IPL Data Ingestion and Analytics Using Azure Data Factory and Databricks

A Comprehensive Guide

# Introduction

In this guide, we will explore the process of ingesting Indian Premier League (IPL) data through Azure Data Factory, Databricks performing analytics, creating external and managed tables, querying the managed table, performing transformations, and finally storing the data as Parquet files in a ]delta lake table. Azure Data Factory is a powerful cloud-based data integration service that allows you to create data-driven workflows for orchestrating and automating data movement and data transformation through databricks platform.

# Step 1: Data Ingestion with Azure Data Factory

Azure Data Factory (ADF) is a versatile tool for ingesting data from various sources. For this scenario, we will focus on IPL data.

## Creating a Data Pipeline

1. \*\*Create an Azure Data Factory instance:\*\* Start by creating a new Data Factory instance in the Azure portal.

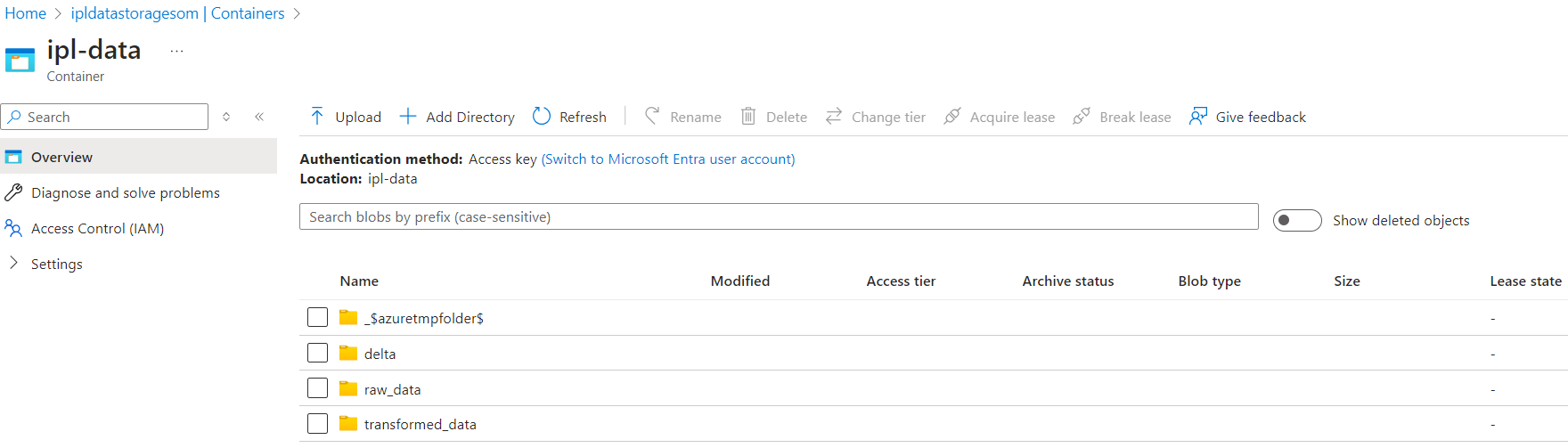
2. \*\*Set up Linked Services:\*\* Linked Services define the connection information needed for ADF to connect to external data sources. Create Linked Services for the IPL data source (e.g., Azure Blob Storage) and the destination (e.g., Azure SQL Database).

3. \*\*Create Datasets:\*\* Datasets represent the data structures within the data stores that ADF interacts with. Create datasets for the IPL data in your source and destination linked services.

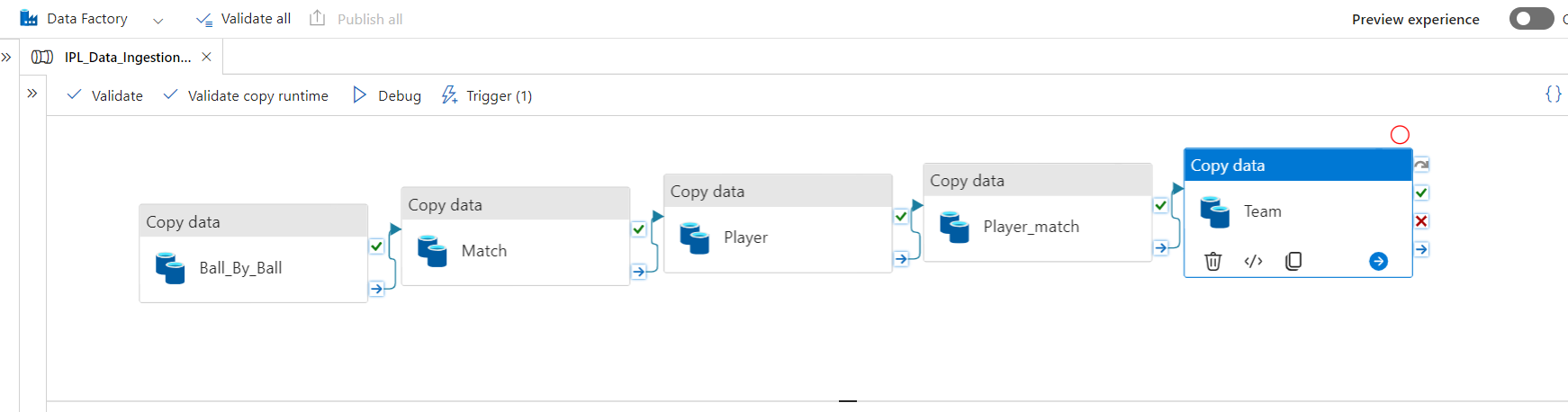
4. \*\*Design the Pipeline:\*\* Create a pipeline in ADF to define the series of steps required to ingest data. Use activities such as Copy Data to move IPL data from the source to the destination.

