

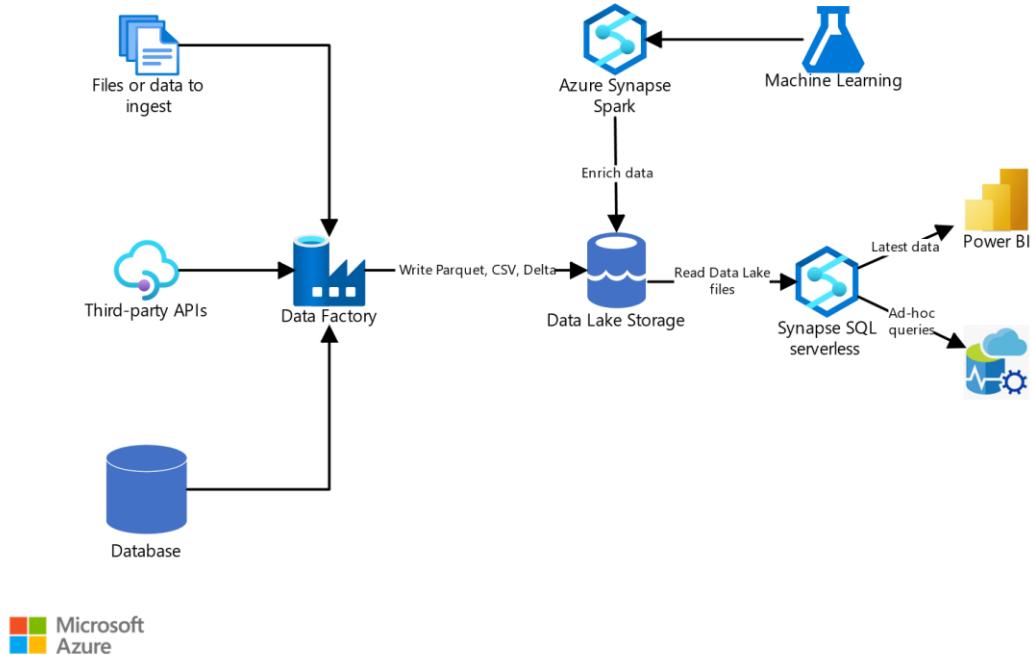
Data-Engineering-Project-Using-Azure-Databricks

Data Lake Exploration and Optimization

Project Overview:

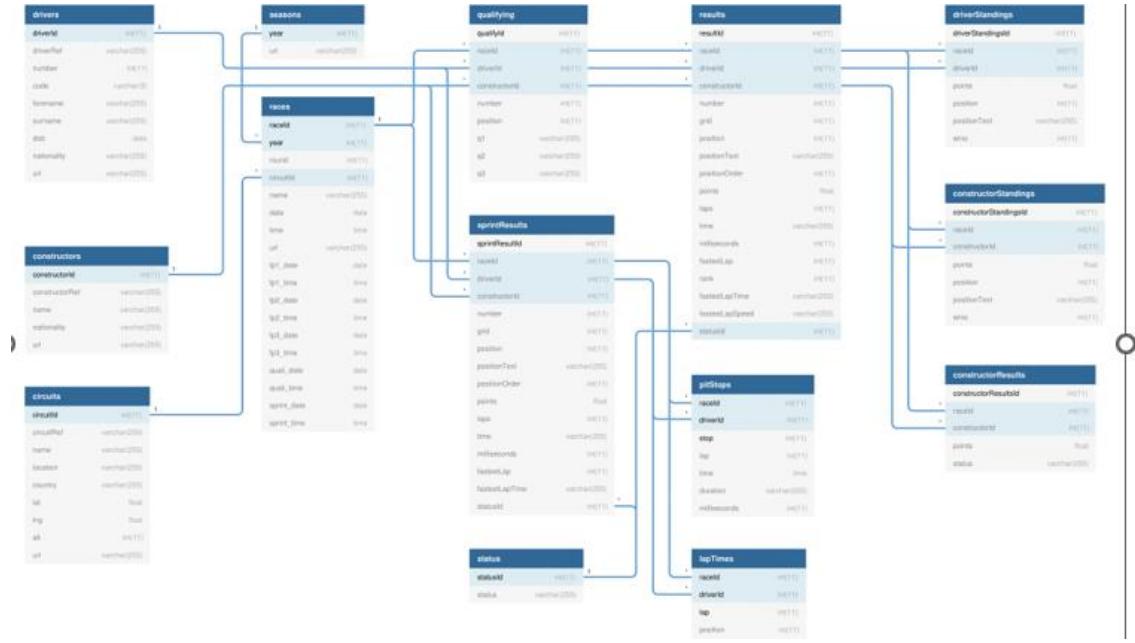
Objective: This project aims to explore and optimize data stored in an Azure Data Lake using PySparkSQL on the Azure Databricks platform. By leveraging PySparkSQL's functionalities, you'll gain insights into the data and improve its efficiency for further analysis or downstream applications.

Architecture diagram:



ER Diagram:

The structure of the database is shown in the following ER Diagram and explained in the Database File.



Components:

- Azure Data Lake Storage (ADLS)**: Stores the raw data in various formats (parquet, CSV, etc.).
- Azure Databricks Cluster**: Provides the processing environment with PySpark installed.
- PySpark Notebook**: Contains Python code using PySparkSQL functions for data exploration, transformation, and optimization.
- Data Visualization Tool (Optional)**: Used to create visualizations like charts or graphs from the processed data (e.g., Matplotlib, Plotly).
- Reporting Tool (Optional)**: Used to generate reports based on the analyzed data (e.g., Power BI, Tableau).

Data Flow:

- Data Ingestion**: Raw data is ingested from various sources (e.g., sensors, databases) into ADLS.

2. **Data Access:** The PySpark notebook in the Azure Databricks cluster reads data from ADLS using PySparkSQL functions.
3. **Data Exploration and Transformation:** The notebook performs various operations on the data:
 - **EDA:** Analyze data characteristics using PySparkSQL functions (e.g., describe, groupBy).
 - **Cleaning:** Address missing values, outliers, and inconsistencies.
 - **Transformation:** Filter, aggregate, join datasets, and perform other manipulations.
 - **Optimization:** Partition data, cache frequently used datasets, define UDFs (if applicable).
4. **Analysis and Visualization:** The processed data is analyzed within the notebook. Optionally, data visualization tools can be used to create charts and graphs.
5. **Reporting (Optional):** The insights and visualizations can be exported to reporting tools for further analysis and communication.

How it works:

Code snippet

```
graph LR
A[Azure Data Lake Storage (ADLS)] --> B{Raw Data}
B --> C{Azure Databricks Cluster}
C --> D{PySpark Notebook}
D --> E{Data Exploration & Transformation}
D --> F{Data Visualization (Optional)}
E --> G{Analysis & Insights}
G --> H{Reporting (Optional)}
```

Data Flow:

1. The user submits a request to explore and optimize data.
2. The Azure Databricks Workspace manages and allocates resources (cluster) for the project.
3. The PySpark notebook in the cluster reads data from ADLS.
4. The notebook performs data exploration, transformation, and optimization.
5. Analysis and insights are generated from the processed data.
6. Optionally, the data can be exported for visualization.
7. Optionally, the insights can be exported to a reporting tool for further analysis and communication.

This block diagram focuses on the high-level interaction between components and the data flow. It provides a clearer picture of the user interaction and the overall processing pipeline.

