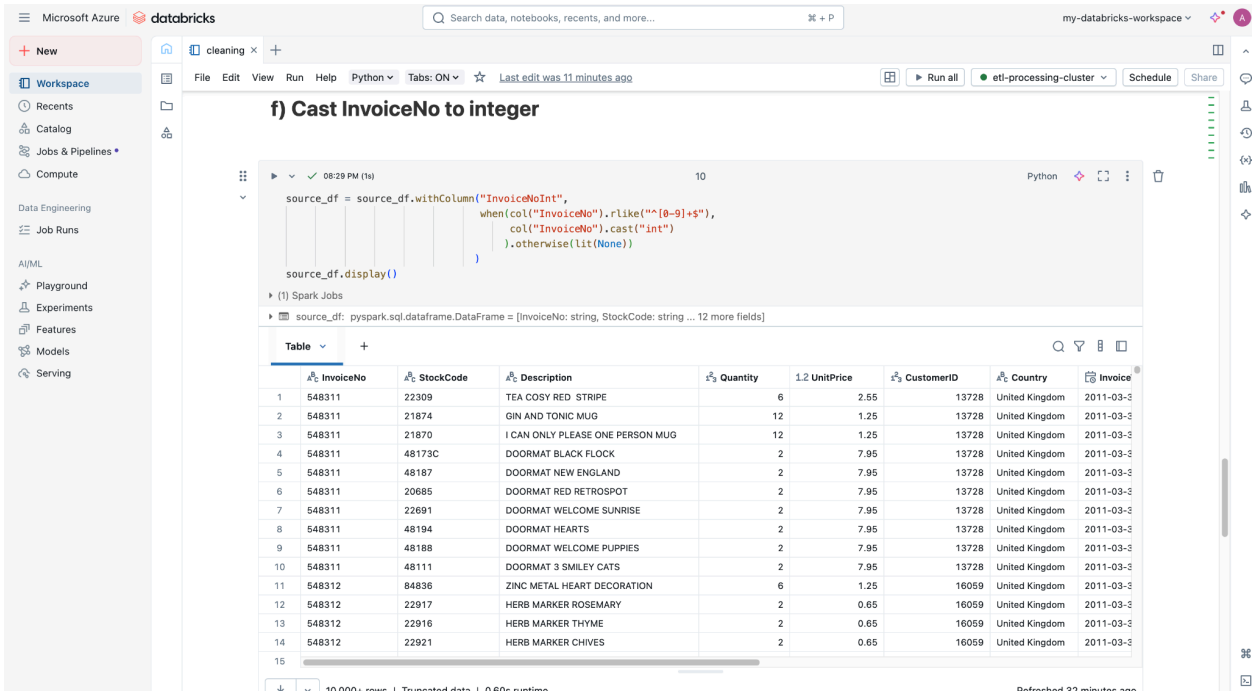
Data_Cleaning/cleaning.ipynb

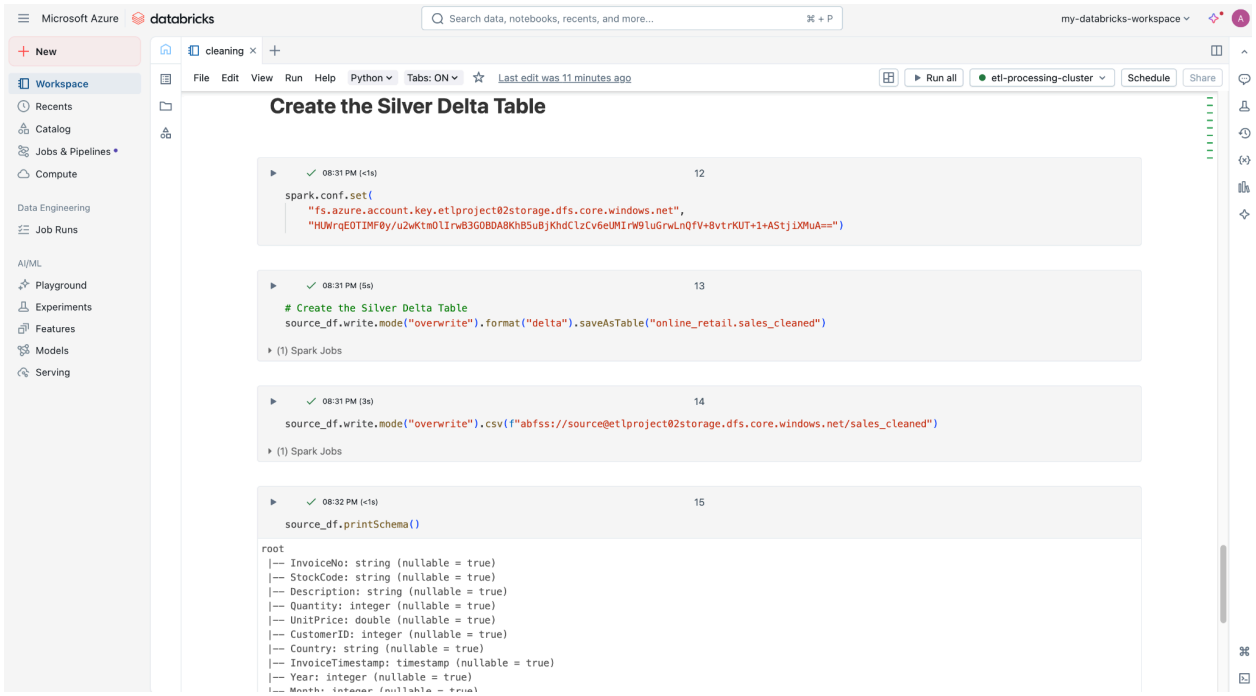



Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
Features
Models
Serving
cleaning
File Edit View Run Help Python Tabs: ON Last edit was 12 minutes ago
Run all
etl-processing-cluster
Schedule
Share
b) Handle null CustomerIDs: Replace with 0
08:25 PM (1s)
4
Python
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()
(1) Spark Jobs
source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
Table
InvoiceNo
StockCode
Description
Quantity
InvoiceDate
UnitPrice
CustomerID
1 548311 22309 TEA COSY RED STRIPE 6 3/30/2011 12:05 2.55 13728 United
2 548311 21874 GIN AND TONIC MUG 12 3/30/2011 12:05 1.25 13728 United
3 548311 21870 I CAN ONLY PLEASE ONE PERSON MUG 12 3/30/2011 12:05 1.25 13728 United
4 548311 48173C DOORMAT BLACK FLOCK 2 3/30/2011 12:05 7.95 13728 United
5 548311 48187 DOORMAT NEW ENGLAND 2 3/30/2011 12:05 7.95 13728 United
6 548311 20685 DOORMAT RED RETROSPOT 2 3/30/2011 12:05 7.95 13728 United
7 548311 22691 DOORMAT WELCOME SUNRISE 2 3/30/2011 12:05 7.95 13728 United
8 548311 48194 DOORMAT HEARTS 2 3/30/2011 12:05 7.95 13728 United
9 548311 48188 DOORMAT WELCOME PUPPIES 2 3/30/2011 12:05 7.95 13728 United
10 548311 48111 DOORMAT 3 SMILEY CATS 2 3/30/2011 12:05 7.95 13728 United
11 548312 84836 ZINC METAL HEART DECORATION 6 3/30/2011 12:12 1.25 16059 United
12 548312 22917 HERB MARKER ROSEMARY 2 3/30/2011 12:12 0.65 16059 United
13 548312 22916 HERB MARKER THYME 2 3/30/2011 12:12 0.65 16059 United
14 548312 22921 HERB MARKER CHIVES 2 3/30/2011 12:12 0.65 16059 United
15

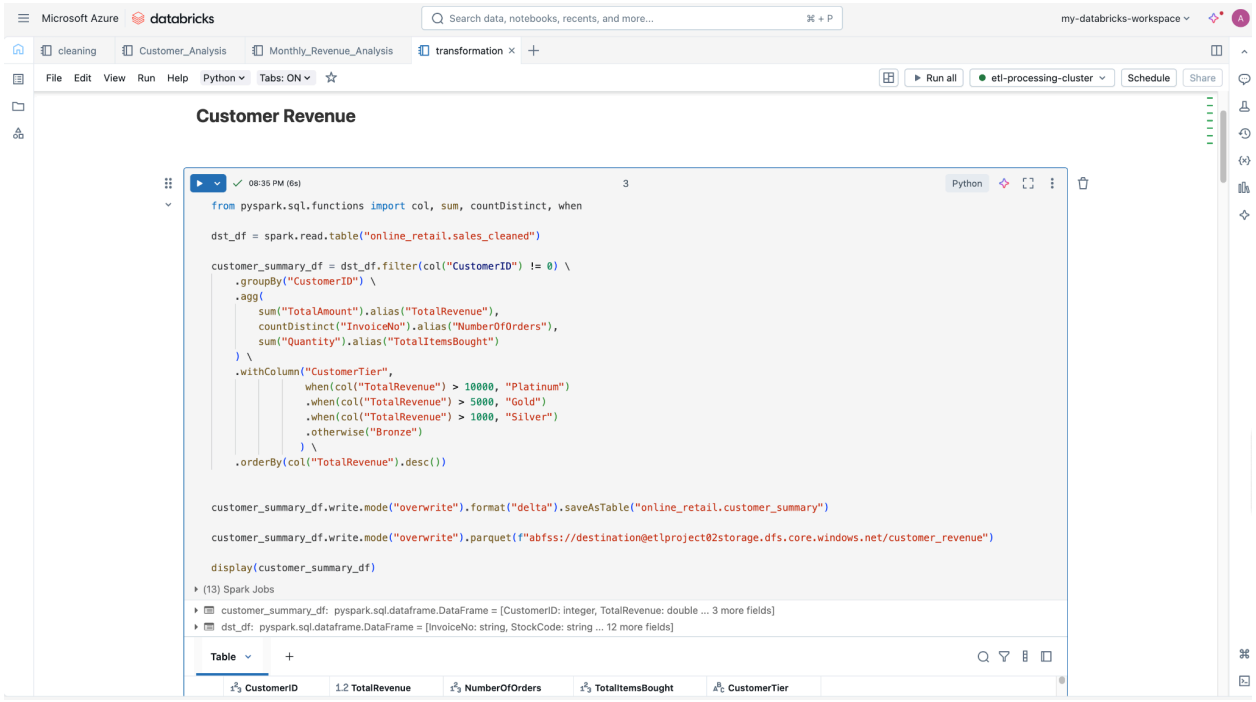

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
Features
Models
Serving
cleaning
File Edit View Run Help Python Tabs: ON Last edit was 12 minutes ago
Run all
etl-processing-cluster
Schedule
Share
c) Convert InvoiceDate to Timestamp and extract time parts
08:25 PM (1s)
6
Python
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()
(1) Spark Jobs
source_df: pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 10 more fields]
Table
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

