Large Scale Data Processing



Project Overview

In today's data-driven world, the ability to efficiently process and analyze large volumes of data is crucial for businesses to gain insights and make informed decisions. This project aims to leverage the power of Azure Databricks and PySpark to perform large-scale data processing tasks, including Extract, Transform, and Load (ETL) operations, on massive datasets. By utilizing Databricks clusters, we ensure scalability and parallel processing capabilities.



About Project

This project leverages Azure Databricks and PySpark for large-scale data processing on a sample CSV file containing employee data. The dataset consists of more than 100,000 records with 11 fields including employee ID, full name, job title, department, gender, ethnicity, age, hire date, annual salary, bonus percentage, country, and exit date.

# Architectural Diagram





Key-Components/Requirements of the projects

# Azure Databricks:

* + Azure Databricks provides a cloud-based platform for big data analytics and machine learning. It offers a collaborative environment for data engineers, data scientists, and analysts to work together seamlessly.
  + Databricks provides managed Spark clusters, eliminating the need for infrastructure management and allowing teams to focus on data processing tasks.

# PySpark**:**

* + PySpark is the Python API for Apache Spark, a powerful open-source framework for distributed data processing. PySpark simplifies development tasks by providing a Python interface to Spark's capabilities.
  + With PySpark, developers can write concise and expressive code to perform complex data transformations, aggregations, and analytics on large datasets.

# ETL Operations:

* + **Extract**: Data ingestion from various sources such as databases, data lakes, streaming platforms, or external APIs.
  + **Transform**: Data transformation tasks including cleansing, filtering, aggregating, joining, and enriching datasets to prepare them for analysis.
  + **Load**: Storing processed data into target systems such as data warehouses, data lakes, or serving layers for downstream consumption.

# Scalability with Databricks Clusters:

* + Databricks clusters dynamically allocate computational resources based on workload requirements, ensuring optimal performance and resource utilization.
  + Autoscaling capabilities automatically adjust cluster size to accommodate changes in workload demand, allowing for seamless scalability without manual intervention.
  + Databricks Runtime optimizes performance with built-in optimizations, caching, and tuning for various workloads, resulting in faster processing times.



Azure Resources Used for this Project

# Azure Data Lake Storage Gen2:

* + This is where the Transformed data is Loaded. Azure Data Lake Storage Gen2 provides a scalable and secure platform for storing large volumes of data. It enables us to manage, access, and analyse data effectively

# Azure Blob Storage:

* + This is where the raw data is stored. Azure Blob Storage integral to Microsoft Azure's storage service, is a cloud-based solution tailored for managing vast amounts of unstructured data, encompassing both text and binary data. Termed "Blob" for "Binary Large Object," it signifies a compilation of binary data treated as a singular entity within a database.