Execution Overview:

1. Setup:

- **Provision Azure Databricks resources:**
 - Create an Azure Databricks workspace.
 - Launch a cluster with appropriate configurations for your data size and processing needs.
- **Mount ADLS:**
 - Use dbutils library to mount your ADLS account to the cluster.
- **Prepare PySpark notebook:**
 - Create a new notebook in your workspace.
 - Import necessary libraries (e.g., pyspark.sql).
 - Define connection details for ADLS.

2. Data Exploration:

- **Read data from ADLS:**

- Use functions like spark.read.parquet or spark.read.csv based on data format.
- **Analyze schema:**
 - Utilize df.printSchema() to understand data structure, including column names and data types.
- **Perform exploratory data analysis (EDA):**
 - Utilize functions like df.describe(), df.groupBy(...).count(), etc., to gain insights into data distribution, missing values, and summary statistics.

3. Data Transformation and Optimization:

- **Data cleaning:**
 - Address missing values with df.fillna(value) or impute them using appropriate methods.
 - Handle outliers or inconsistencies as needed.
- **Data transformation:**
 - Filter specific data subsets using df.filter(condition).
 - Aggregate data with functions like df.groupBy(...).agg(avg("numeric_column")).
 - Join datasets from different files using df.join(df2, on="column_name", how="inner").
- **Data optimization:**
 - Partition data based on specific columns using df.repartition(n, col1, col2).
 - Cache frequently accessed data frames using df.cache().
 - Consider cost-based optimization using EXPLAIN and indexing techniques for large datasets.

4. Analysis and Reporting:

- **Analyze data:**
 - Utilize the processed data frame for further analysis and visualization.
- **Visualize insights (Optional):**
 - Integrate libraries like Matplotlib or Plotly to create visualizations.

- **Generate reports (Optional):**
 - Export data or visualizations to formats like CSV, parquet, or notebooks for sharing or further analysis.

5. Cleanup:

- **Terminate Azure Databricks cluster:**
 - Release resources when processing is complete.
- **Clear notebook variables (Optional):**
 - Ensure proper memory management within the notebook.

Azure Resources Used for this Project:

Here are the Azure resources used for the data exploration and optimization project in Azure Databricks:

Essential Resources:

- **Azure Databricks Workspace:** Serves as the central platform for creating and managing your Databricks projects, including notebooks, clusters, and data access.
- **Azure Databricks Cluster:** Provides a scalable compute environment with Apache Spark installed, where your PySpark notebooks execute the data exploration and optimization operations.
- **Azure Data Lake Storage (ADLS):** Serves as the data source, storing your raw data in various formats like parquet, CSV, etc.

Optional Resources:

- **Azure Synapse Analytics:** Can be used for data warehousing and potentially integrated with Azure Databricks for advanced analytics and data management. (Optional for this specific project)
- **Azure Data Factory:** Can be used for data orchestration and automating data pipelines, potentially moving data from various sources to ADLS before exploration. (Optional for this specific project)
- **Azure Key Vault:** Can be used to securely manage secrets and credentials used within your PySpark notebook for accessing ADLS or other resources. (Optional for this specific project)

Project Requirements:

The requirements for this project are broken down into six different parts which are

1. Data Access:

- **Format:** Specify the format of the data stored in ADLS (e.g., parquet, CSV, JSON). Knowing the format helps determine the appropriate reading method in PySparkSQL.
- **Location:** Provide the path to the data location within your ADLS Gen2 storage account. This includes the container name and folder path.
- **Credentials:** Outline the access method for ADLS. You might need to configure service principals or use managed identities to grant your Azure Databricks cluster access to your ADLS data.

2. Data Exploration:

- **Specific Tasks:** List specific exploratory analysis tasks you want to achieve. This may include:
 - **Descriptive statistics:** Analyzing data distribution like mean, standard deviation, and count for numerical columns.
 - **Data quality checks:** Identifying missing values, outliers, and inconsistencies within the data.
 - **Data understanding:** Analyzing data schema, column names, and data types to understand the data structure.
- **Visualization:** Define if you need to create visualizations (e.g., histograms, scatter plots) during exploration to gain visual insights.

3. Data Transformation:

- **Filtering:** Define specific conditions for filtering data subsets based on specific column values or conditions.
- **Aggregation:** Specify the desired aggregation operations like calculating averages, sums, or counts across groups of data.

- **Joining:** If applicable, define the join conditions and types (e.g., inner join, left join) for combining data from multiple datasets or tables within ADLS.
- **Other Transformations:** Specify any additional transformations needed, such as renaming columns, handling null values, or converting data types.

4. Data Optimization:

- **Partitioning:** Outline the columns for partitioning the data. This improves query performance by allowing Spark to efficiently access relevant data subsets.
- **Caching:** Identify frequently used data frames or transformations that can benefit from caching in memory for faster access during subsequent operations.
- **Cost-based Optimization:** If dealing with large datasets, consider using EXPLAIN to analyze query execution plans and identify potential optimizations like indexing relevant columns.

5. Analysis and Reporting:

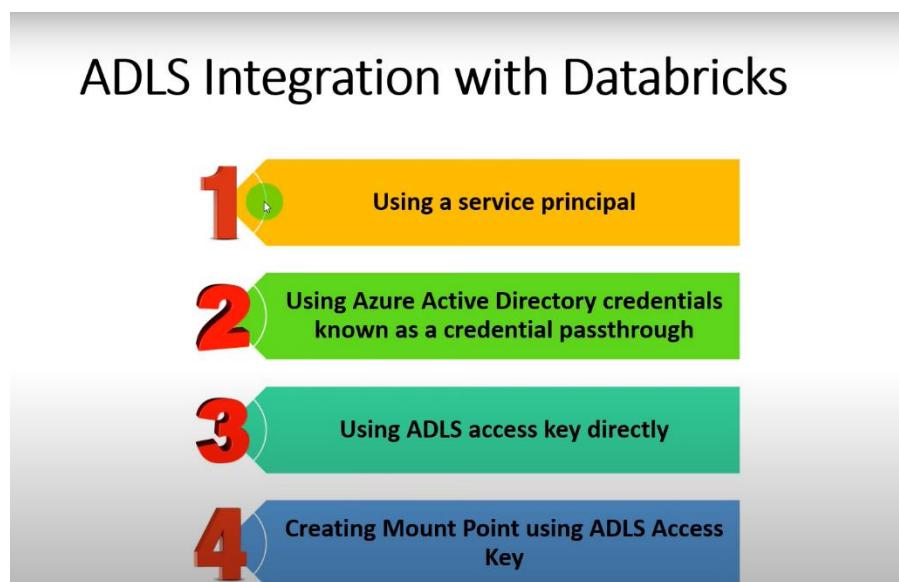
- **Insights:** Define the specific insights you want to extract from the data. This might involve analyzing trends, relationships between variables, or identifying patterns within the data.
- **Reporting:** Specify how you want to report the findings. This could include generating visualizations using libraries like Matplotlib or Plotly, creating reports in formats like CSV or PDF, or integrating the results with other reporting tools.