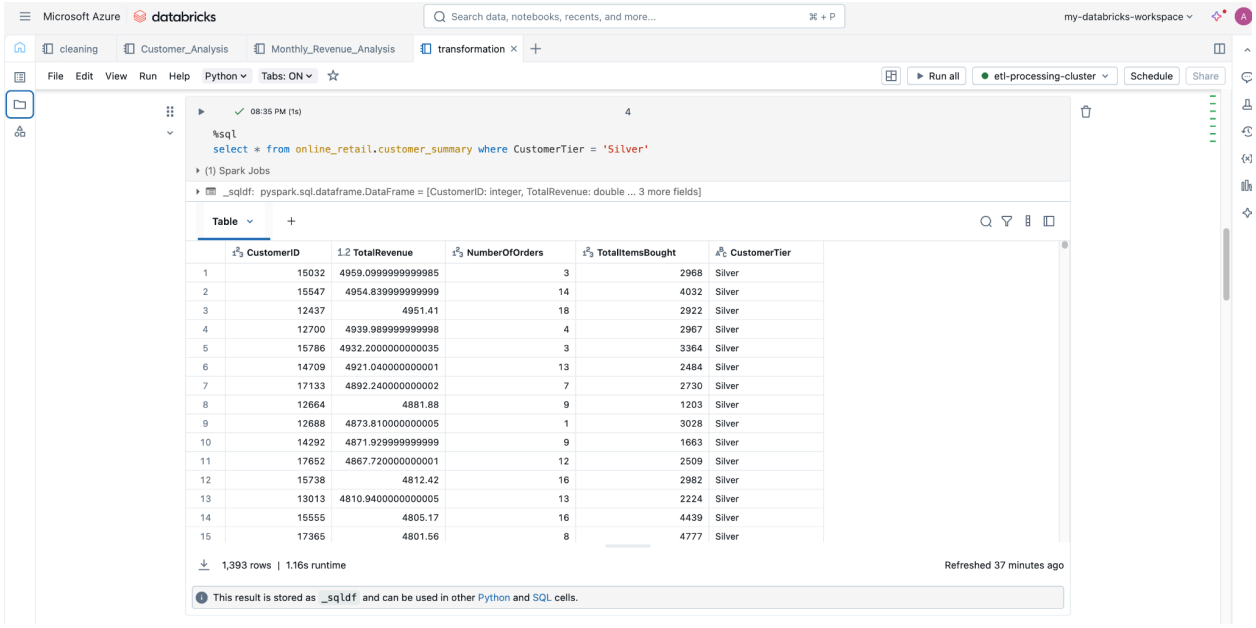

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
Features
Models
Serving
cleaning
File Edit View Run Help Python Tabs: ON Last edit was 12 minutes ago
Run all
etl-processing-cluster
Schedule
Share
e) Add the TotalAmount/Revenue column
08:28 PM (1s)
8
Python
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]
Table
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
15
20 rows | 0.53s runtime
Refreshed 34 minutes ago

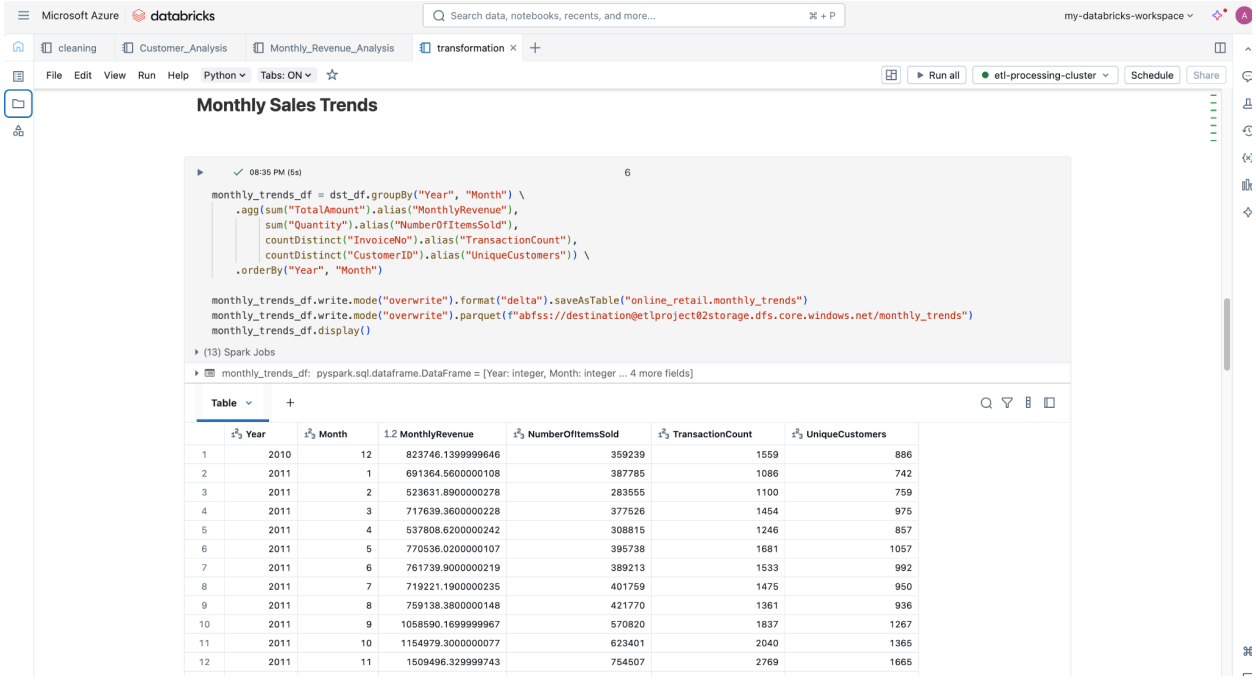

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
Features
Models
Serving
cleaning
File Edit View Run Help Python Tabs: ON Last edit was 11 minutes ago
Run all
etl-processing-cluster
Schedule
Share
f) Cast InvoiceNo to integer
08:29 PM (1s)
10
Python
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]
Table
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
15

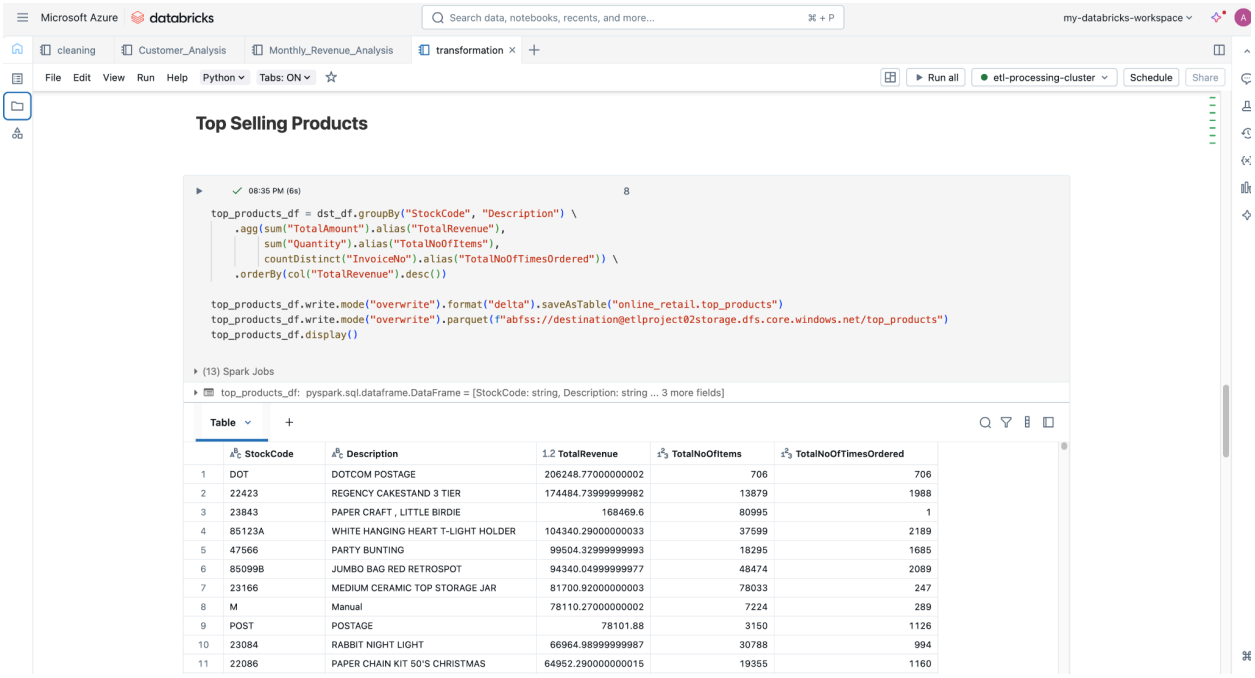


Data_Transformation/transformation.ipynb

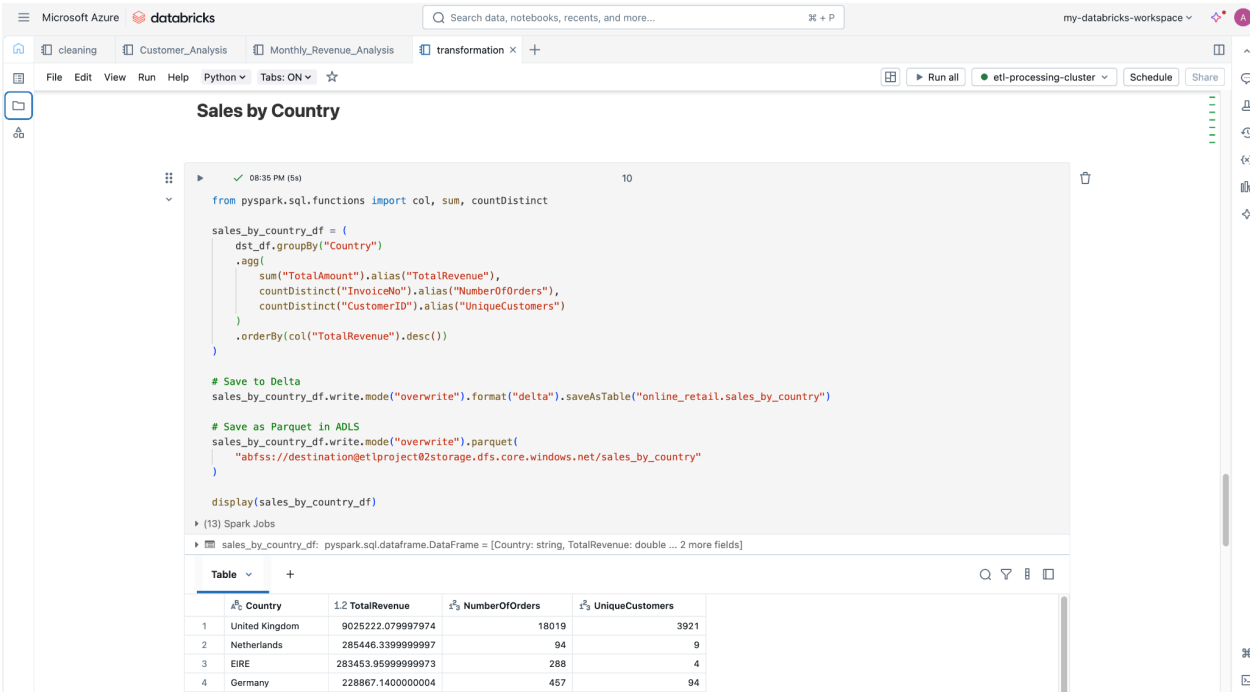



Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
File Edit View Run Help Python Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
08:35 PM (1s)
4
%sql
select * from online_retail.customer_summary where CustomerTier = 'Silver'
(1) Spark Jobs
_sqldf: pyspark.sql.dataframe.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 4939.989999999998 4 2967 Silver
5 15786 4932.2000000000035 3 3364 Silver
6 14709 4921.040000000001 13 2484 Silver
7 17133 4892.240000000002 7 2730 Silver
8 12664 4881.88 9 1203 Silver
9 12688 4873.810000000005 1 3028 Silver
10 14292 4871.929999999999 9 1663 Silver
11 17652 4867.720000000001 12 2509 Silver
12 15738 4812.42 16 2982 Silver
13 13013 4810.9400000000005 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.