# Databricks Cluster

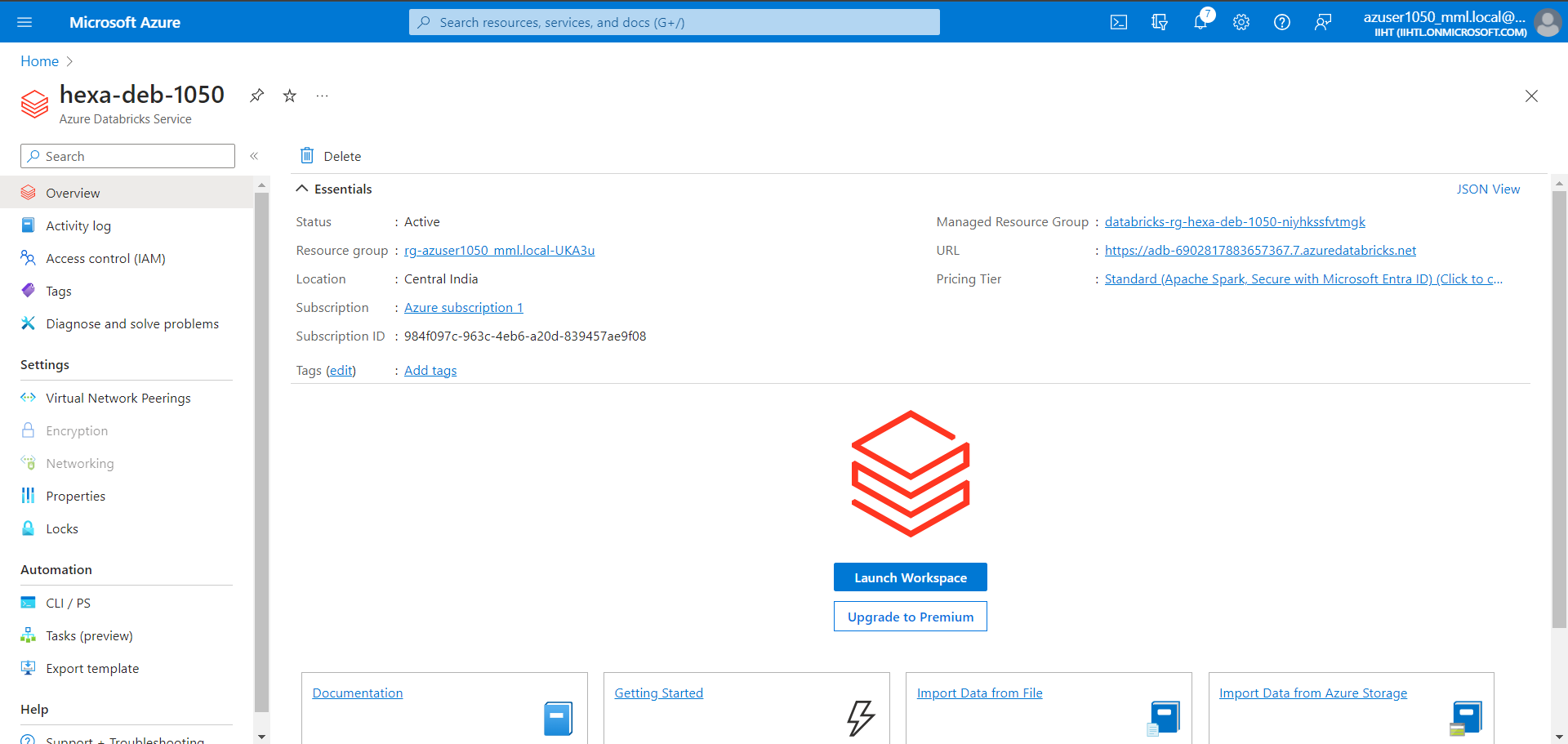
* + An Azure Databricks cluster process the data depending on the user instructions in the Azure Notebook. It serves as a computational resource facilitating the processing of extensive data and execution of analytics workloads through the Apache Spark platform within the Microsoft Azure cloud.