Sharing the SS below:

**Creating blob data storage with data lake feature enabled for raw\_data, transformed\_data.**



**Performing IPL raw data ingestion from Http sites to raw data folder under ipl-data container in data lake(Blob store with hierarchical name space) through ADF Ingestion pipeline.**

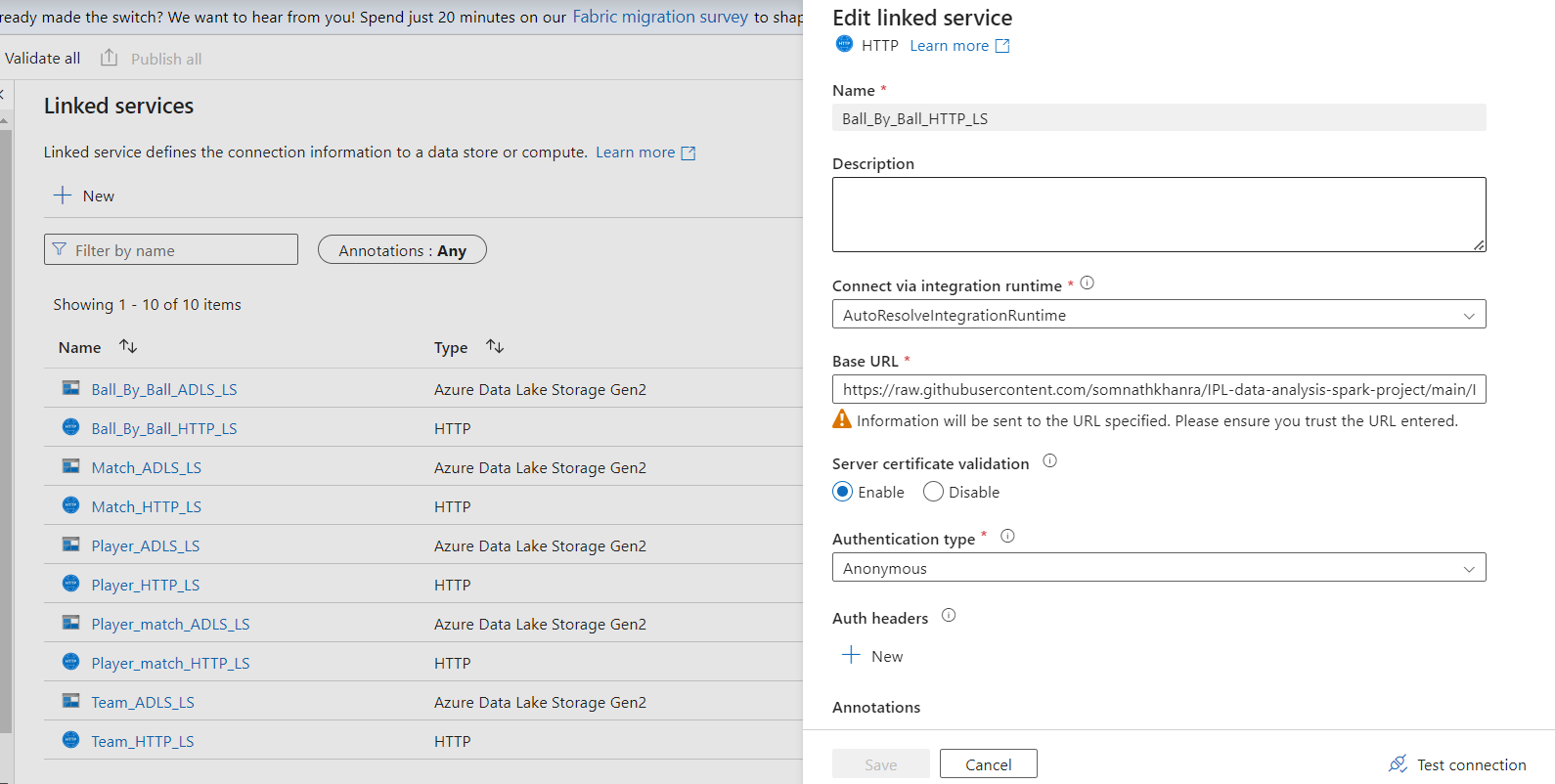
****

**Triggering the pipeline for scheduled data ingestion performing purpose**

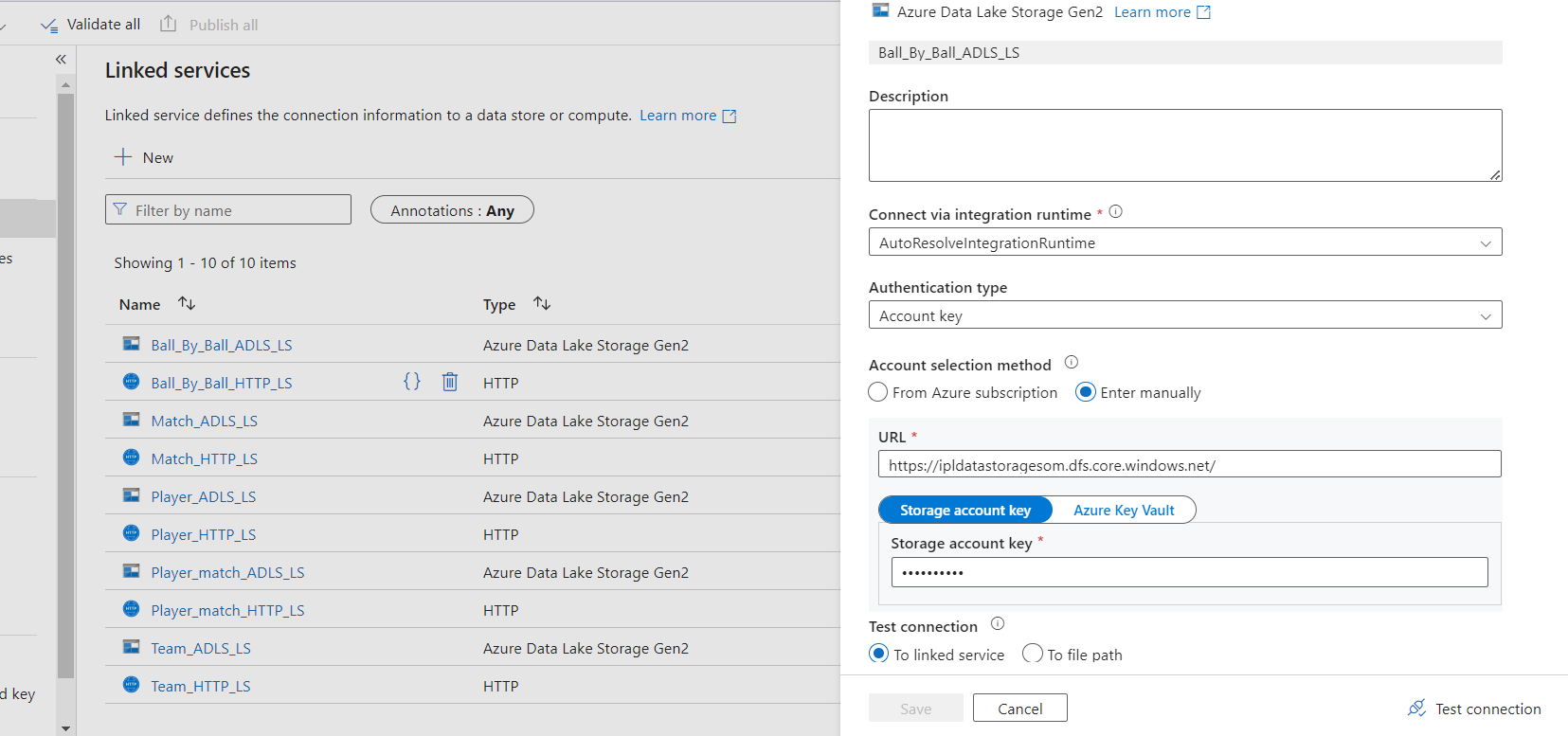
**A screenshot of a computer

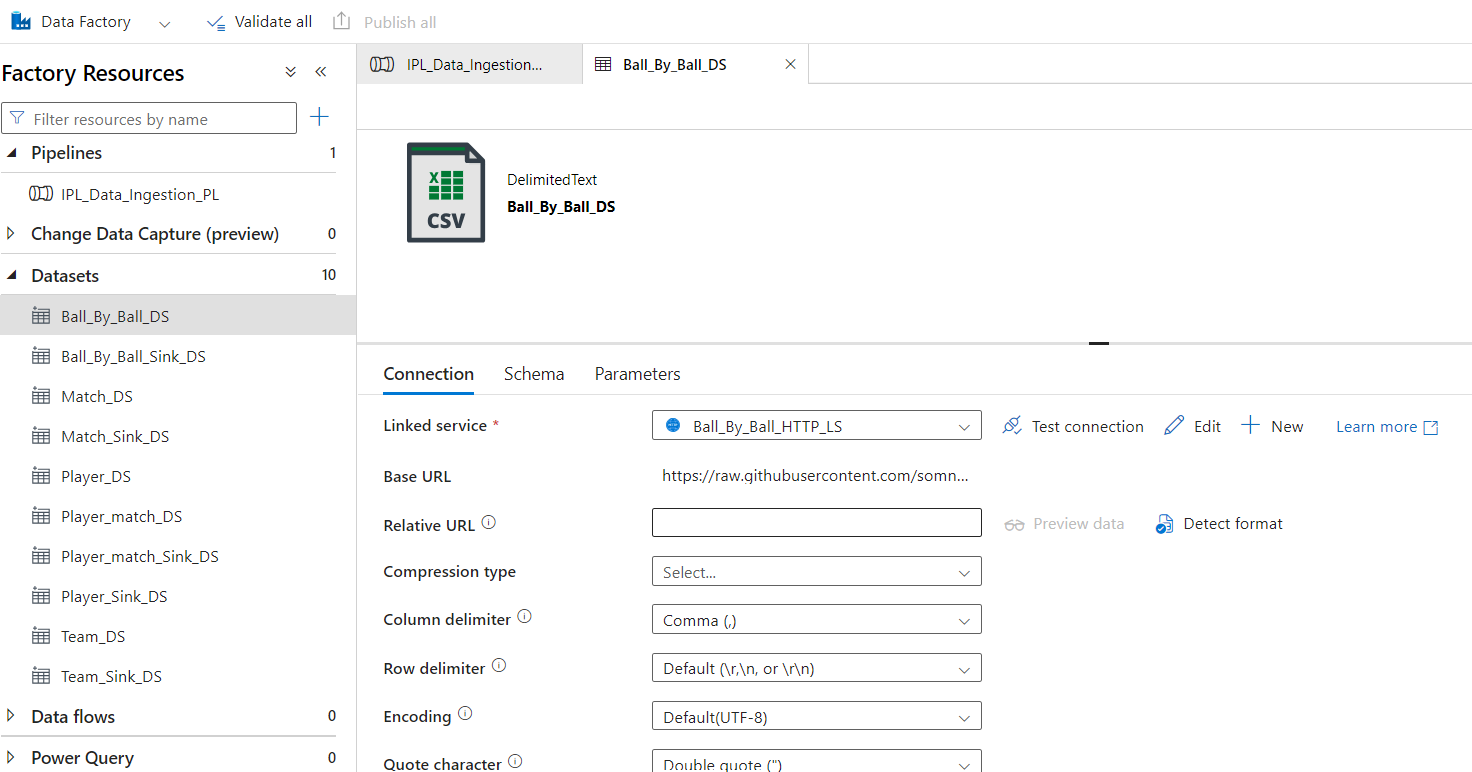
Description automatically generated**

**Configuring Linked services and Datasets to configure ADF pipeline for Source and Target system.**

****

**&&**

****

**And Source and Sink DataSets:  
**

**&&**

**A screenshot of a computer

Description automatically generated**

**&&**

**A screenshot of a computer

Description automatically generated**

# Step 2: Performing Analytics On Databricks

Once the data is ingested into the Azure SQL Database, we can perform analytics using various Azure services such as Azure Databricks or Delta lake services.