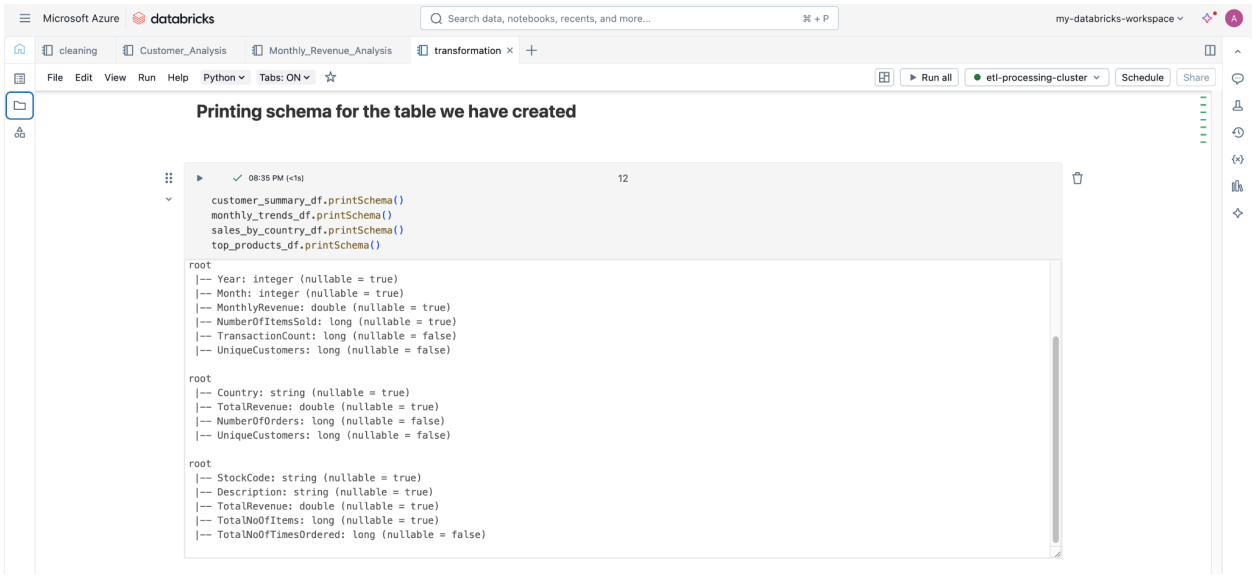

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
File Edit View Run Help Python Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
Monthly Sales Trends
08:35 PM (5s)
6
monthly_trends_df = 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()
(13) Spark Jobs
monthly_trends_df: pyspark.sql.dataframe.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 754507 2769 1665



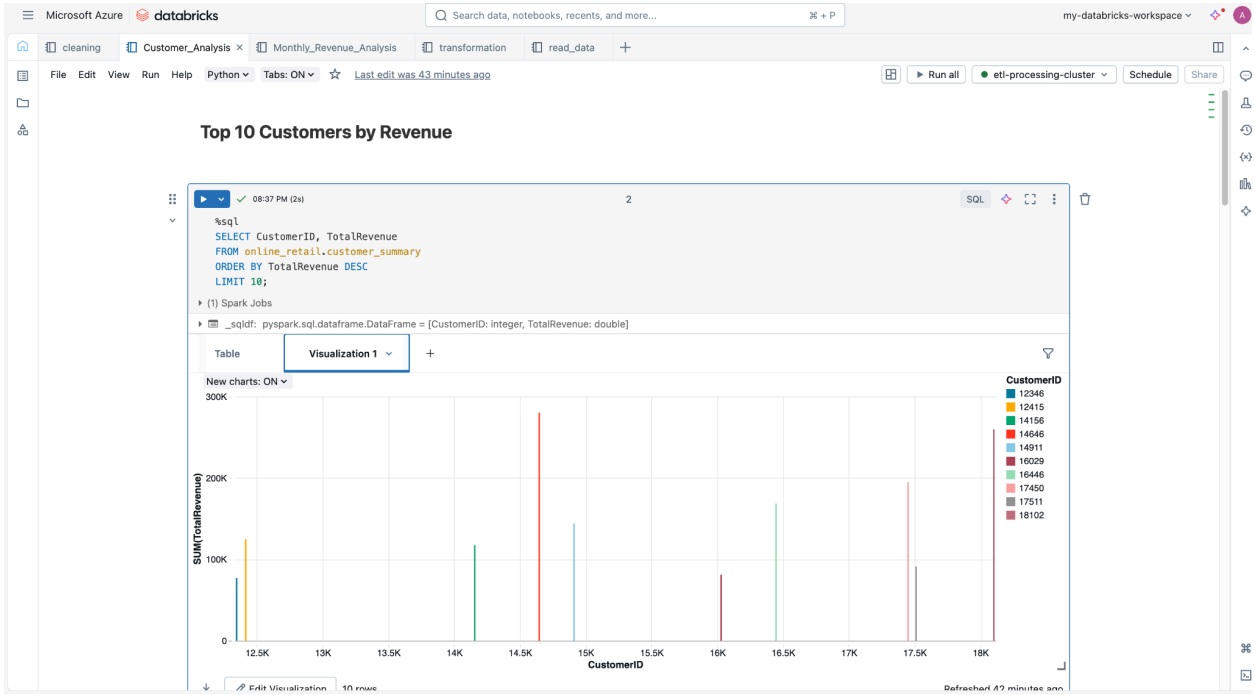
	StockCode	Description	TotalRevenue	TotalNoOfItems	TotalNoOfTimesOrdered
1	DOT	DOTCOM POSTAGE	206248.77000000002	706	706
2	22423	REGENCY CAKESTAND 3 TIER	174484.73999999982	13879	1988
3	23843	PAPER CRAFT , LITTLE BIRDIE	168469.6	80995	1
4	85123A	WHITE HANGING HEART T-LIGHT HOLDER	104340.29000000033	37599	2189
5	47566	PARTY BUNTING	99504.32999999993	18295	1685
6	85099B	JUMBO BAG RED RETROSPOT	94340.04999999977	48474	2089
7	23166	MEDIUM CERAMIC TOP STORAGE JAR	81700.92000000003	78033	247
8	M	Manual	78110.27000000002	7224	289
9	POST	POSTAGE	78101.88	3150	1126
10	23084	RABBIT NIGHT LIGHT	66964.98999999987	30788	994
11	22086	PAPER CHAIN KIT 50'S CHRISTMAS	64952.290000000015	19355	1160