How It works

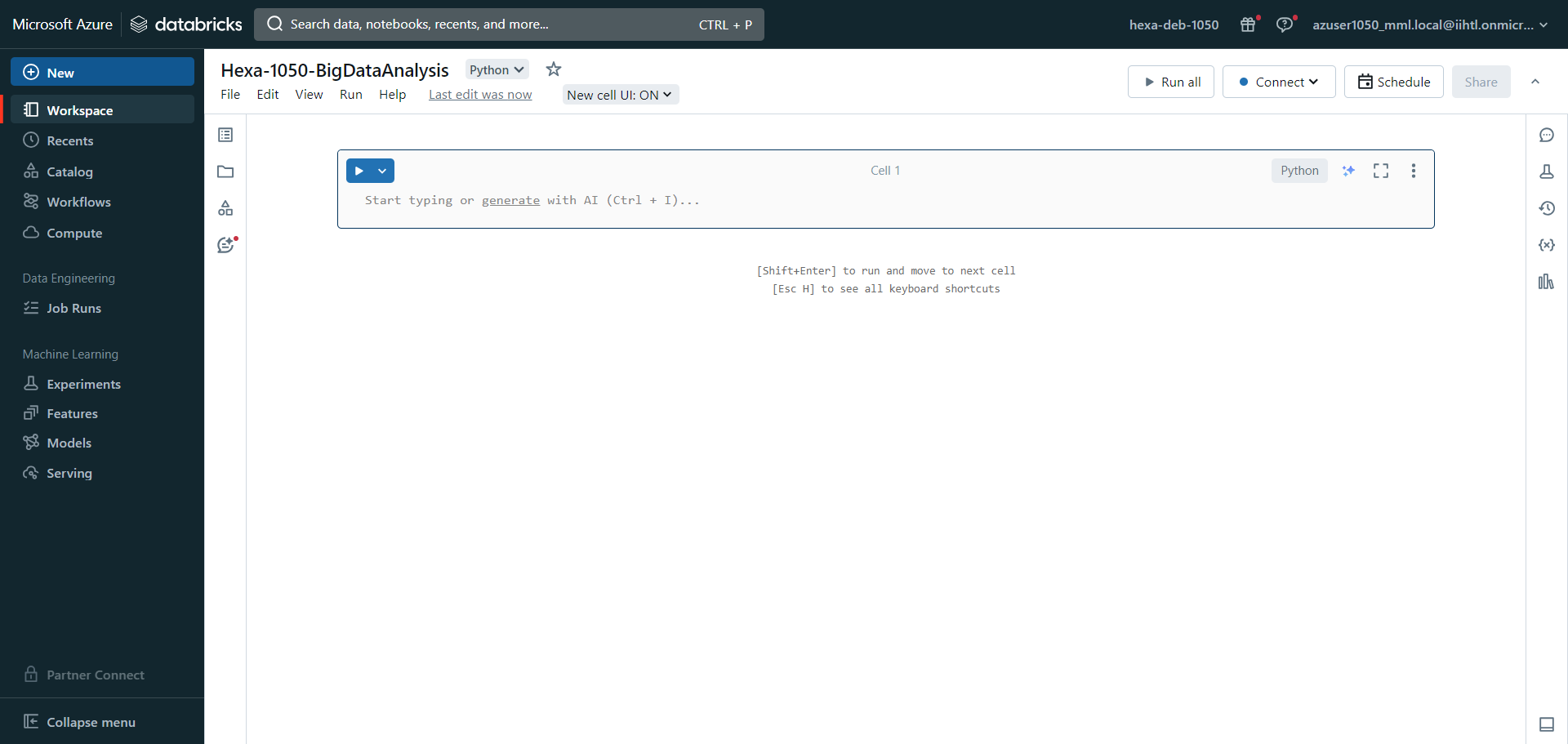
# Setting Up Azure Databricks Environment:

* + Sign in to the Azure portal and create an Azure Databricks workspace. Configure workspace settings, including pricing tier, region, and workspace name