## Using Azure Databricks

1. \*\*Create an Azure Databricks Workspace:\*\* Set up the workspace and create a new cluster.

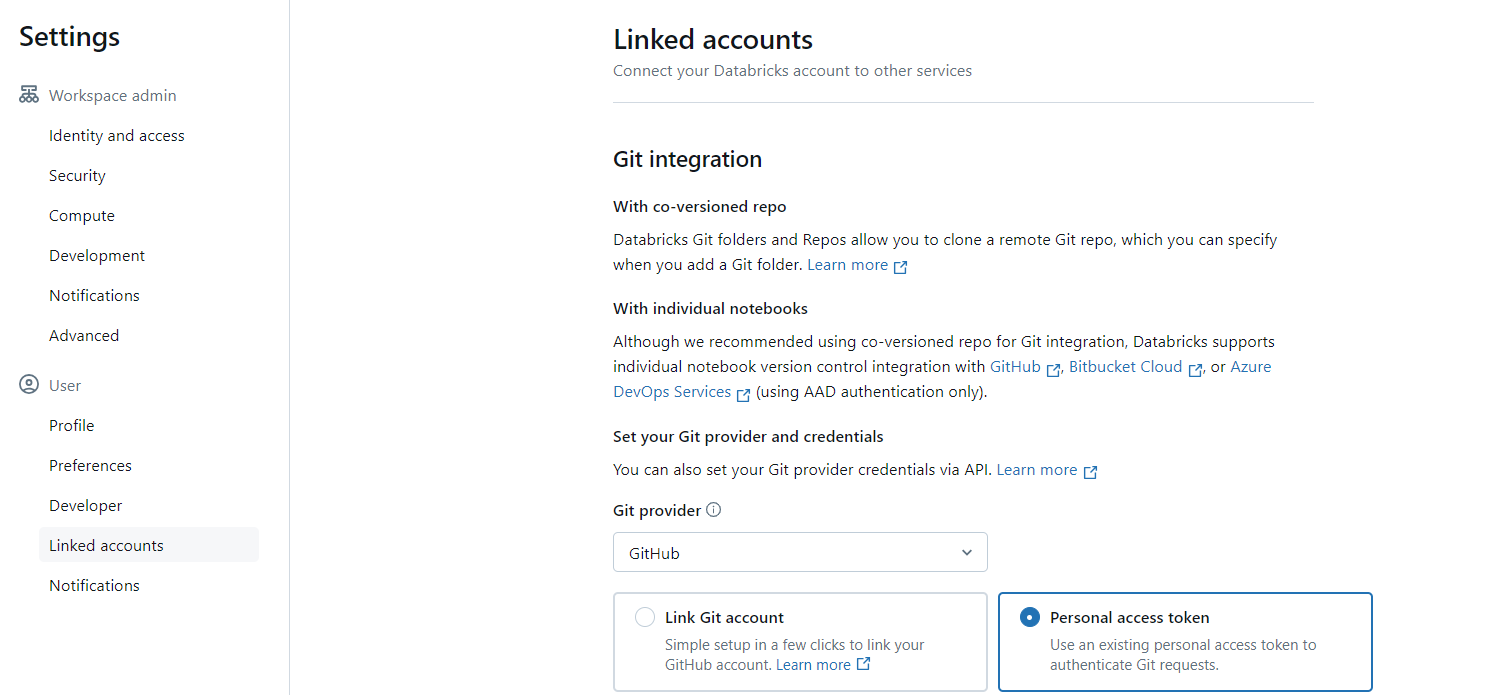
2. \*\*Load Data:\*\* Configure Data laketo mount data files to Databricks file system to load the IPL data into a Databricks notebook.

3. \*\*Perform Analysis:\*\* Utilize Databricks notebooks to perform exploratory data analysis (EDA), visualize data, and derive insights using Spark SQL and Python.