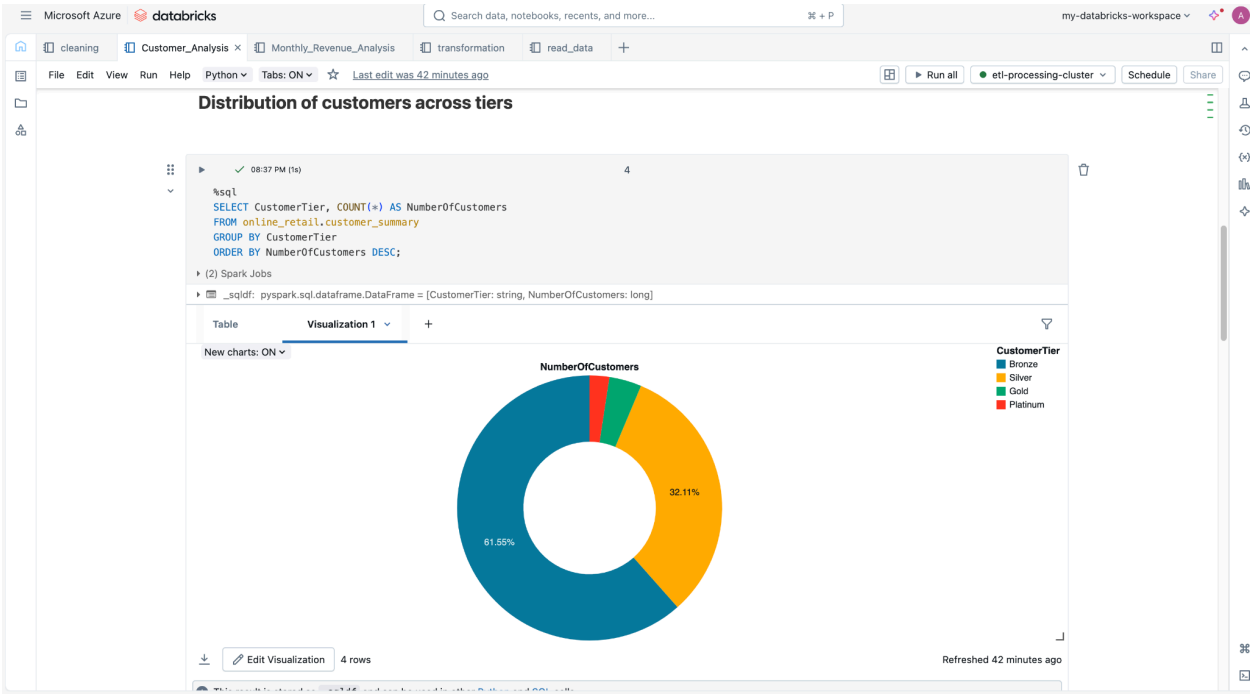


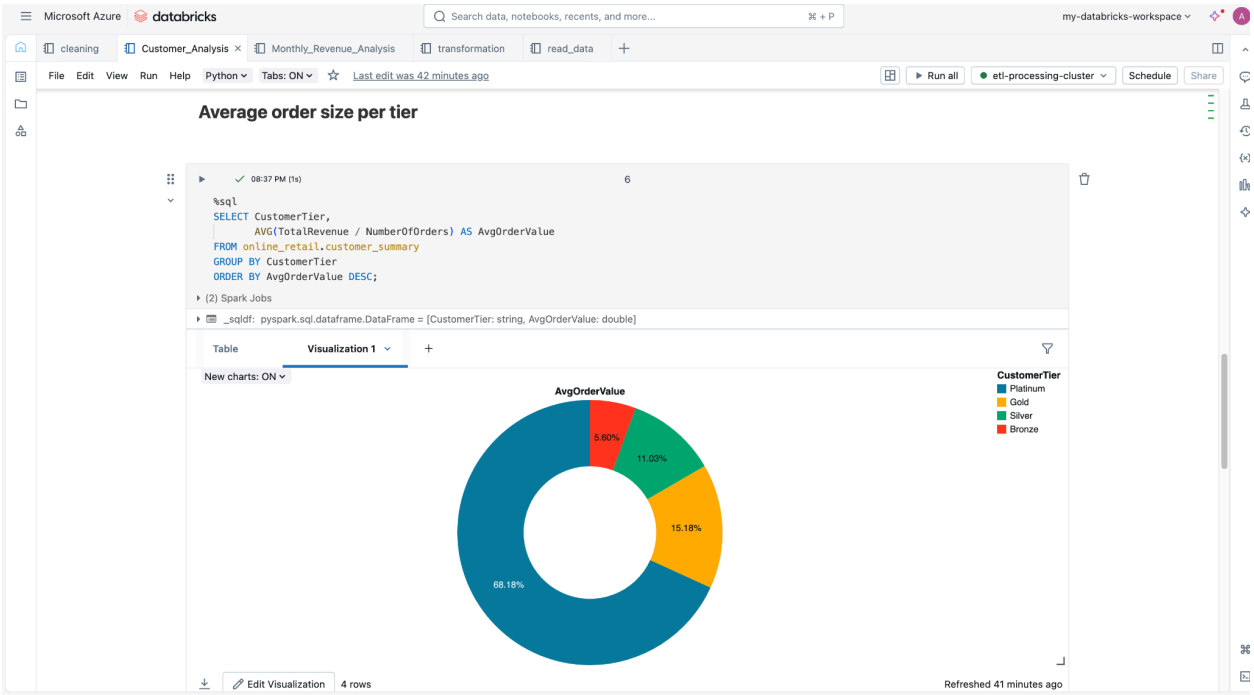
	Country	TotalRevenue	NumberOfOrders	UniqueCustomers
1	United Kingdom	9025222.079997974	18019	3921
2	Netherlands	285446.3399999997	94	9
3	EIRE	283453.95999999973	288	4
4	Germany	228867.1400000004	457	94

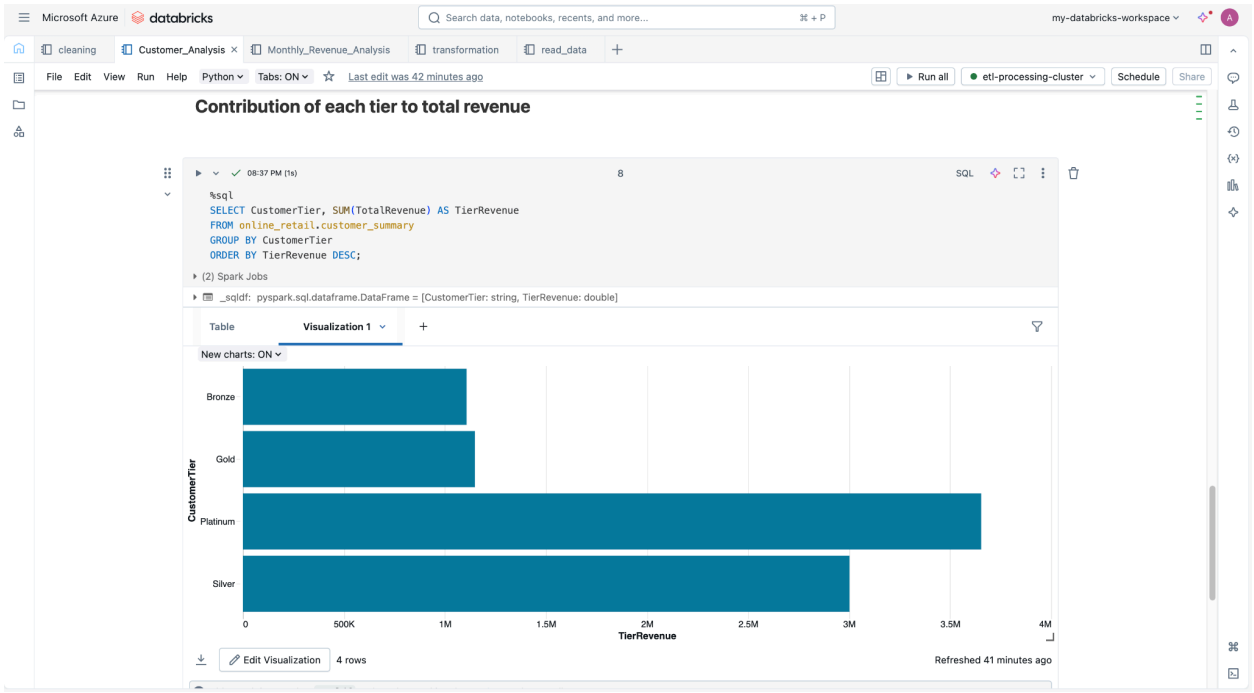


Data_Visualizaton/Customer_Analysis.ipynb

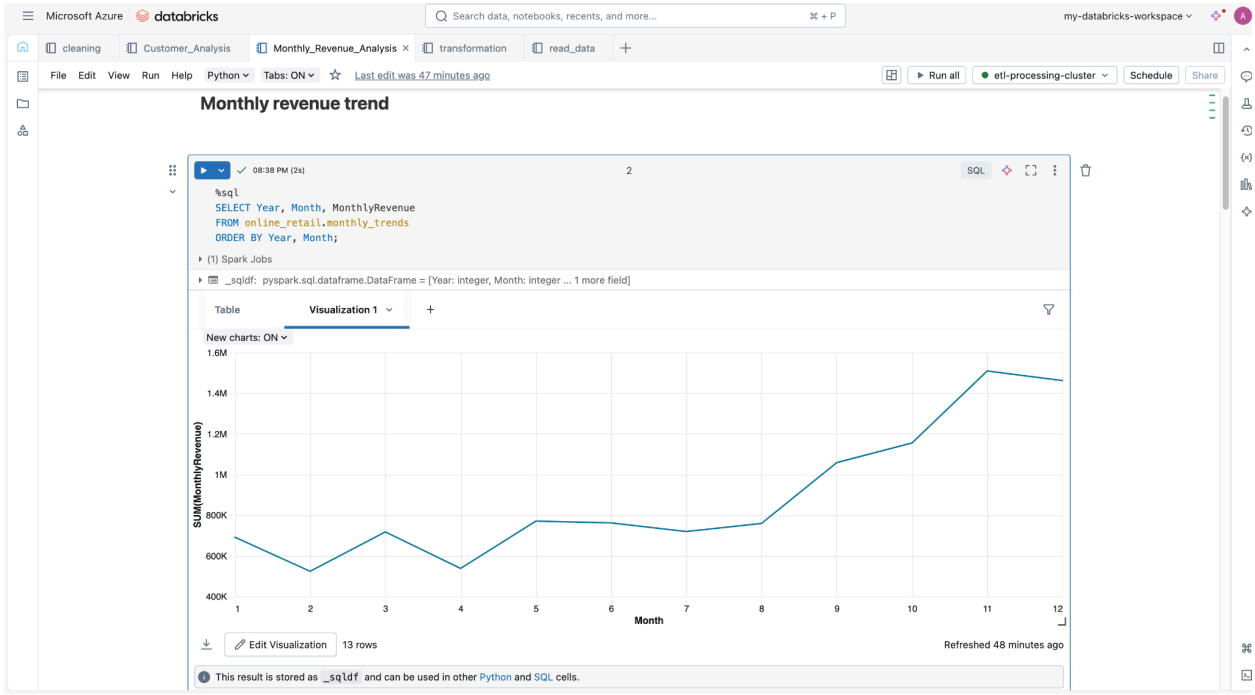


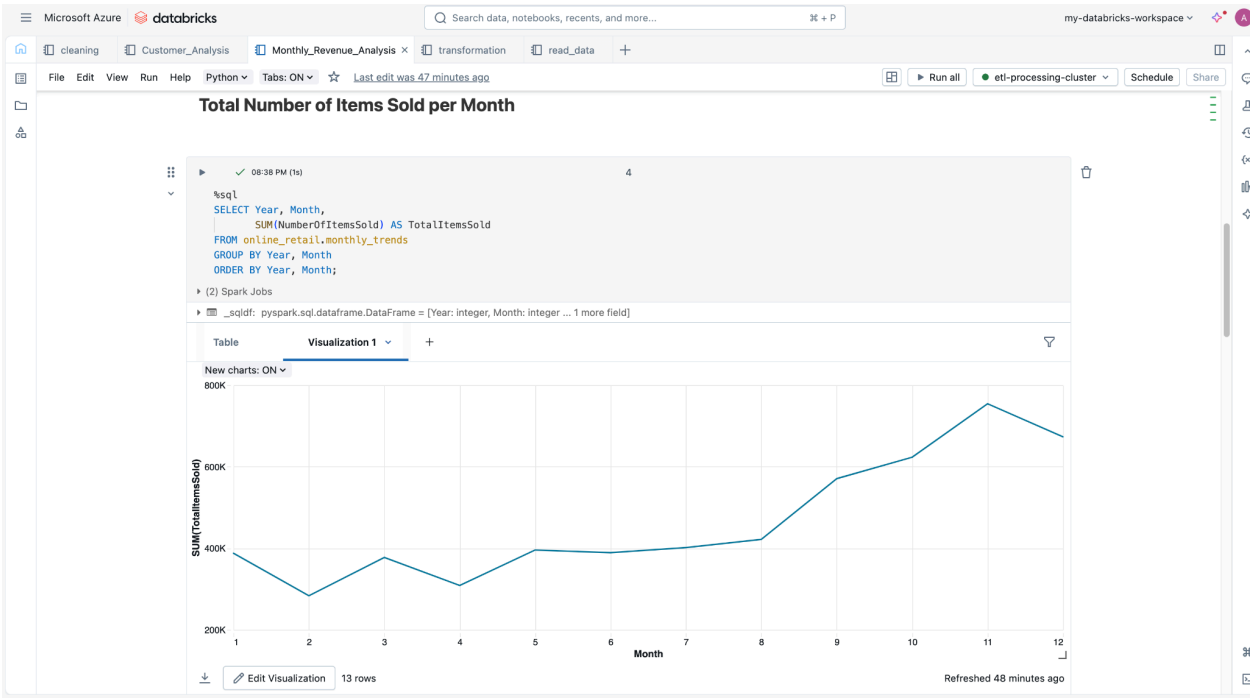

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Last edit was 42 minutes ago
Run all
etl-processing-cluster
Schedule
Share
Distribution of customers across tiers
08:37 PM (1s)
4
%sql
SELECT CustomerTier, COUNT(*) AS NumberOfCustomers
FROM online_retail.customer_summary
GROUP BY CustomerTier
ORDER BY NumberOfCustomers DESC;
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [CustomerTier: string, NumberOfCustomers: long]
Table
Visualization 1
New charts: ON
NumberOfCustomers
CustomerTier
Bronze
Silver
Gold
Platinum
32.11%
61.55%
Edit Visualization
4 rows
Refreshed 42 minutes ago