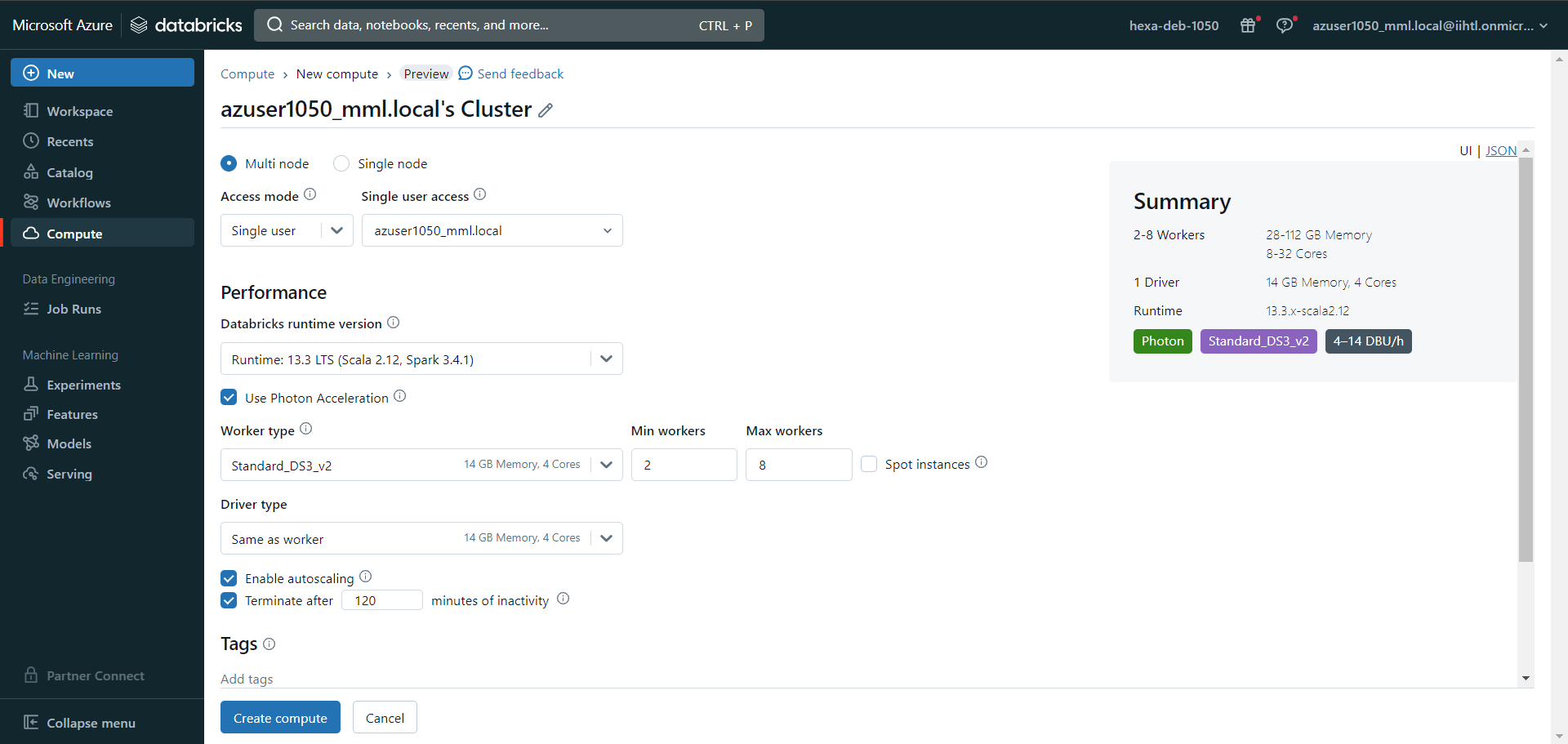
# Developing PySpark Notebooks:

* + Create a new PySpark notebook within the Databricks workspace. Begin writing PySpark code to perform ETL operations, data transformations, and other data processing tasks.



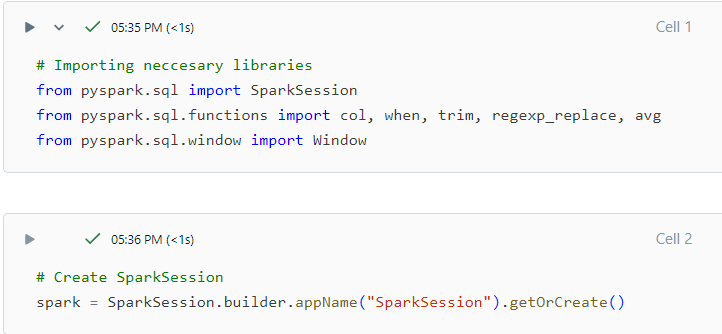
# Create Cluster and Connecting to notebook

* + The cluster is created with 4 working nodes and autoscaling is enabled which automatically adjust cluster size to accommodate changes in workload demand, allowing for seamless scalability without manual intervention.