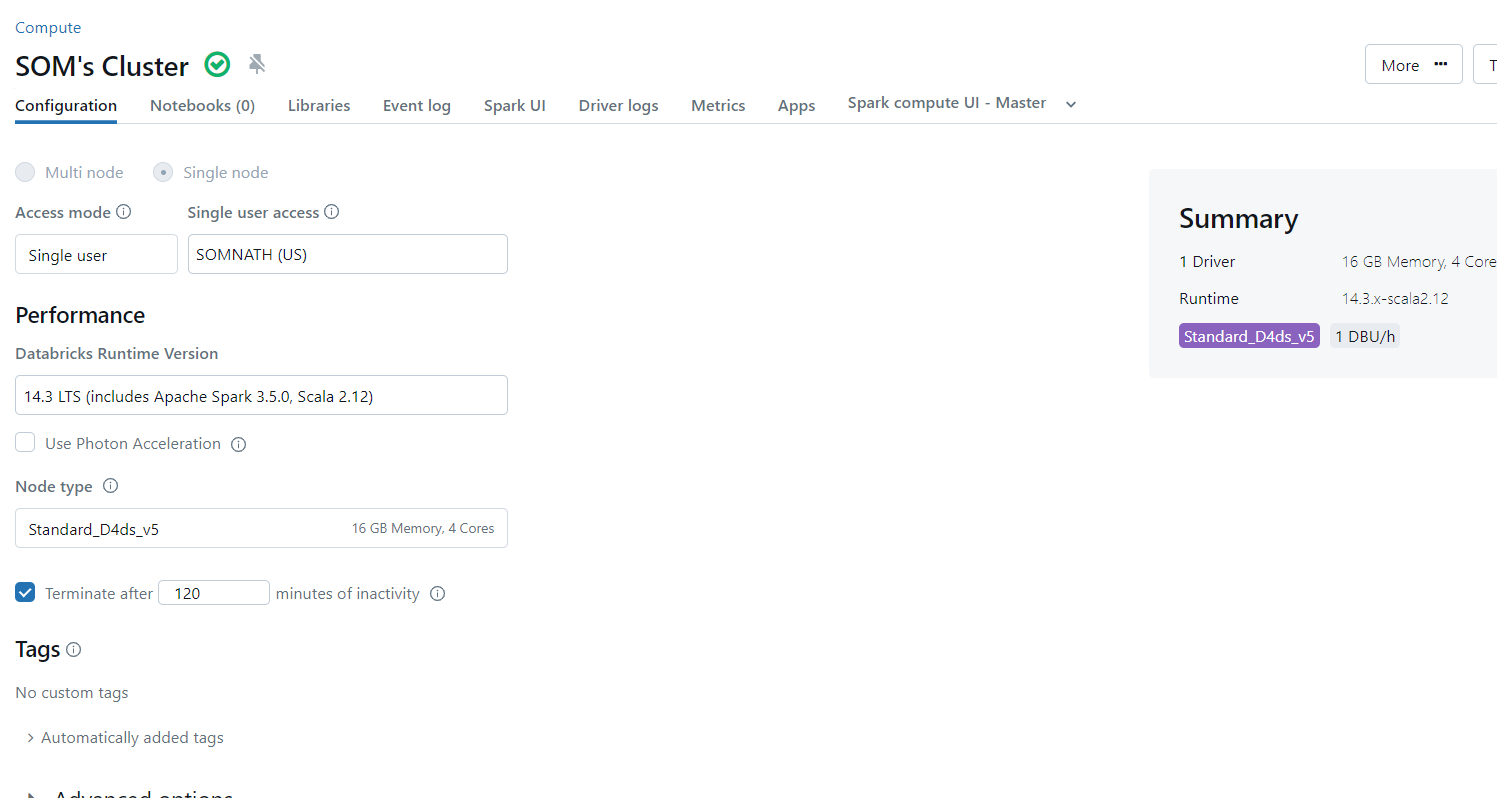
Sharing the SS below:

**Git Configuration with Databricks workspace and perform CI/CD with current version.**

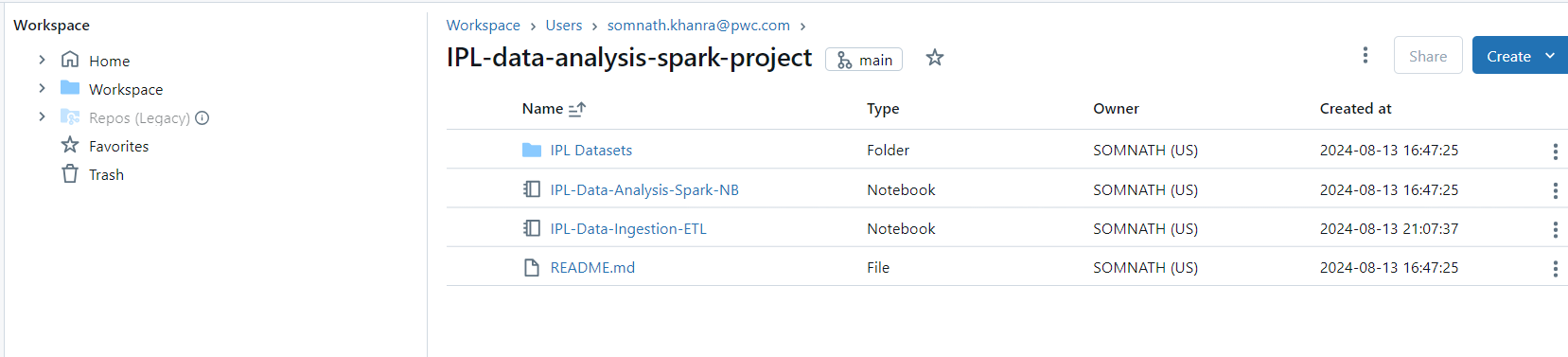
**For me, Gave already configured token with repo read/write action permission from my Git account. You could set your account linking likewise.**

****

**Cluster Configuration to run Databricks notebook or Jobs.**

****

**&&**

****

# Step 3: Creating External and Managed Tables(Delta Table)

In this step, we will create external and managed tables to organize the IPL data within Azure Databricks.

**Writing data to Delta Table after creation(In Delta Format)**

**A screenshot of a computer

Description automatically generated**

**&&**

**A screenshot of a computer

Description automatically generated**

**&&**

**Delta Data location with Parquet Format**

**A screenshot of a computer

Description automatically generated**

## Creating External Tables

1. \*\*Define External Table:\*\* Use the SQL command to create an external table pointing to the raw data in Azure Blob Storage or Data Lake.