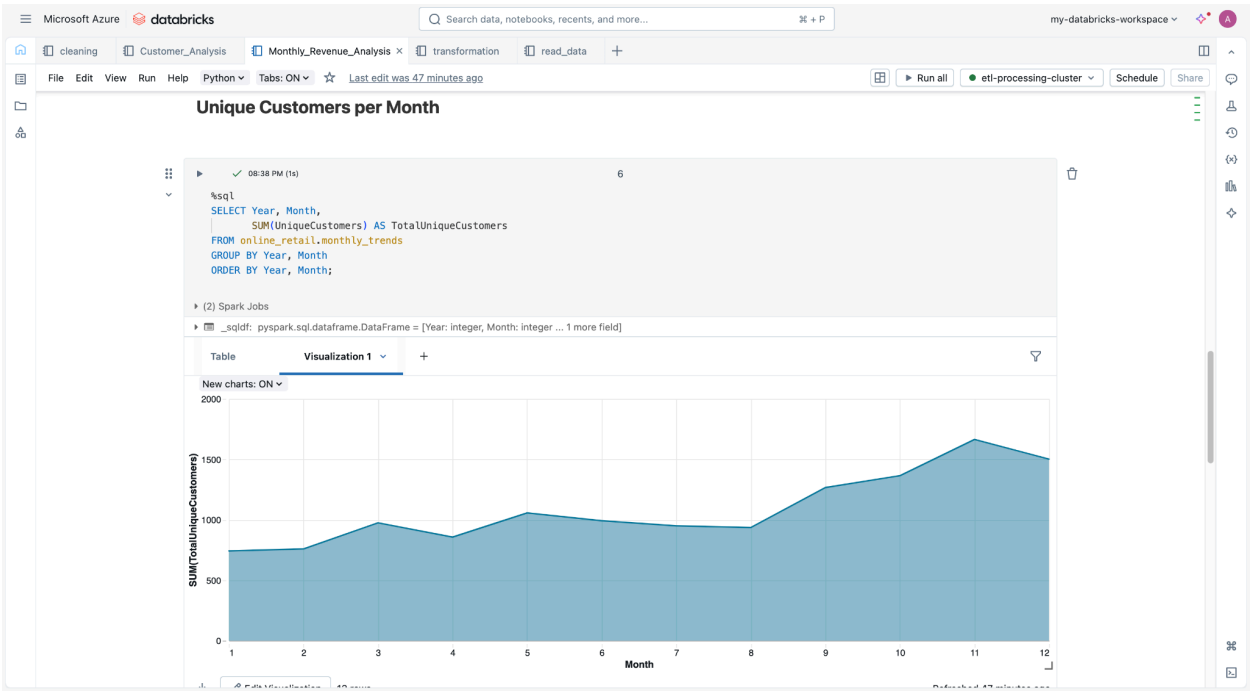

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Last edit was 42 minutes ago
Run all
etl-processing-cluster
Schedule
Share
Average order size per tier
08:37 PM (1s)
6
%sql
SELECT CustomerTier,
AVG(TotalRevenue / NumberOfOrders) AS AvgOrderValue
FROM online_retail.customer_summary
GROUP BY CustomerTier
ORDER BY AvgOrderValue DESC;
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [CustomerTier: string, AvgOrderValue: double]
Table
Visualization 1
New charts: ON
AvgOrderValue
CustomerTier
Platinum
Gold
Silver
Bronze
5.60%
11.03%
15.18%
68.18%
Edit Visualization
4 rows
Refreshed 41 minutes ago



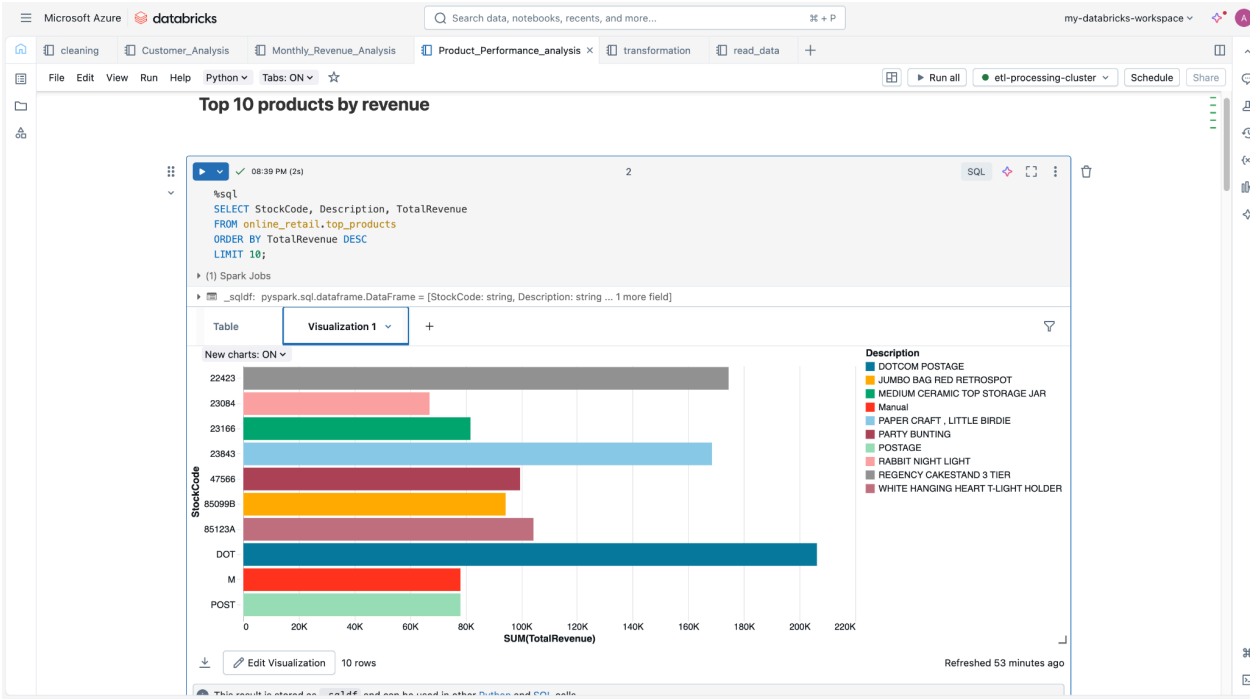
Data_Visualizaton/Monthly_Revenue_Analysis.ipynb