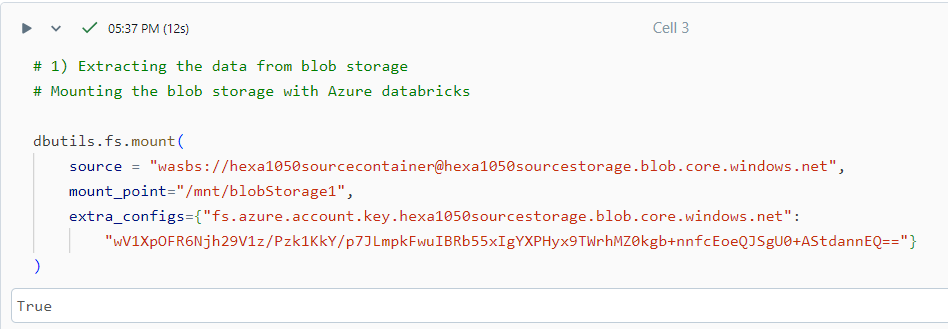
# Importing Necessary libraries and Creating Spark Session:

* + Use SparkSession.builder to configure and create a SparkSession. specify the application name using .appName() and configure any additional Spark options using .config(). Finally, call .getOrCreate() to either create a new SparkSession

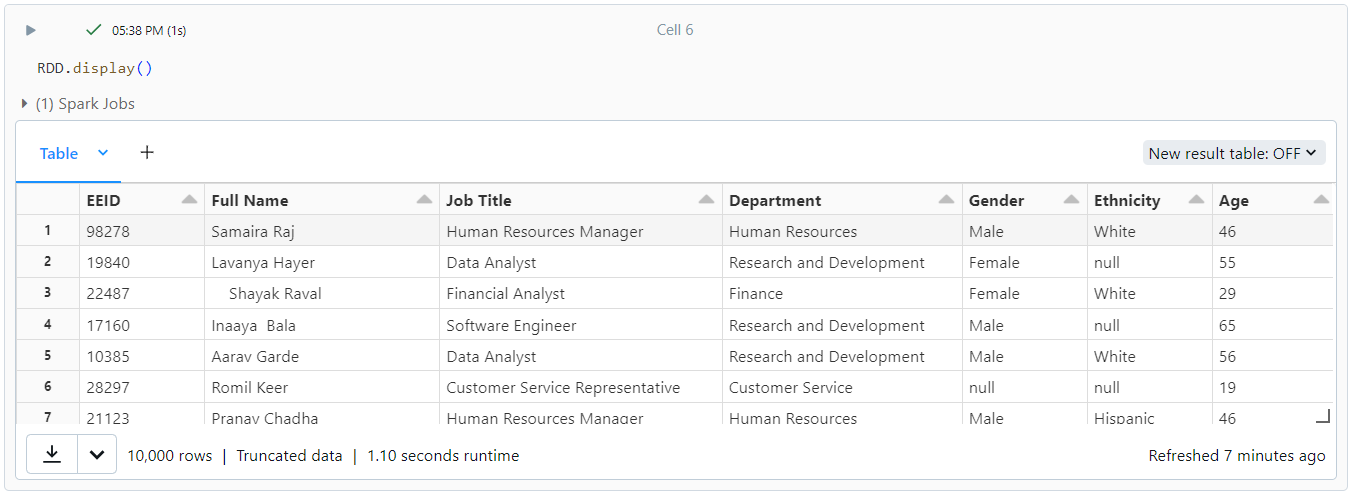


# Extracting Data from Source storage

* + Connecting data source (Azure Blob Storage) by mounting it to the Databricks File System (DBFS) to simplify data access
  + It helps to retrieve raw data for processing and analysis within the PySpark environment.