A screenshot of a computer

Description automatically generated

&&

## A screenshot of a computer Description automatically generated

## Creating Managed Tables

2. \*\*Define Managed Table:\*\* Load data from the external table into a managed table for better control over the data lifecycle.

```sql

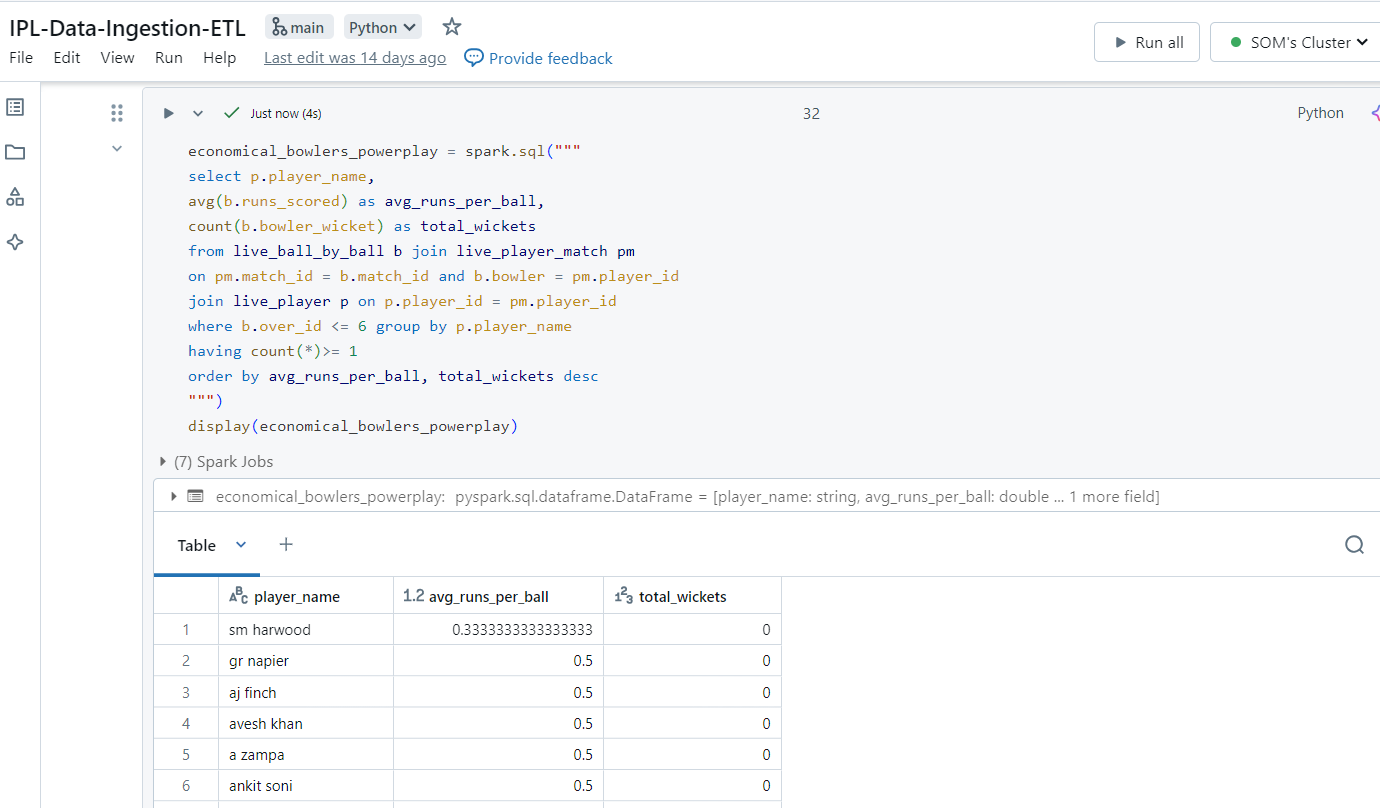
CREATE TABLE managed\_ipl\_data AS

SELECT \* FROM external\_ipl\_data;