6. Non-Functional Requirements:

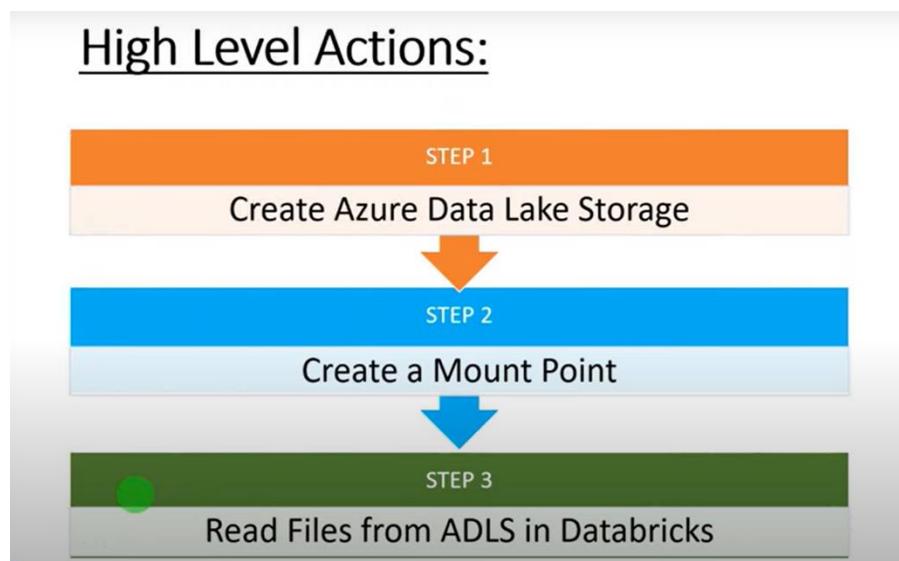
- **Performance:** Specify any performance expectations for data processing and query execution times.
- **Scalability:** Define if the project needs to handle growing data volumes in the future, requiring adjustments to cluster size or optimization techniques.

- **Security:** Outline any security requirements related to access controls, data encryption, or audit logging for both your data in ADLS and your processing environment in Azure Databricks.

Four Methods to Integrate ADLS with Databricks



In this Project we are using ADLS Access Key directly to connect with Azure Databricks



Tasks performed:

1. Data Exploration:

Login in to Azure Portal and create a Storage Account

Name “hexaprojectadls”

The screenshot shows the 'Create a storage account' wizard on the Microsoft Azure portal. The 'Project details' section is visible, showing the selected subscription ('Azure subscription 1'), resource group ('rg-azuser1080_mml.local-6M3n1'), storage account name ('hexaprojectadls'), region ('(US) East US 2'), and performance level ('Standard'). The 'Performance' section also includes an option for 'Premium'. At the bottom, there are 'Previous', 'Next', and 'Review + create' buttons.

Enabling Hierarchical Namespace

The screenshot shows the 'Configure security settings' section of the storage account creation wizard. Under the 'Hierarchical Namespace' heading, the 'Enable hierarchical namespace' checkbox is checked. Other sections like 'Access protocols' are partially visible. The bottom of the screen shows the standard Windows taskbar with various pinned icons.

Review & and Create Storage Account

Basics

Subscription	Azure subscription 1
Resource group	rg-azuser1080_mml.local-6M3nl
Location	East US 2
Storage account name	hexaprojectadls
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

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

Previous Next Create Give feedback

Storage Account Created

Deployment succeeded

Deployment 'hexaprojectadls_1708764989020' to resource group 'rg-azuser1080_mml.local-6M3nl' was successful.

Go to resource Pin to dashboard

Deployment details

Start time: 2/24/2024, 2:26:42 PM
Correlation ID: 2918111-81d3-4acd-a781-6208a9b19612

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 >

32°C Haze

Click Go to Resource

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes links like 'The Cool Connectio...', 'Marketing Simulatio...', 'New Product Develo...', 'How does a Hair Dr...', 'Future Of Logistics...', 'TAPMI Library', 'Take Home Salary C...', 'Job Search Welcome', and 'Candidate Progress'. The main title bar says 'hexaprojectadls - Microsoft Azure'. The left sidebar has sections for 'Overview', 'Activity log', 'Tags', 'Diagnose and solve problems', 'Access Control (IAM)', 'Data migration', 'Events', 'Storage browser', 'Data storage' (with 'Containers', 'File shares', 'Queues', 'Tables'), 'Security + networking' (with 'Networking' and 'Access keys'), and a weather widget showing '32°C Haze'. The main content area is titled 'hexaprojectadls | Overview'. It displays details about the resource group ('rg-azuser1080_mmlocal-6M3nl'), location ('eastus2'), primary/secondary location ('Primary: East US 2, Secondary: Central US'), subscription ('Azure subscription 1'), subscription ID ('984f097c-963c-4eb6-a20d-839457ae9f08'), disk state ('Primary: Available, Secondary: Available'), and tags ('Add tags'). Below this are tabs for 'Properties', 'Monitoring', 'Capabilities (5)', 'Recommendations (0)', 'Tutorials', and 'Tools + SDKs'. Under 'Properties', there are sections for 'Data Lake Storage' (Hierarchical namespace: Enabled, Default access tier: Hot, Blob anonymous access: Disabled, Blob soft delete: Enabled (7 days), Container soft delete: Enabled (7 days), Versioning: Disabled) and 'Security' (Require secure transfer for REST API operations: Enabled, Storage account key access: Enabled, Minimum TLS version: Version 1.2, Infrastructure encryption: Disabled). There is also a 'Networking' section. The bottom right corner shows system status: ENG IN, 24-02-2024, 14:27.

Create Container

Name adls-container

The screenshot shows the Microsoft Azure portal interface, similar to the previous one but with a different focus. The top navigation bar and sidebar are identical. The main content area is titled 'hexaprojectadls | Containers'. It shows a list of existing containers: '\$logs'. On the right, a modal window titled 'New container' is open. It has fields for 'Name' (set to 'adls-container') and 'Anonymous access level' (set to 'Private (no anonymous access)'). A note below the field says: 'The access level is set to private because anonymous access is disabled on this storage account.' At the bottom of the modal are 'Create' and 'Give feedback' buttons. The bottom right corner shows system status: ENG IN, 24-02-2024, 14:27.

Container Created

A screenshot of a Microsoft Edge browser window showing the Azure Storage Container list. The URL is https://portal.azure.com/#@ihtl.onmicrosoft.com/resource/subscriptions/984f097c-963c-4eb6-a20d-839457ae9f08/resourcegroups/g-azuser1080_mmml.local-6M3nl/providers/Microsoft.... The page title is "hexaprojectadls - Microsoft Azure". The left sidebar shows "hexaprojectadls | Containers" under "Storage account". The main content area shows a table of containers:

Name	Last modified	Anonymous access level	Lease state
\$logs	2/24/2024, 2:27:07 PM	Private	Available
adls-container	2/24/2024, 2:28:02 PM	Private	Available

A success message box is visible in the top right corner: "Successfully created storage container" and "Successfully created storage container 'adls-container'". The system tray at the bottom shows the date as 24-02-2024.

Upload a CSV file in the Container

A screenshot of a Microsoft Edge browser window showing the Azure Storage Blob list for the "adls-container". The URL is https://portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/-/overview/storageAccountId/%2Fsubscriptions%2F984f097c-963c-4eb6-a20d-839457ae9f08%2Fresource.... The page title is "adls-container - Microsoft Azure". The left sidebar shows "adls-container" under "Container". The main content area shows a table of blobs:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
No results						

The system tray at the bottom shows the date as 24-02-2024.