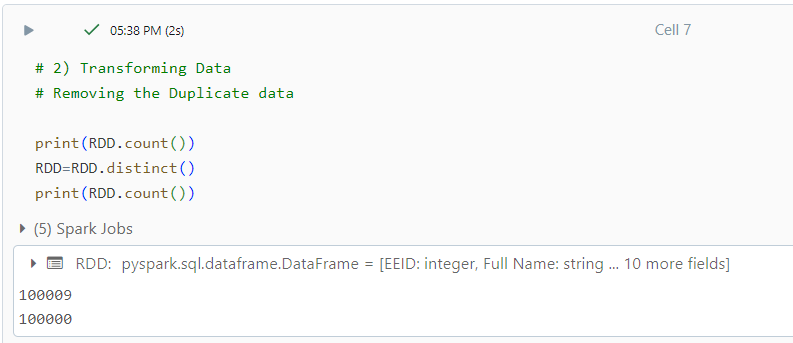


# Transforming the raw data

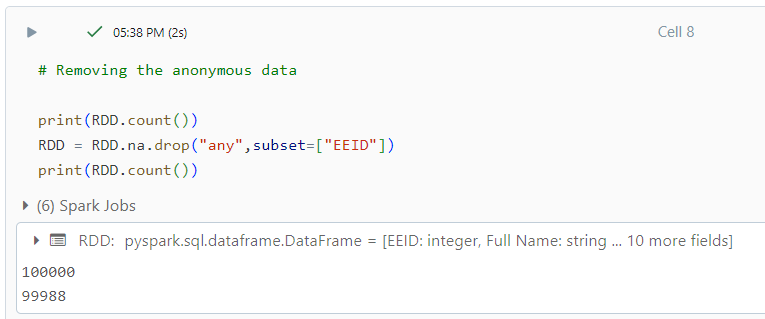
* + Utilize PySpark DataFrame transformations and functions to cleanse, transform, and prepare the data for analysis.
  + Implement business logic and data processing steps to transform raw dataset up to mark for data analysis purpose.