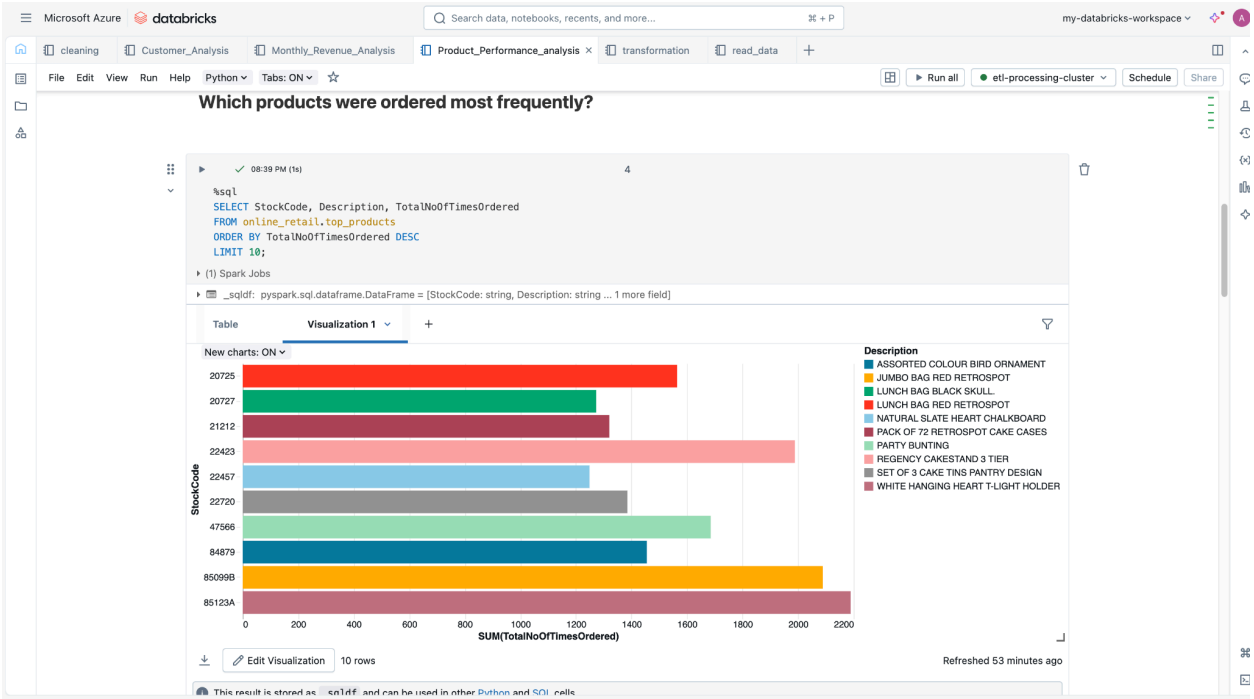


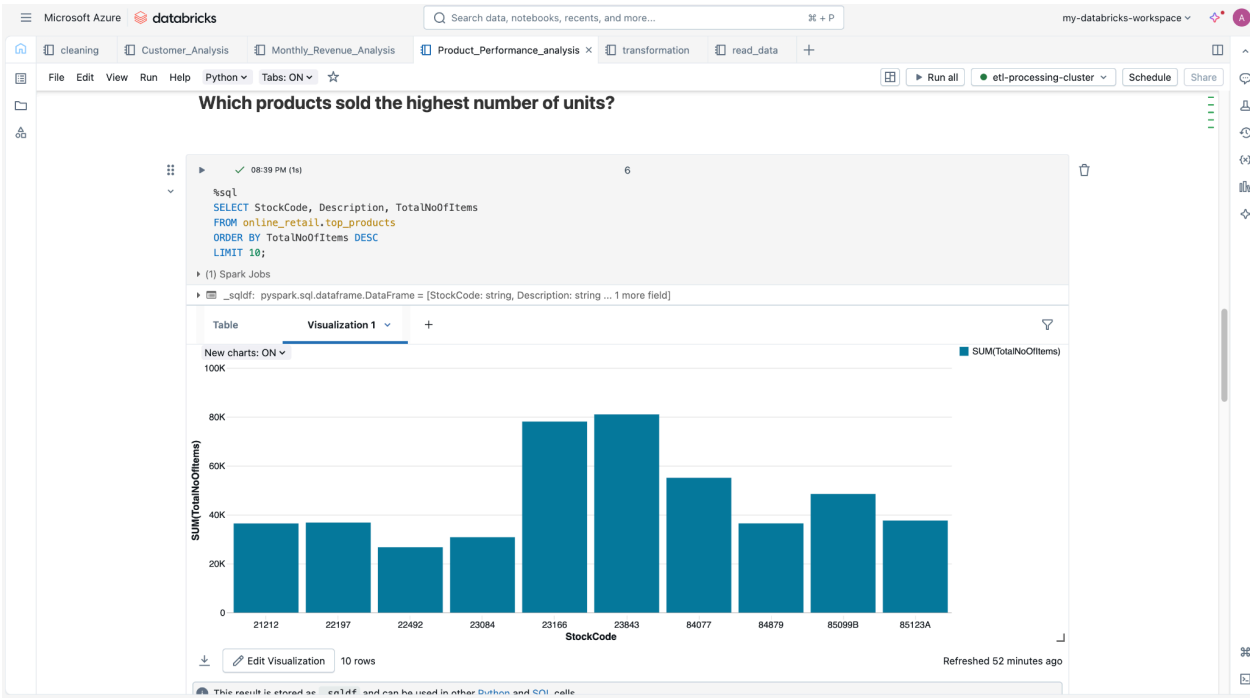

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Last edit was 47 minutes ago
Run all
etl-processing-cluster
Schedule
Share
Total Number of Items Sold per Month
08:38 PM (1s)
4
%sql
SELECT Year, Month,
SUM(NumberOfItemsSold) AS TotalItemsSold
FROM online_retail.monthly_trends
GROUP BY Year, Month
ORDER BY Year, Month;
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [Year: integer, Month: integer ... 1 more field]
Table
Visualization 1
New charts: ON
800K
600K
400K
200K
SUM(TotalItemsSold)
1 2 3 4 5 6 7 8 9 10 11 12
Month
Edit Visualization
13 rows
Refreshed 48 minutes ago

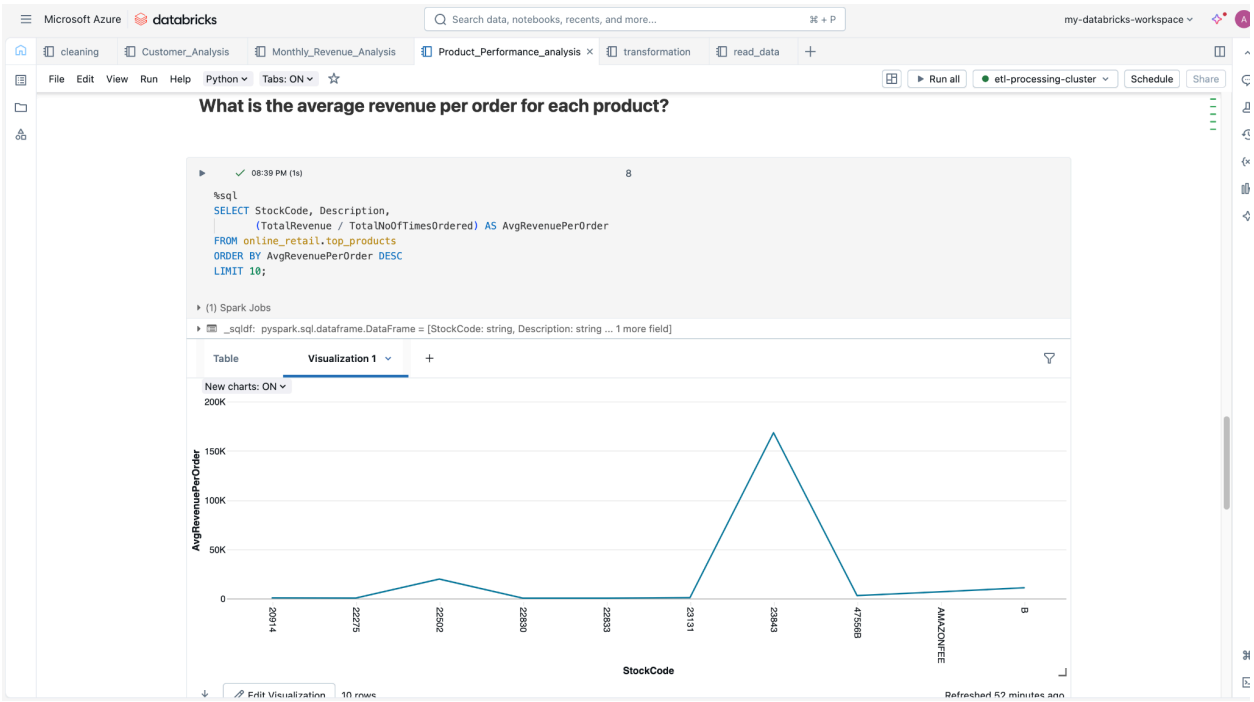

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Last edit was 47 minutes ago
Run all
etl-processing-cluster
Schedule
Share
Unique Customers per Month
08:38 PM (1s)
6
%sql
SELECT Year, Month,
SUM(UniqueCustomers) AS TotalUniqueCustomers
FROM online_retail.monthly_trends
GROUP BY Year, Month
ORDER BY Year, Month;
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [Year: integer, Month: integer ... 1 more field]
Table
Visualization 1
New charts: ON
2000
1500
1000
500
0
SUM(TotalUniqueCustomers)
1 2 3 4 5 6 7 8 9 10 11 12
Month

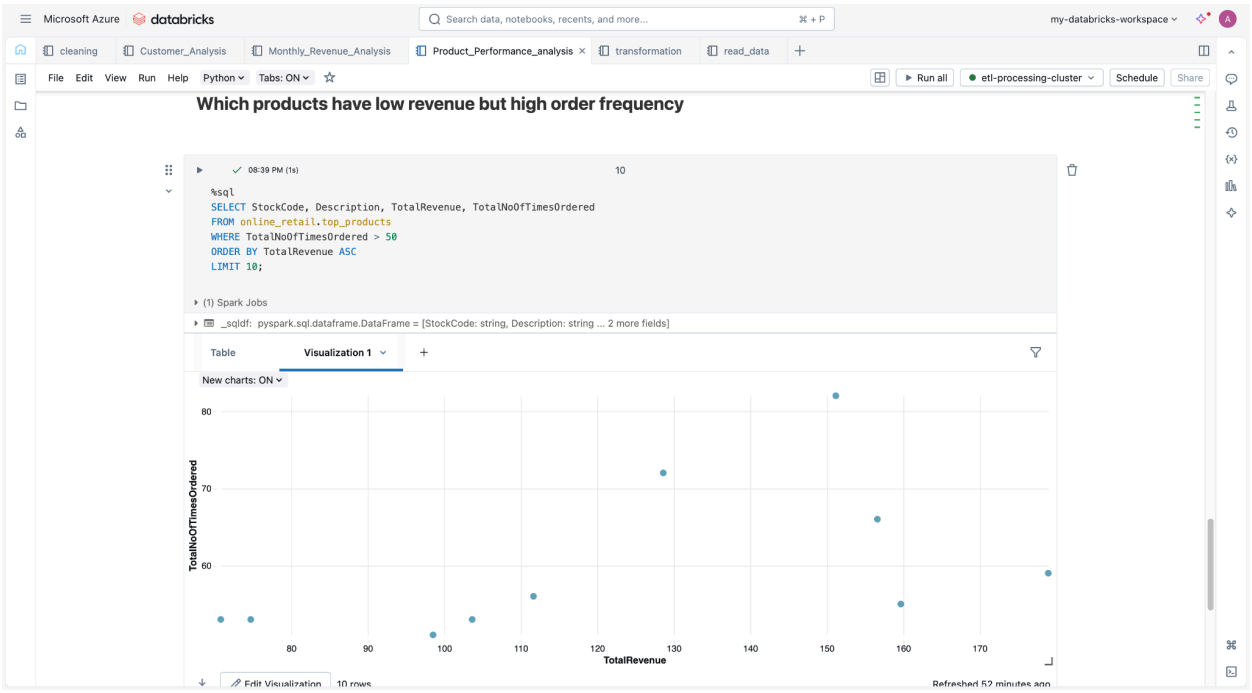
Data_Visualizaton/Product_Performance_Analysis.ipynb