The screenshot shows the Microsoft Azure Storage Container Overview page for the 'adls-container' in the 'hexaprojectadls_1708764989020' account. The left sidebar includes options like Overview, Diagnose and solve problems, Access Control (IAM), Settings (Shared access tokens, Manage ACL, Access policy, Properties, Metadata), and a search bar. The main area displays a table with columns Name, Modified, Access tier, and Archive status. A modal window titled 'Upload blob' is open, showing a cloud icon and a message indicating 1 file(s) selected: 'annual-enterprise-survey-2021-financial-year-provisio...'. It also includes a checkbox for 'Overwrite if files already exist' and a 'Upload' button. The taskbar at the bottom shows various pinned icons and the date/time as 24-02-2024.

File Uploaded

The screenshot shows the Microsoft Azure Storage Container Overview page for the 'adls-container' in the 'hexaprojectadls_1708764989020' account. The interface is identical to the previous screenshot, but a success message box is displayed in the top right corner stating 'Successfully uploaded blob(s)' and 'Successfully uploaded 1 blob(s.)'. The table in the main area now lists the uploaded file: 'annual-enterprise-survey-2021-financial-year-provisio...' with a modified date of '2/24/2024, 2:29:21 PM', an access tier of 'Hot (Inferred)', and an archive status of 'Not yet archived'. The taskbar at the bottom shows various pinned icons and the date/time as 24-02-2024.

Now go to Storage Account get Access key

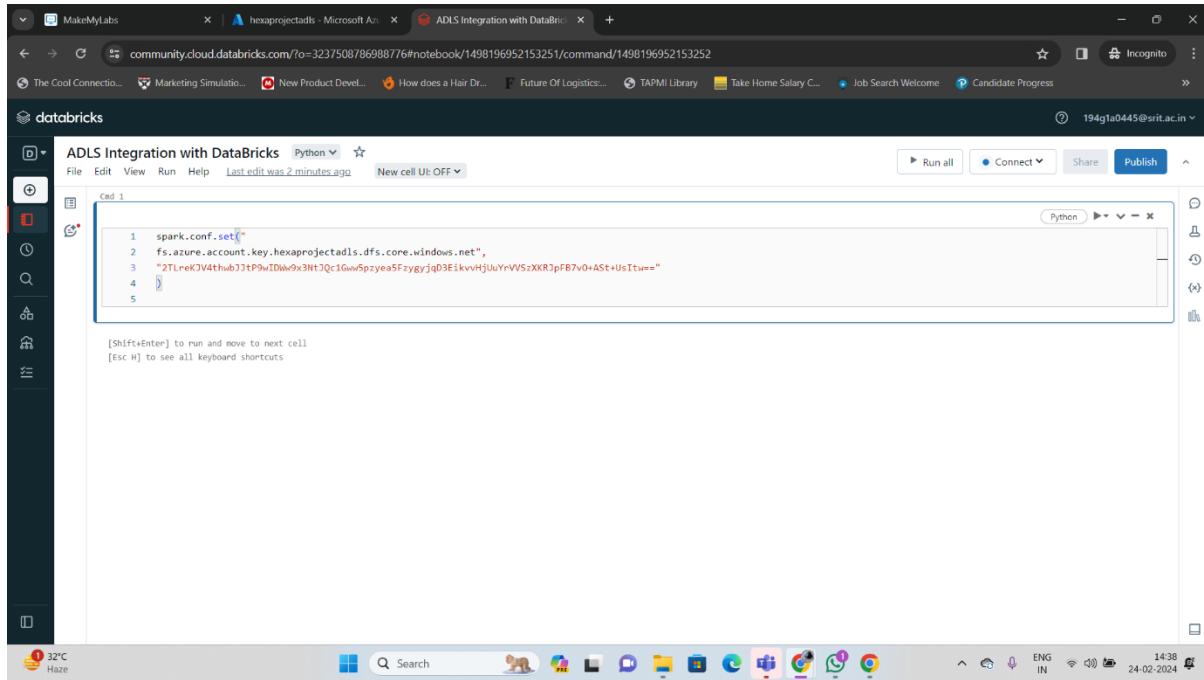
The screenshot shows the Microsoft Azure portal interface. The left sidebar is titled 'hexaprojectadls' and includes sections for Data storage (Containers, File shares, Queues, Tables), Security + networking (Networking, Access keys, Shared access signature, Encryption, Microsoft Defender for Cloud), and Data management (Storage tasks (preview)). The main content area is titled 'Containers' and lists two containers: 'Slogs' and 'adls-container'. Each container entry includes columns for Name, Last modified, Anonymous access level, and Lease state. A search bar at the top is set to 'Search resources, services, and docs (G+)'. The bottom status bar shows the URL as https://portal.azure.com/#@ihtl.onmicrosoft.com/resource/subscriptions/984f097c-963c-4eb6-a20d-839457ae9f08/resourceGroups/rg-azuser1080_mmilocal-6M3n/providers/Microsoft.Storage/storageAccounts/hexaprojectadls/keys.

Copy Access Key

The screenshot shows the Microsoft Azure portal interface. The left sidebar is identical to the previous screenshot. The main content area is titled 'Access keys' and displays information about the 'hexaprojectadls' storage account. It includes a note about the importance of keeping keys secure and updating them. Two access keys are listed: 'key1' and 'key2'. Each key has a 'Rotate key' button, a timestamp (Last rotated: 2/24/2024 (0 days ago)), and a 'Copy to clipboard' button. Below each key is a 'Connection string' field with a 'Show' button. The bottom status bar shows the URL as https://portal.azure.com/#@ihtl.onmicrosoft.com/resource/subscriptions/984f097c-963c-4eb6-a20d-839457ae9f08/resourceGroups/rg-azuser1080_mmilocal-6M3n/providers/Microsoft.Storage/storageAccounts/hexaprojectadls/keys.