```

# Step 4: Querying the Managed Table

**With the managed table created, you can now perform queries to analyze the IPL data.**



&&

A screenshot of a computer program

Description automatically generated

&&

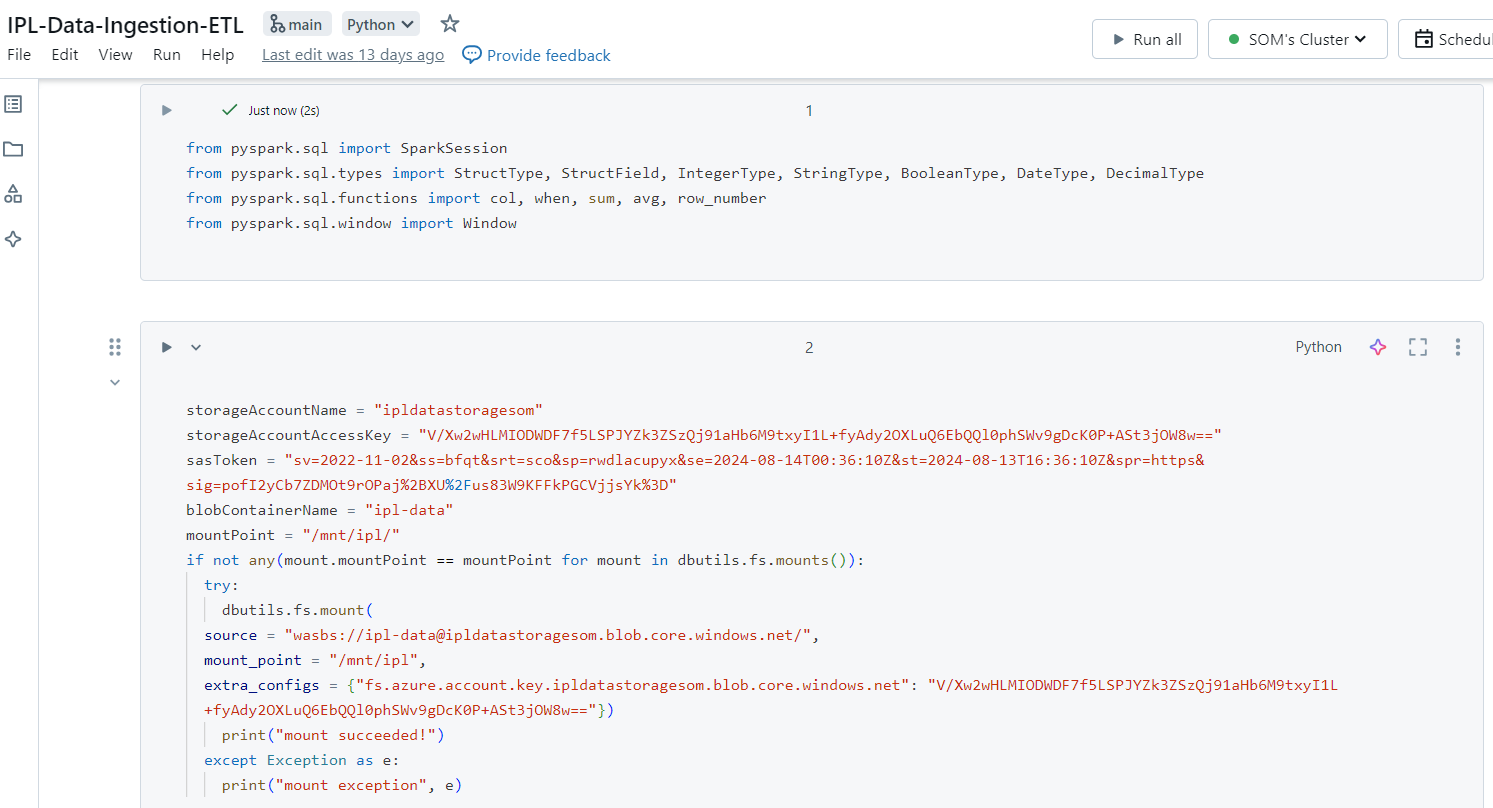
A screenshot of a computer program

Description automatically generated

# Step 5: Performing Data Transformation

Data transformation is crucial for preparing data for analysis. Use Spark SQL or DataFrame API in Databricks to clean and transform the data.