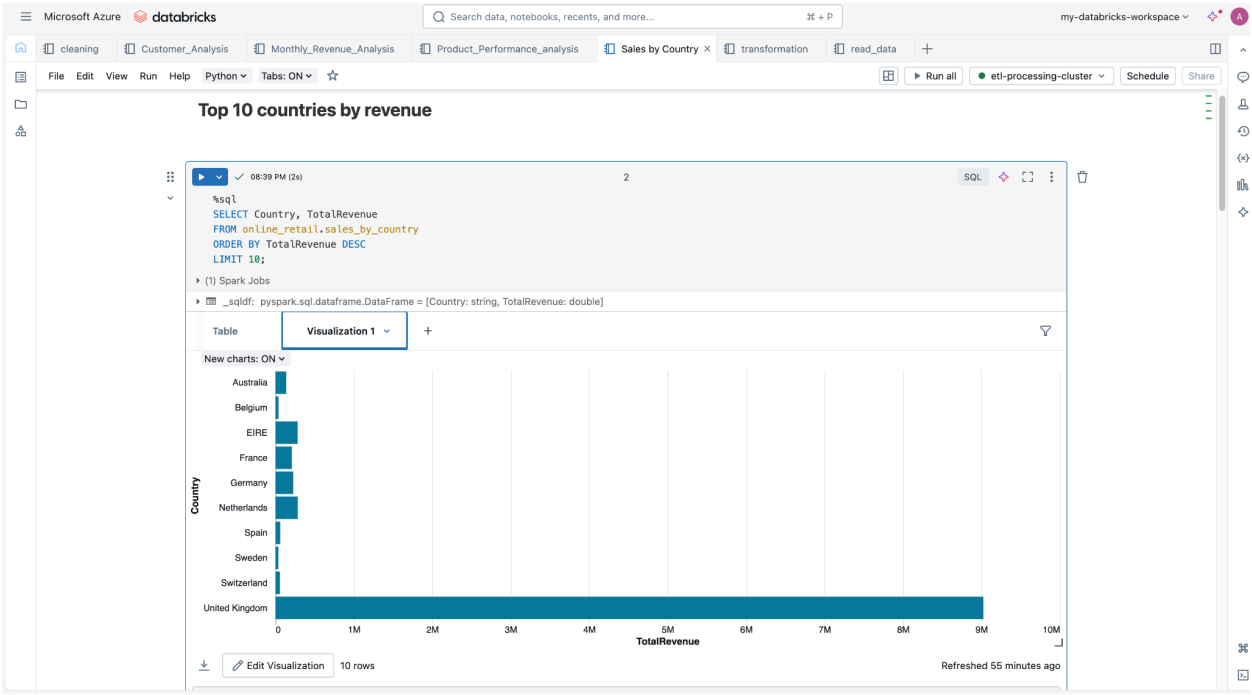


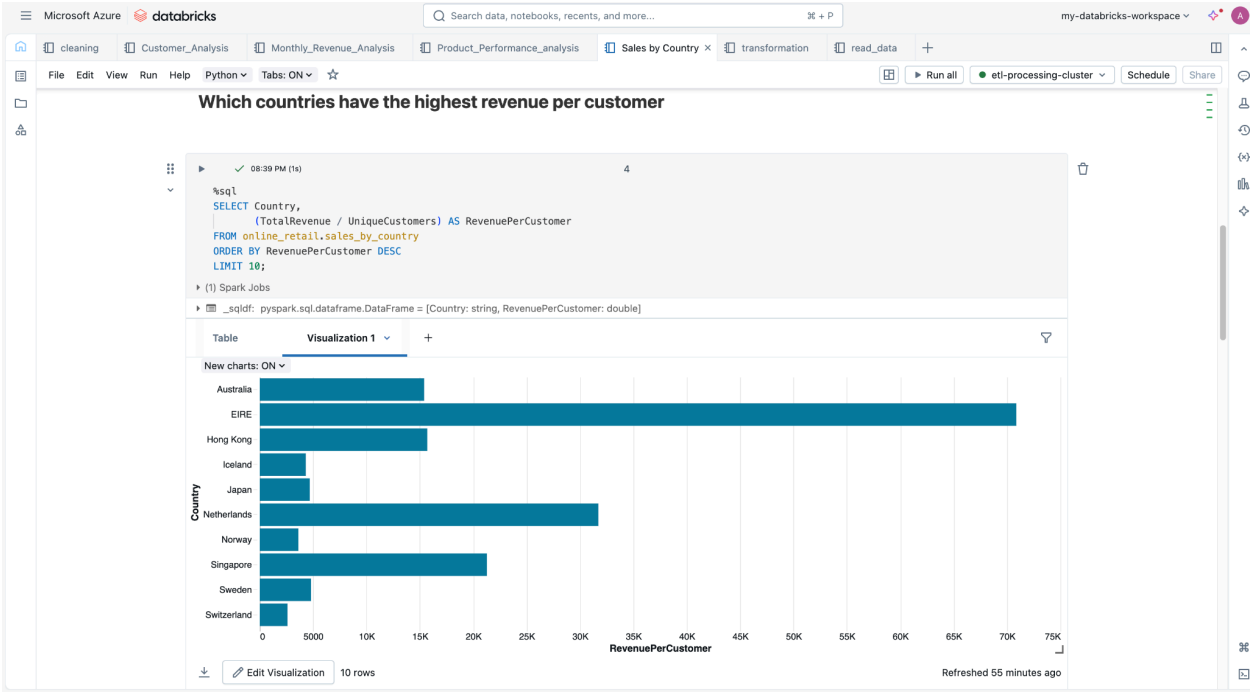

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
Product_Performance_analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
Which products sold the highest number of units?
08:39 PM (1s)
6
%sql
SELECT StockCode, Description, TotalNoOfItems
FROM online_retail.top_products
ORDER BY TotalNoOfItems DESC
LIMIT 10;
(1) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [StockCode: string, Description: string ... 1 more field]
Table
Visualization 1
New charts: ON
SUM(TotalNoOfItems)
100K
80K
60K
40K
20K
0
SUM(TotalNoOfItems)
21212
22197
22492
23084
23166
23843
84077
84879
85099B
85123A
StockCode
Edit Visualization
10 rows
Refreshed 52 minutes ago

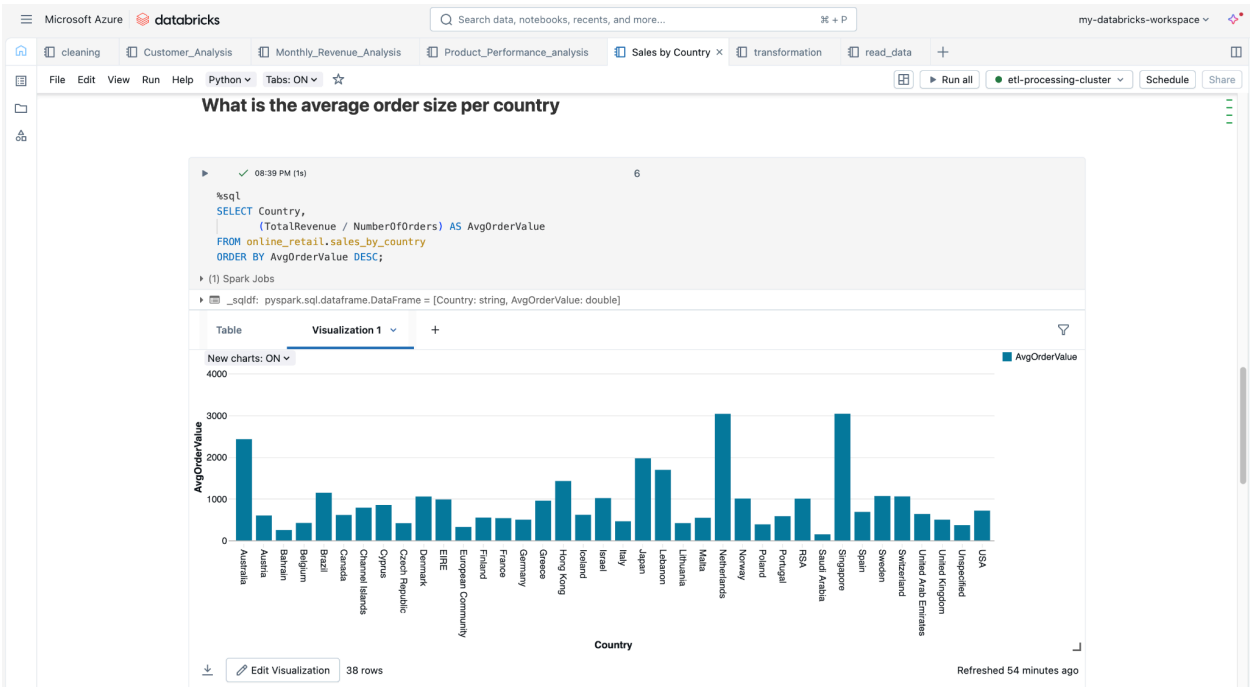

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
Product_Performance_analysis
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
What is the average revenue per order for each product?
08:39 PM (1s)
8
%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.DataFrame = [StockCode: string, Description: string ... 1 more field]
Table
Visualization 1
New charts: ON
200K
150K
100K
50K
0
AvgRevenuePerOrder
20914
22275
22502
22830
22833
23131
23843
47566B
AMAZONFEE
B
StockCode
Edit Visualization
10 rows
Refreshed 52 minutes ago

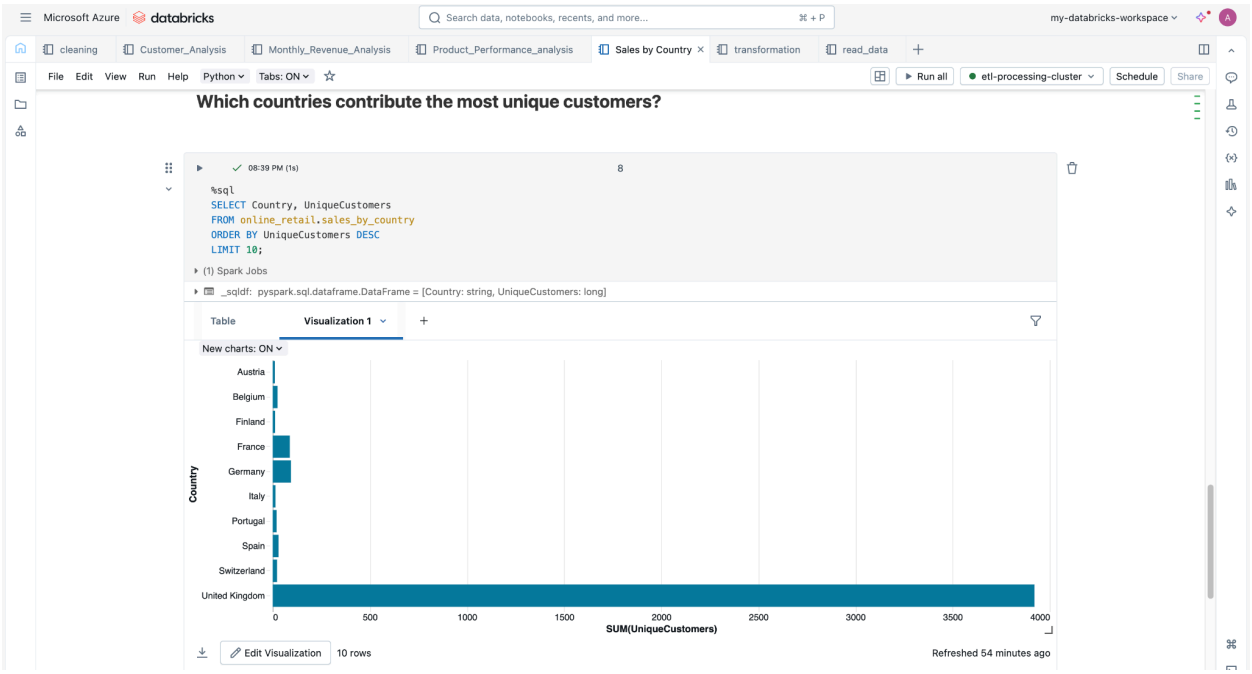


Data_Visualizaton/Sales_by_Country.ipynb




Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
Product_Performance_analysis
Sales by Country
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
Which countries have the highest revenue per customer
08:39 PM (1s)
4
%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.DataFrame = [Country: string, RevenuePerCustomer: double]
Table
Visualization 1
New charts: ON
Australia
EIRE
Hong Kong
Iceland
Japan
Netherlands
Norway
Singapore
Sweden
Switzerland
Country
0
5000
10K
15K
20K
25K
30K
35K
40K
45K
50K
55K
60K
65K
70K
75K
RevenuePerCustomer
Edit Visualization
10 rows
Refreshed 55 minutes ago