Go to Databricks Portal

Run PySpark command by giving Storage account name and access key

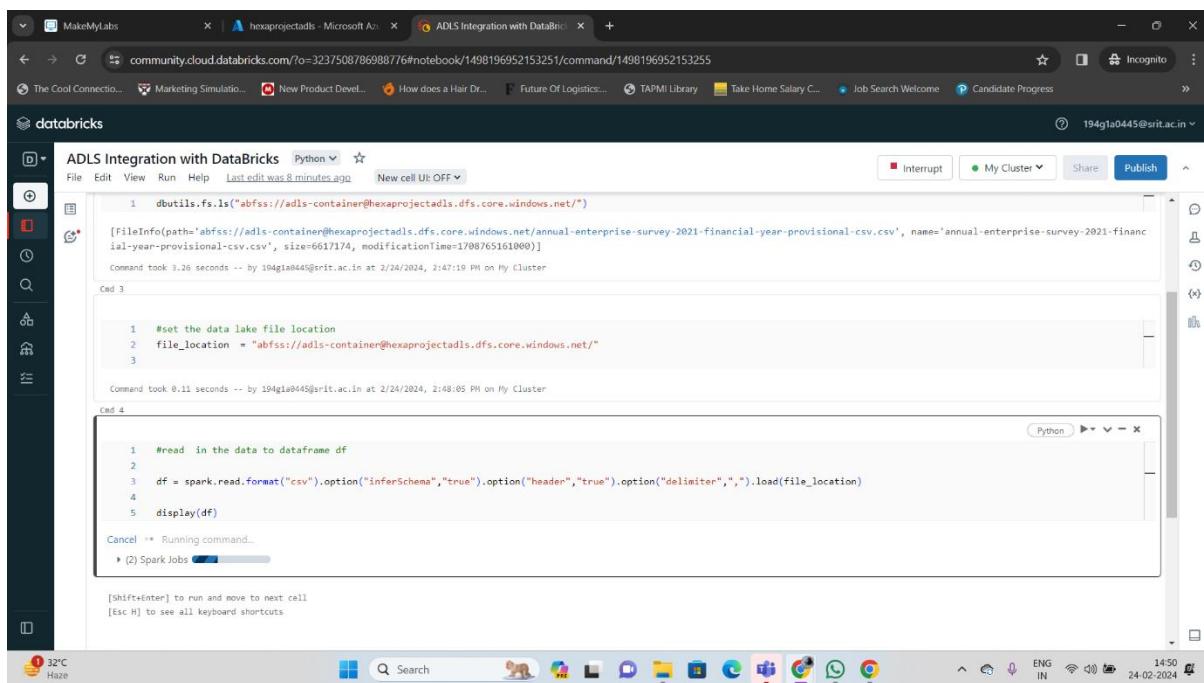


The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** MakeMyLabs, hexaprojectadls - Microsoft Az, ADLS Integration with DataBri...
- Address Bar:** community.cloud.databricks.com/?o=3237508786988776#notebook/1498196952153251/command/1498196952153252
- Header:** The Cool Connectio..., Marketing Simulatio..., New Product Devel..., How does a Hair Dr..., Future Of Logistics..., TAPMI Library, Take Home Salary C..., Job Search Welcome, Candidate Progress
- User Info:** 194gta0445@srit.ac.in
- Content Area:** A Databricks notebook titled "ADLS Integration with DataBrics". It contains a single code cell labeled "Cmd 1" with the following Python code:

```
1 spark.conf.set("fs.azure.account.key.hexaprojectadls.dfs.core.windows.net",  
2 "2TlreKjV4thubJjtP9w7Dwv9x3NTQc1Gww5pzyea5FzgyjqpD3EikvvHjUvYrVVSzXKR7pFB7v0+AST+UzTt=="  
3 )  
4  
5
```

Instructions below the code cell: [Shift+Enter] to run and move to next cell [Esc H] to see all keyboard shortcuts
- Bottom Bar:** ENG IN, 24-02-2024, 14:38



The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** MakeMyLabs, hexaprojectadls - Microsoft Az, ADLS Integration with DataBri...
- Address Bar:** community.cloud.databricks.com/?o=3237508786988776#notebook/1498196952153251/command/1498196952153255
- Header:** The Cool Connectio..., Marketing Simulatio..., New Product Devel..., How does a Hair Dr..., Future Of Logistics..., TAPMI Library, Take Home Salary C..., Job Search Welcome, Candidate Progress
- User Info:** 194gta0445@srit.ac.in
- Content Area:** A Databricks notebook titled "ADLS Integration with DataBrics". It contains several code cells:
 - Cell 1: `dbutils.fs.ls("abfss://adls-container@hexaprojectadls.dfs.core.windows.net/")`. Output: [FileInfo(path='abfss://adls-container@hexaprojectadls.dfs.core.windows.net/annual-enterprise-survey-2021-financial-year-provisional-csv.csv', size=6617174, modificationTime='1798865161000')]
 - Cell 2: `#set the data lake file location`, `file_location = "abfss://adls-container@hexaprojectadls.dfs.core.windows.net/"`. Output: Command took 0.11 seconds -- by 194gta0445@srit.ac.in at 2/24/2024, 2:47:19 PM on My Cluster
 - Cell 3: `#read in the data to dataframe df`, `df = spark.read.format("csv").option("inferSchema", "true").option("header", "true").option("delimiter", ",").load(file_location)`, `display(df)`. Output: (2) Spark Jobs
- Bottom Bar:** ENG IN, 24-02-2024, 14:50

To read data to Dataframe [Df] & Display

The screenshot shows a Databricks notebook titled "ADLS Integration with DataBrics". The code cell contains the following Python code:

```
1 #read in the data to dataframe df
2
3 df = spark.read.format("csv").option("inferSchema","true").option("header","true").option("delimiter",",").load(file_location)
4
5 display(df)
```

The output of the code shows a table with 10 rows and 7 columns. The columns are: Year, Industry_aggregation_NZSIOC, Industry_code_NZSIOC, Industry_name_NZSIOC, Units, Variable_code, and Variable.

Year	Industry_aggregation_NZSIOC	Industry_code_NZSIOC	Industry_name_NZSIOC	Units	Variable_code	Variable
1 2021	Level 1	99999	All industries	Dollars (millions)	H01	Total inc...
2 2021	Level 1	99999	All industries	Dollars (millions)	H04	Sales, go...
3 2021	Level 1	99999	All industries	Dollars (millions)	H05	Interest, r...
4 2021	Level 1	99999	All industries	Dollars (millions)	H07	Non-ope...
5 2021	Level 1	99999	All industries	Dollars (millions)	H08	Total exp...
6 2021	Level 1	99999	All industries	Dollars (millions)	H09	Interest e...
7 2021	Level 1	99999	All industries	Dollars (millions)	H10	Indirect...
8 2021	Level 1	99999	All industries	Dollars (millions)		
9 2021	Level 1	99999	All industries	Dollars (millions)		
10 2021	Level 1	99999	All industries	Dollars (millions)		