**Configuring connection and mounting to access data lake data from DBFS.**



&&

A screenshot of a computer

Description automatically generated

&&

A screenshot of a computer

Description automatically generated

**Loading data into raw\_data folder and performing cleaning raw data and transforming based on needs.**

**A screenshot of a computer

Description automatically generated**

**Doing the same for data sets:**

**A screenshot of a computer

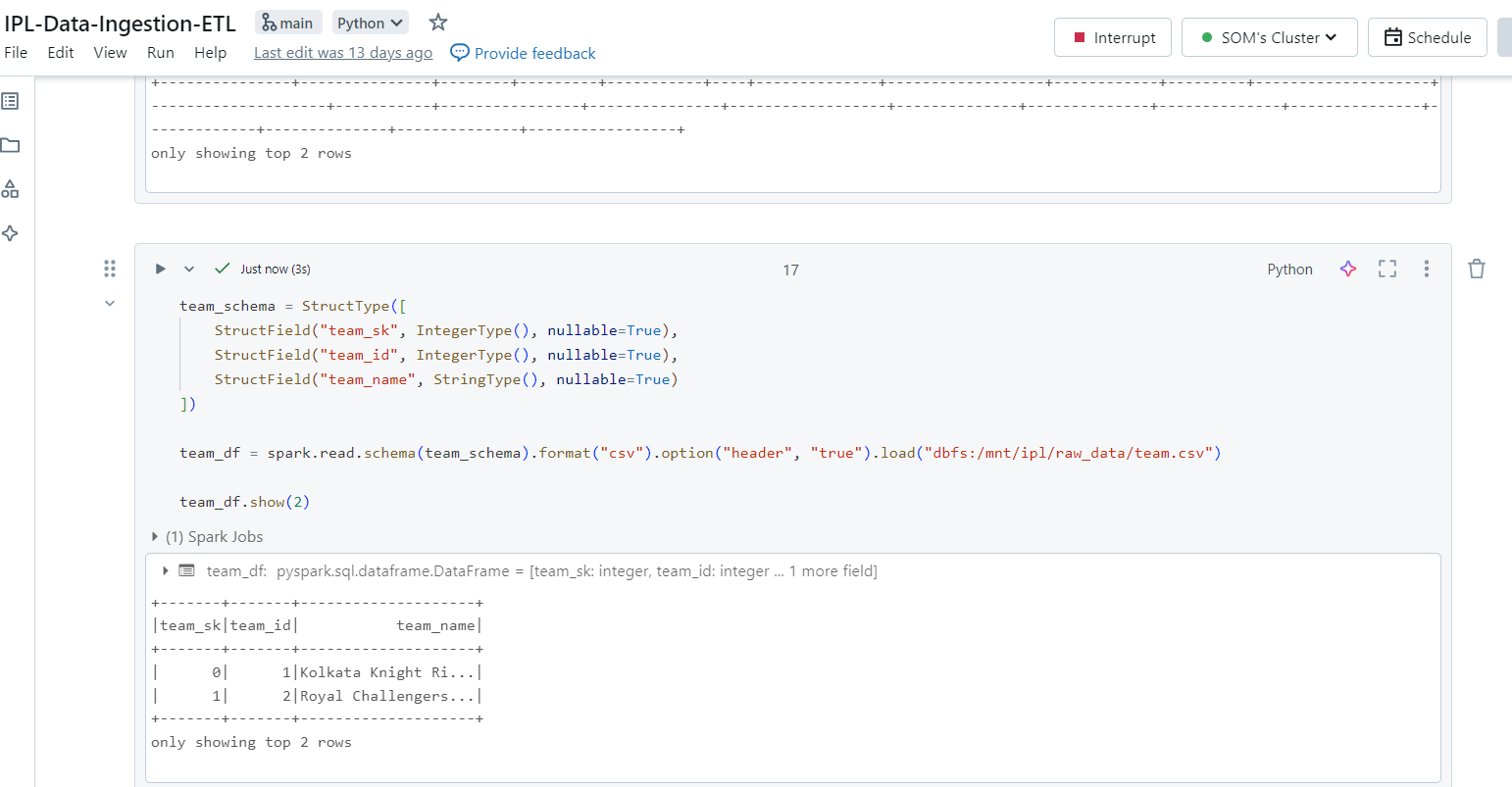
Description automatically generated**

**&&**

**A screenshot of a computer

Description automatically generated**

**&&**

****

**&&**

**A screenshot of a computer

Description automatically generated**

**Sending transformed data files to the target folder called transformed\_data in data lake.**

****

**&&**

**A screenshot of a computer

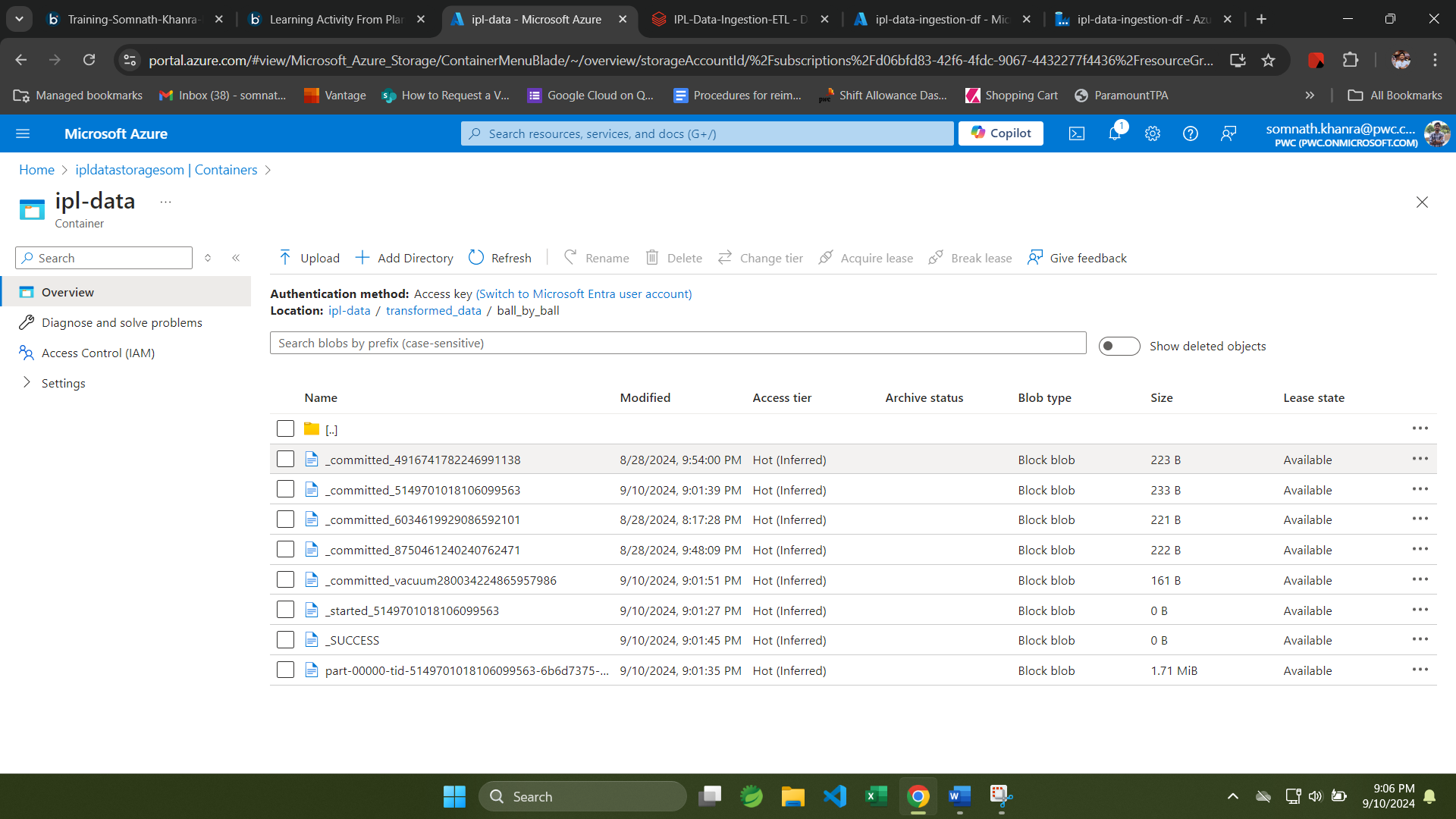
Description automatically generated**

**&&**

**A screenshot of a computer

Description automatically generated**

**&&**

****

# Step 6: Storing Data as Parquet in Data Lake

Finally, store the transformed data as Parquet files in Azure Data Lake for efficient storage and querying.

## Writing Data to Data Lake

1. \*\*Save Data as Parquet:\*\* Use the DataFrame API to write the transformed data to Azure Data Lake.

```python

transformed\_df.write.mode("overwrite").parquet("path\_to\_data\_lake/transformed\_ipl\_data.parquet")

```

A close-up of a text

Description automatically generated

&&

A screenshot of a computer

Description automatically generated

**Visualize The Transformed Query Data using Pandas**

**A screenshot of a computer program

Description automatically generated**

**&&**

**A green bar graph with white text

Description automatically generated**

**&&**

**A screenshot of a computer

Description automatically generated**

**&&**

**A graph of different colored bars

Description automatically generated with medium confidence**

# Conclusion

In this guide, we have walked through the entire process of ingesting IPL data using Azure Data Factory, performing analytics, creating tables, querying data, transforming data, and storing it as Parquet files in a data lake. By following these steps, you can leverage the power of Azure's data integration and analytics services to derive valuable insights from your IPL data.