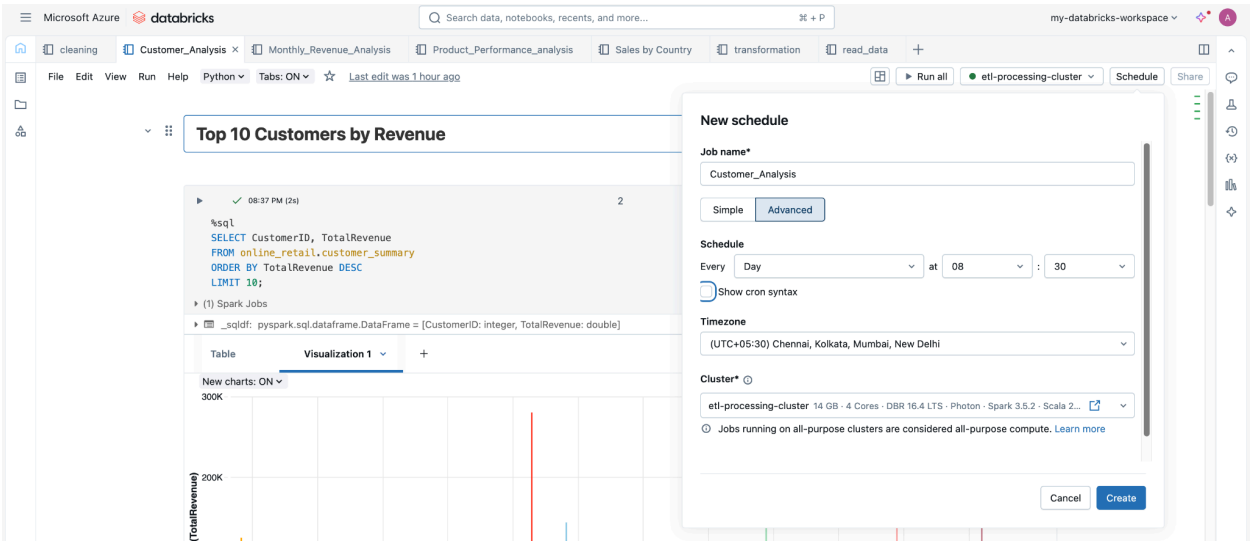

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
my-databricks-workspace
cleaning
Customer_Analysis
Monthly_Revenue_Analysis
Product_Performance_analysis
Sales by Country
transformation
read_data
File Edit View Run Help
Python
Tabs: ON
Run all
etl-processing-cluster
Schedule
Share
What is the average order size per country
08:39 PM (1s)
6
%sql
SELECT Country,
(TotalRevenue / NumberOfOrders) AS AvgOrderValue
FROM online_retail.sales_by_country
ORDER BY AvgOrderValue DESC;
(1) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [Country: string, AvgOrderValue: double]
Table
Visualization 1
New charts: ON
AvgOrderValue
4000
3000
2000
1000
0
Australia
Austria
Bahrain
Belgium
Brazil
Canada
Channel Islands
Cyprus
Czech Republic
Denmark
EIRE
European Community
Finland
France
Germany
Greece
Hong Kong
Iceland
Israel
Italy
Japan
Lebanon
Lithuania
Malta
Netherlands
Norway
Poland
Portugal
RSA
Saudi Arabia
Singapore
Spain
Sweden
Switzerland
United Arab Emirates
United Kingdom
Unspecified
USA
Country
Edit Visualization
38 rows
Refreshed 54 minutes ago



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

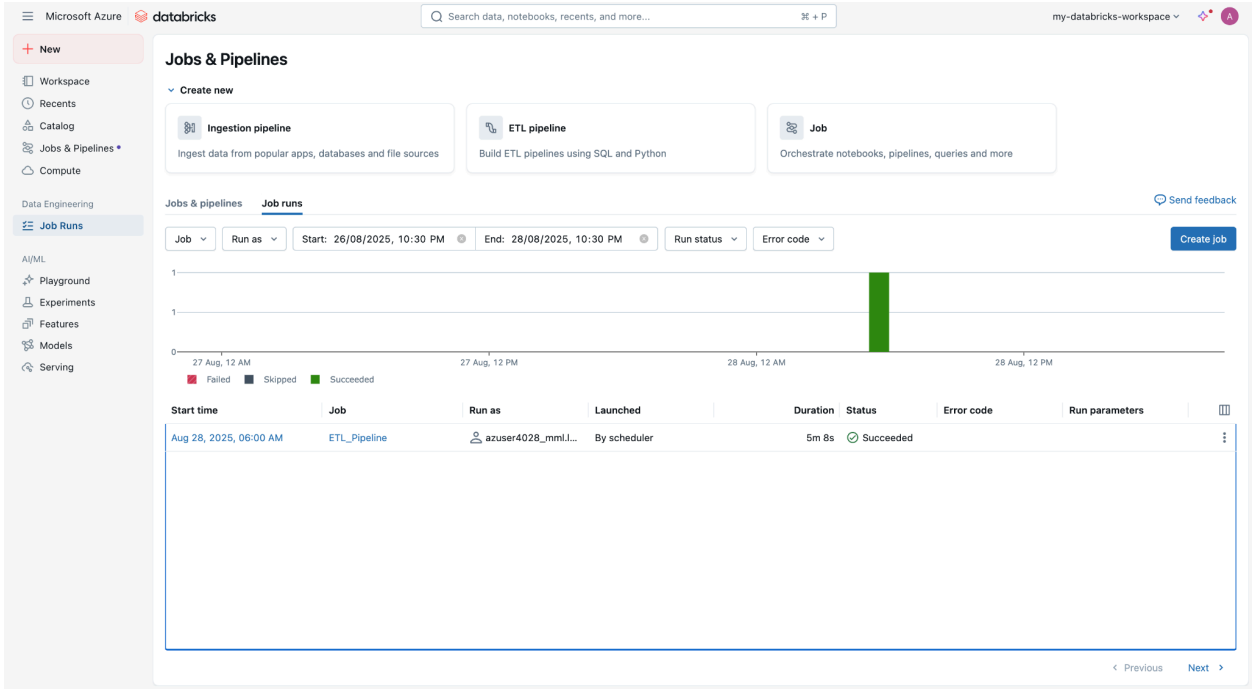
Step 5: Schedule a job:

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.

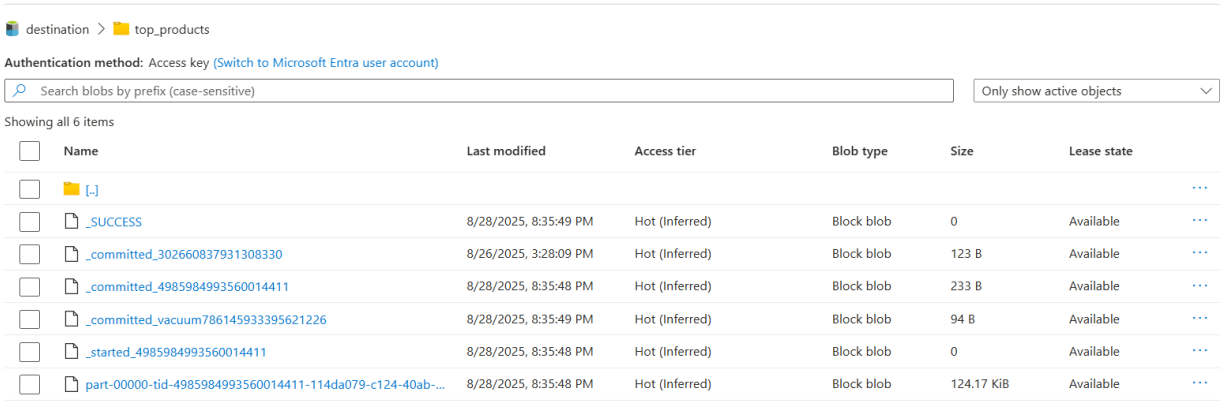


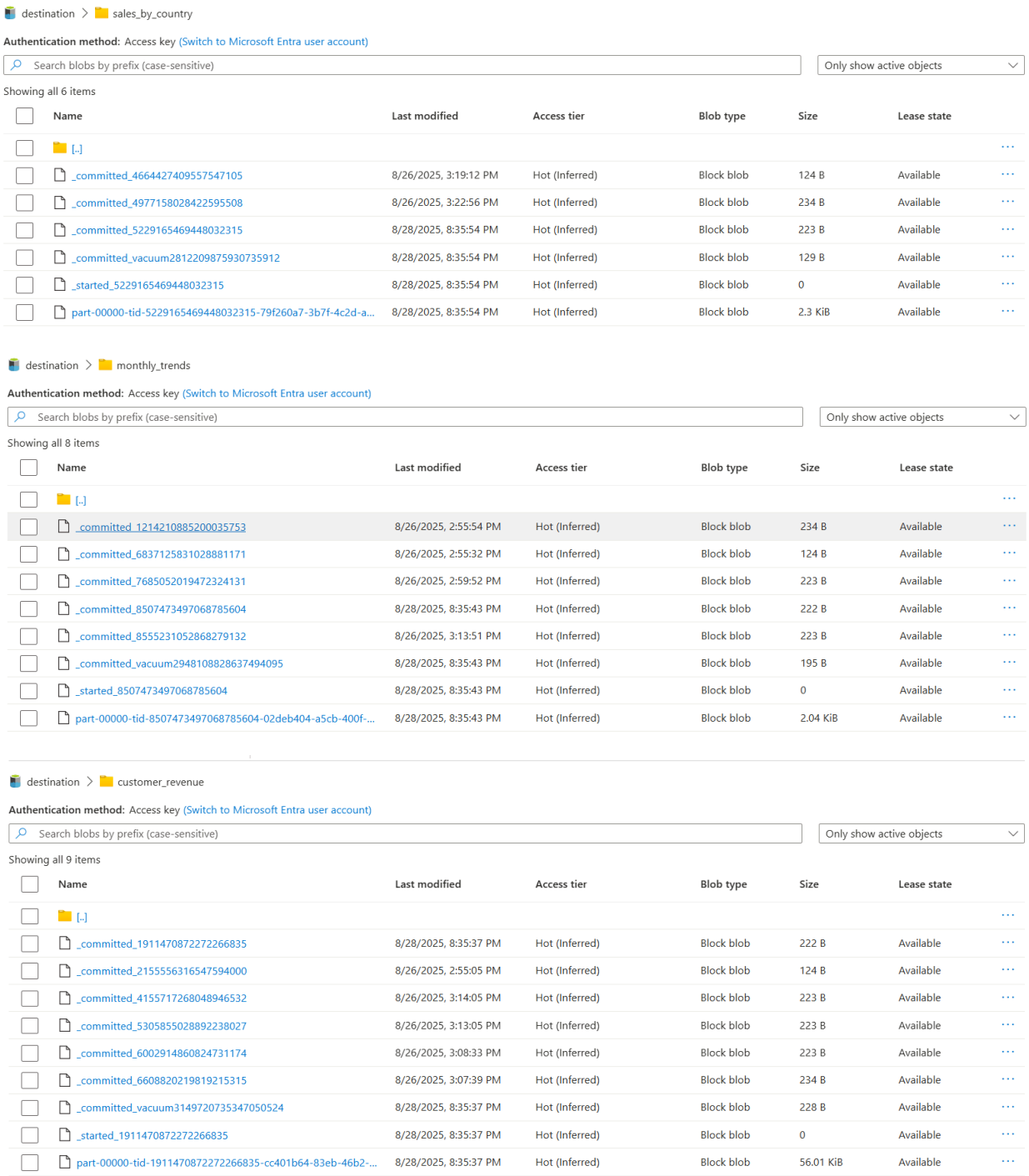
2. Monitoring the Scheduled Jobs

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



Successful output generated





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.