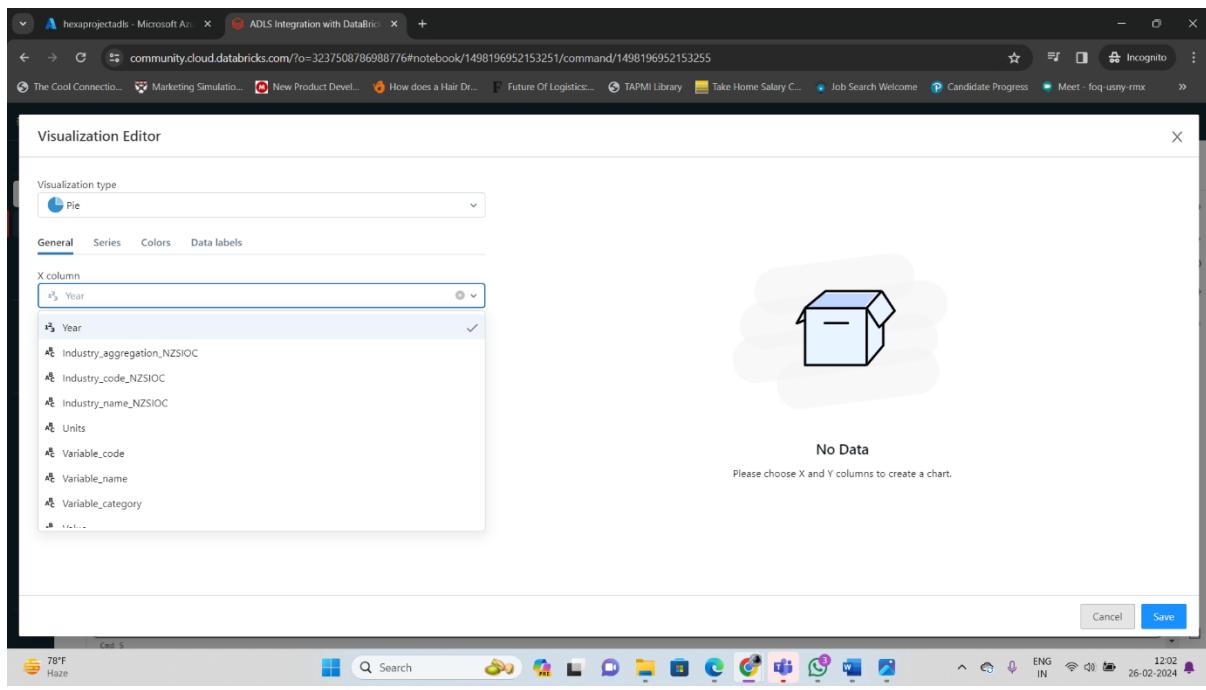
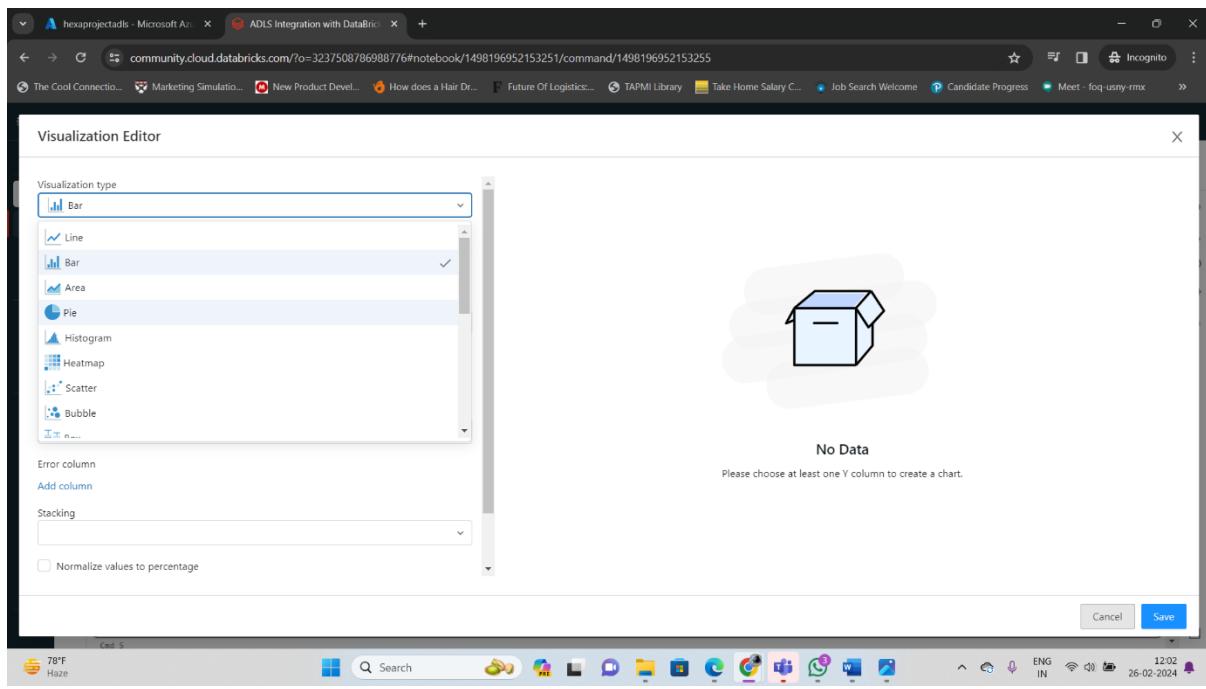
Perform Visualization in order to Explore Data

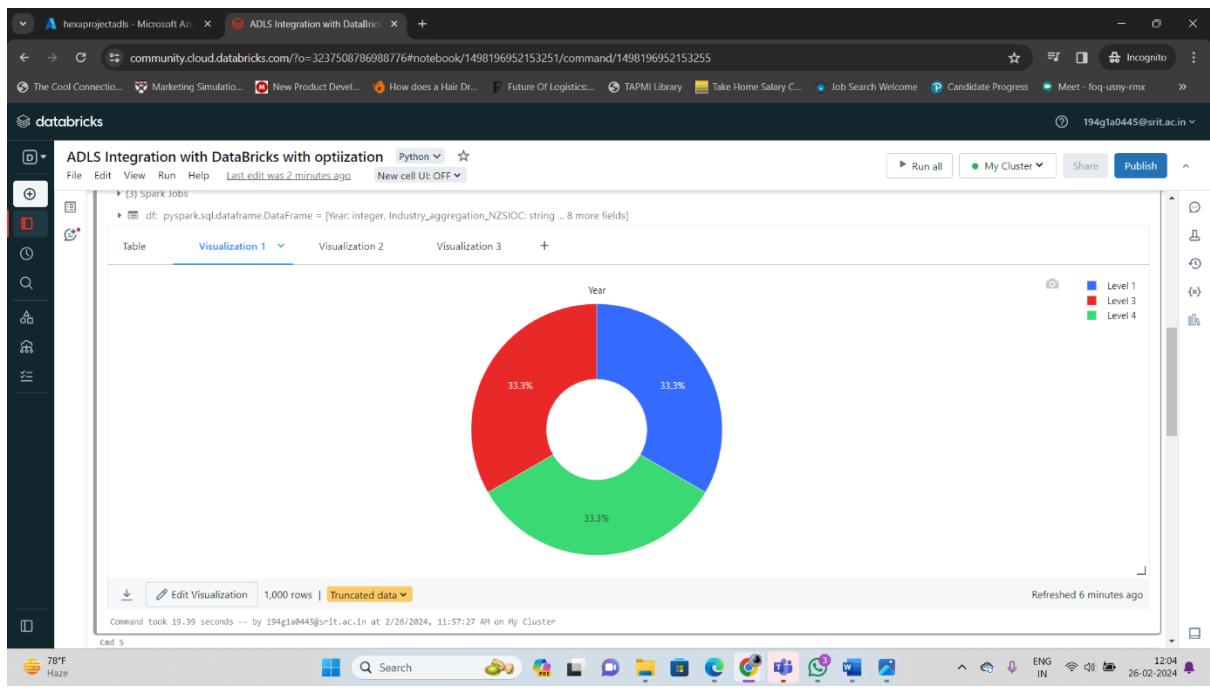
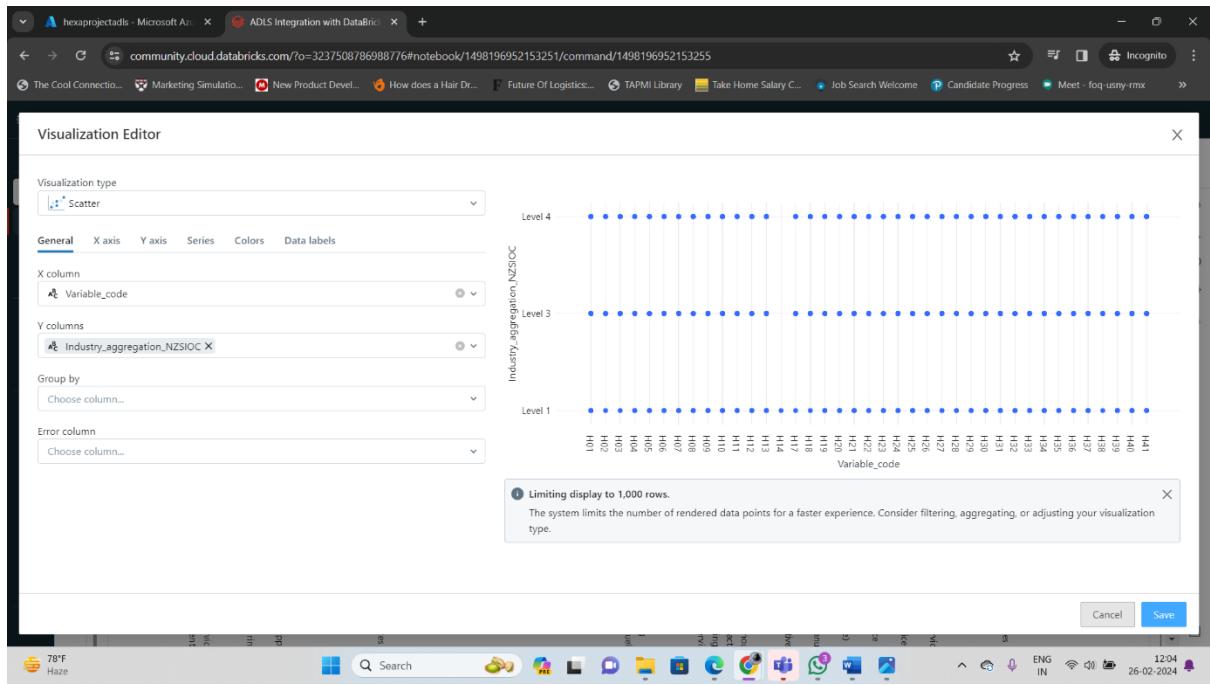
The screenshot shows a Databricks notebook titled "ADLS Integration with DataBrics with optimization". The code cell contains the same Python code as the previous screenshot:

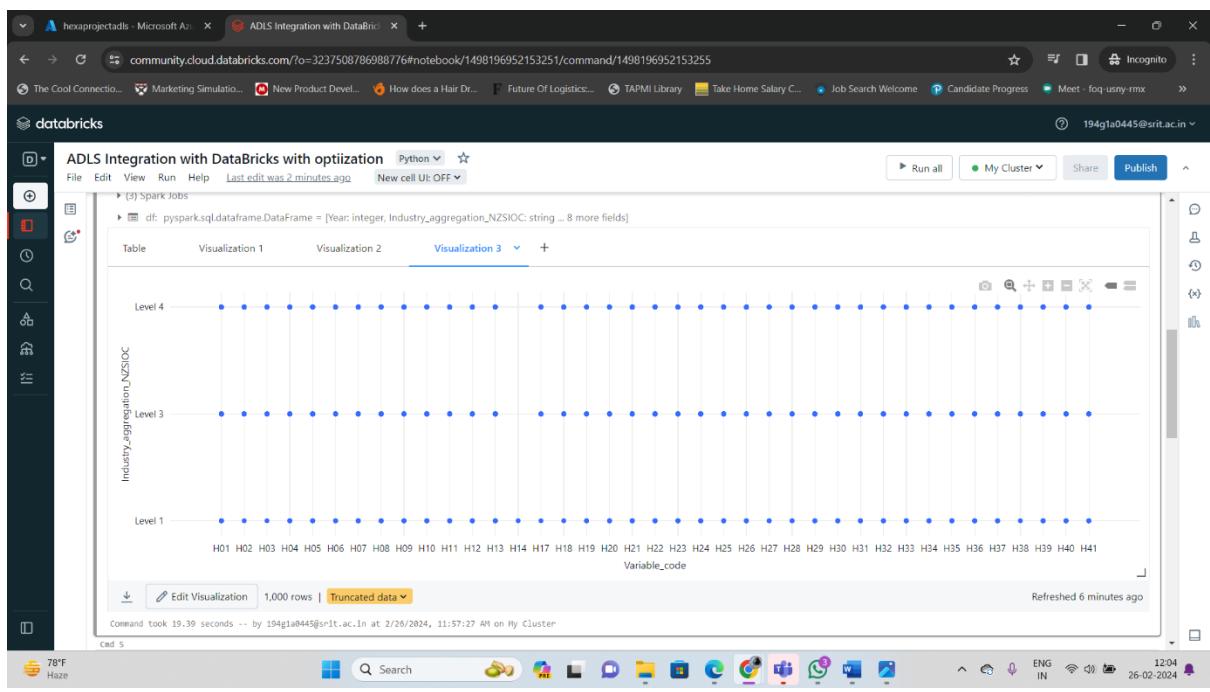
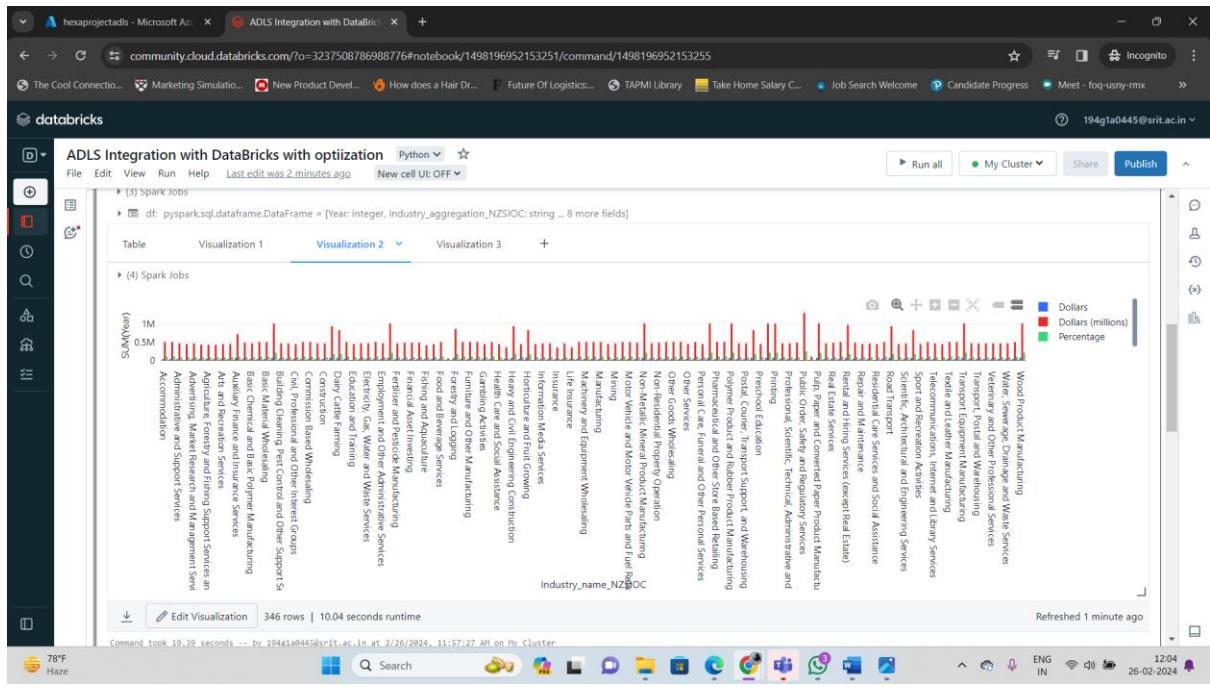
```
1 #read in the data to dataframe df
2
3 df = spark.read.format("csv").option("inferSchema","true").option("header","true").option("delimiter",",").load(file_location)
4
5 display(df)
```

The output of the code shows a table with 10 rows and 7 columns. The columns are: Year, Industry_aggregation_NZSIOC, Industry_code_NZSIOC, Industry_name_NZSIOC, Units, Variable_code, and Variable. A tooltip "Visualization" is shown over the first row of the table.

Year	Industry_aggregation_NZSIOC	Industry_code_NZSIOC	Industry_name_NZSIOC	Units	Variable_code	Variable
1 2021	Data Profile	99999	All industries	Dollars (millions)	H01	Total inc...
2 2021	Legacy Visualization	99999	All industries	Dollars (millions)	H04	Sales, go...
3 2021	Level 1	99999	All industries	Dollars (millions)	H05	Interest, r...
4 2021	Level 1	99999	All industries	Dollars (millions)	H07	Non-ope...
5 2021	Level 1	99999	All industries	Dollars (millions)	H08	Total exp...
6 2021	Level 1	99999	All industries	Dollars (millions)	H09	Interest e...
7 2021	Level 1	99999	All industries	Dollars (millions)	H10	Indirect...
8 2021	Level 1	99999	All industries	Dollars (millions)		
9 2021	Level 1	99999	All industries	Dollars (millions)		
10 2021	Level 1	99999	All industries	Dollars (millions)		







2. Data Optimization:

For the same set of we perform Optimization

Key Highlights:

- 1) Delta table Optimization
 - Partitioning
- 2) File Management Optimization
 - Compacting
 - Z-ordering
- 3) Data Skipping
- 4) Caching

➤ Partitioning

Partitioning is a crucial technique for optimizing data lake performance and managing large datasets effectively. Here's a breakdown of its role and considerations:

What is data lake partitioning?

Data lake partitioning involves dividing your data into smaller, manageable segments based on specific criteria called **partition keys**. These keys could be any relevant attribute like:

- **Time:** Year, month, day, hour, etc. (e.g., data partitioned by year)
- **Location:** Country, state, city, etc. (e.g., data partitioned by country)
- **Category:** Product type, user type, etc. (e.g., data partitioned by product type)

Benefits of partitioning:

- **Faster query performance:** When querying the data lake, the query engine only needs to scan the relevant partitions based on the query filters. This significantly reduces the amount of data scanned, leading to faster query execution times.
- **Improved scalability:** As your data volume grows, partitioning allows you to add new data efficiently by creating new partitions. This makes scaling your data lake more manageable.
- **Simplified data management:** Partitioning simplifies tasks like data deletion, archiving, and backup, as you can focus on specific partitions based on their keys.

When is partitioning not recommended?

- **Small data lakes:** For datasets smaller than a terabyte, the overhead of managing partitions might outweigh the performance benefits.
- **Unpredictable access patterns:** If your access patterns are unpredictable and don't follow a specific key, partitioning might not be as effective.

➤ **File Management Optimization:**

- **Compacting:** This process combines smaller data files into larger ones, which can improve performance by:
 - **Reducing the number of files:** Fewer files mean fewer directory entries to scan and less metadata overhead.
 - **Improving sequential reads:** Larger files enable faster data reads by minimizing seek times on storage devices.
- **Z-ordering:** This technique physically re-orders data files based on a chosen attribute (Z-key), ensuring data related to specific queries becomes physically closer on storage. This can

significantly speed up queries that frequently access specific values of the Z-key.

➤ **Data Skipping:**

This optimization helps avoid processing irrelevant data during analytics. It involves techniques like predicate pushdown, which allows the query engine to filter data on the source side (e.g., within the storage layer) before transferring it for processing. This reduces network transfer and processing overhead.

➤ **Caching:**

Caching stores frequently accessed data in a readily available location, like memory or a dedicated caching layer. This significantly reduces the time to retrieve data for subsequent queries that access the same information, leading to faster query responses.

These techniques, along with partitioning, work together to create a well-optimized data lake that facilitates efficient data management and faster analytics.

Here's a summary table for easier reference:

Technique	Description	Benefits
Partitioning	Dividing data based on specific criteria	Faster queries, improved scalability, simplified data management
Compacting	Combining smaller files into larger ones	Reduced overhead, faster sequential reads
Z-ordering	Physically re-ordering files based on a chosen attribute	Faster queries accessing specific values of the Z-key
Data Skipping	Filtering data at the source based on query filters	Reduced network transfer and processing overhead
Caching	Storing frequently accessed data in a readily available location	Faster query response times

In this project we have used Partitioning

We are doing Partitioning by column with name
“Industry_code_NZSIOC”

The screenshot shows a Databricks notebook titled "ADLS Integration with DataBrics with optimization". It contains two code cells and a table view.

Code Cell 5:

```
1 df.write.option("header",True).partitionBy("Industry_code_NZSIOC").mode("overwite").csv("dbfs:/default.annual_enterprise_survey_2021_financial_year_provisional_csv_csv")
```

Code Cell 6:

```
1 df1 = spark.read.format("csv").option("header",True).option("sep",",").load("dbfs:/default.annual_enterprise_survey_2021_financial_year_provisional_csv_csv")
```

Table View:

Year	Industry_aggregation_NZSIC	Industry_name_NZSIOC	Units	Variable_code	Variable_name
1 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H01	Total income
2 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H02	Sales of goods not further
3 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H03	Sales of other goods and s

The Column has been removed and we got Optimize Data

The screenshot shows a Databricks notebook titled "ADLS Integration with DataBrics with optimization". It contains two code cells and a table view.

Code Cell 5:

```
1 df1 = spark.read.format("csv").option("header",True).option("sep",",").load("dbfs:/default.annual_enterprise_survey_2021_financial_year_provisional_csv_csv")
```

Code Cell 6:

```
1 df1: pyspark.sql.dataframe.DataFrame = [Year: string, Industry_aggregation_NZSIOC: string ... 8 more fields]
```

Table View:

Year	Industry_aggregation_NZSIC	Industry_name_NZSIOC	Units	Variable_code	Variable_name
1 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H01	Total income
2 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H02	Sales of goods not further
3 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H03	Sales of other goods and s
4 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H05	Interest, dividends and do
5 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H06	Government funding, gran
6 2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H07	Non-operating income
2021	Level 3	Other Store-Based Retailing and Non Store Retailing	Dollars (millions)	H08	Total expenditure

Spark (Only PySpark and SQL)

- Spark architecture, Data Sources API, and Dataframe API.
- PySpark - Ingested CSV, simple, and complex JSON files into the data lake as parquet files/ tables.
- PySpark - Transformations such as Filter, Join, Simple Aggregations, GroupBy, Window functions etc.
- PySpark - Created global and temporary views.
- Spark SQL - Created databases, tables, and views.
- Spark SQL - Transformations such as Filter, Join, Simple Aggregations, GroupBy, Window functions etc.
- Spark SQL - Created local and temporary views.
- Implemented full refresh and incremental load patterns using partitions.

Delta Lake

- Performed Read, Write, Update, Delete, and Merge to delta lake using both PySpark as well as SQL.
- History, Time Travel, and Vacuum.
- Converted Parquet files to Delta files.
- Implemented incremental load pattern using delta lake.

Azure Data Factory

- Created pipelines to execute Databricks notebooks.
- Designed robust pipelines to deal with unexpected scenarios such as missing files.
- Created dependencies between activities as well as pipelines.
- Scheduled the pipelines using data factory triggers to execute at regular intervals.
- Monitored the triggers/ pipelines to check for errors/ outputs.

Technologies/Tools Used:

- Pyspark
- Spark SQL
- Delta Lake
- Azure Databricks
- Azure Data Factory
- Azure Date Lake Storage Gen2
- Azure Key vault
- Power BI(Optional)

REFERENCE LINKS:

<https://learn.microsoft.com/en-us/azure/architecture/example-scenario/data/synapse-exploratory-data-analytics>

<https://learn.microsoft.com/en-us/azure/databricks/delta/optimize>