Transformations done:-

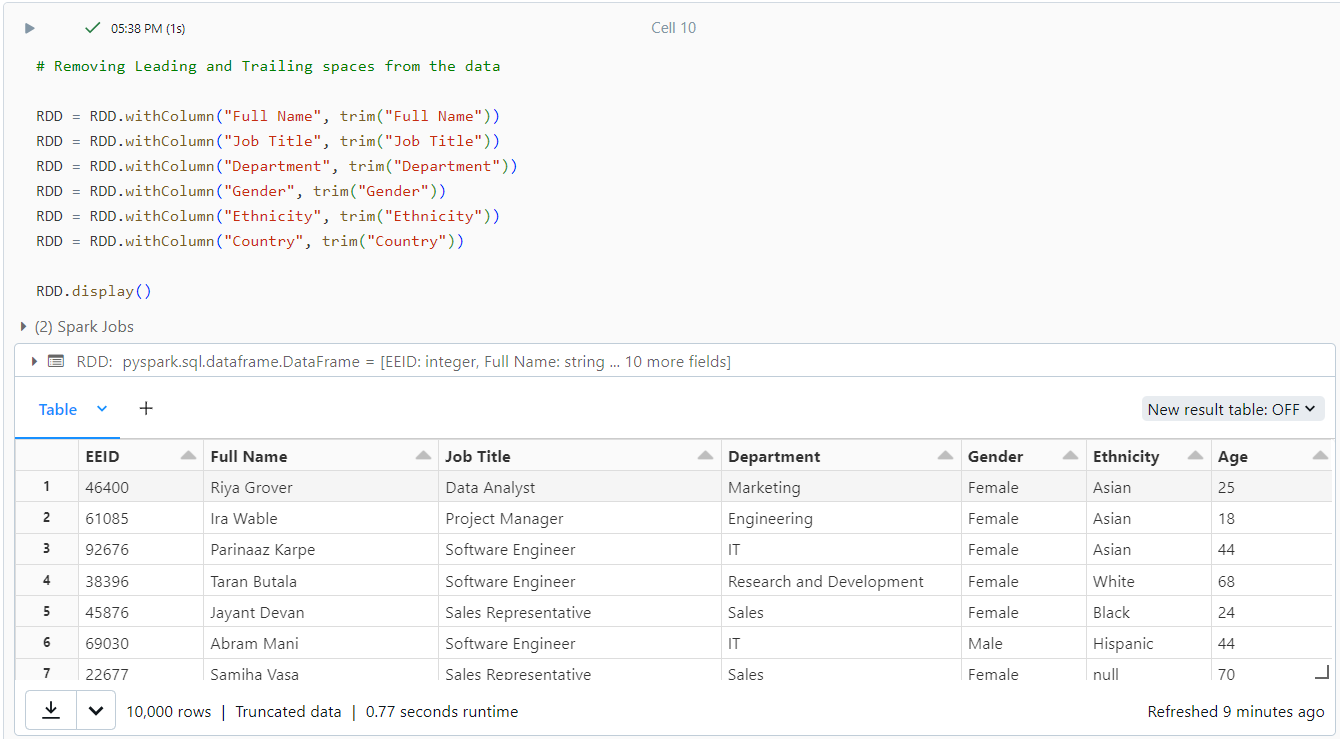
* Removing the Duplicate records



* Handling anonymous data

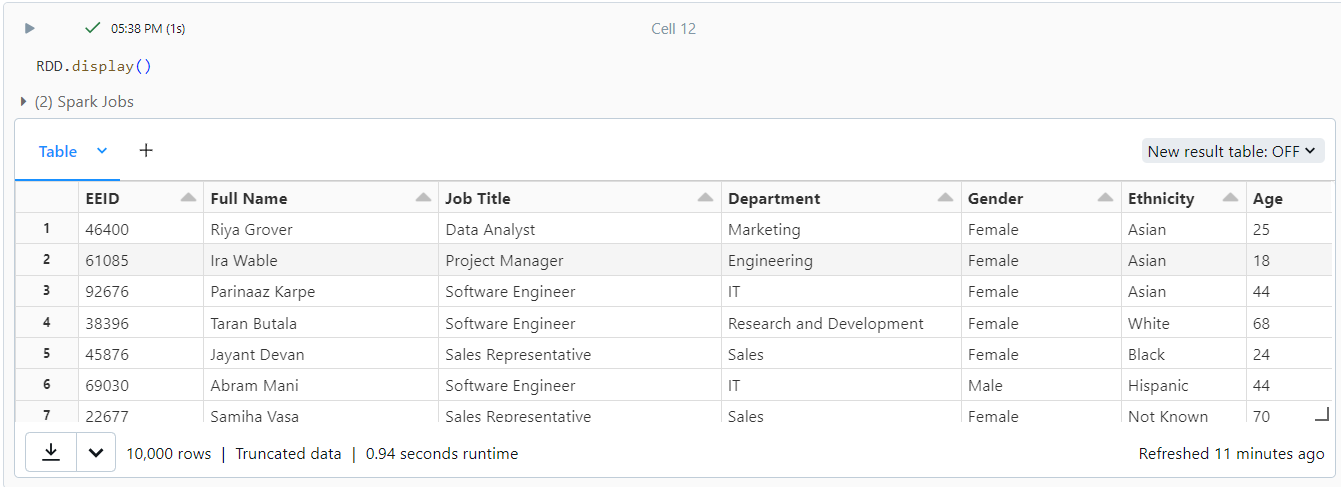


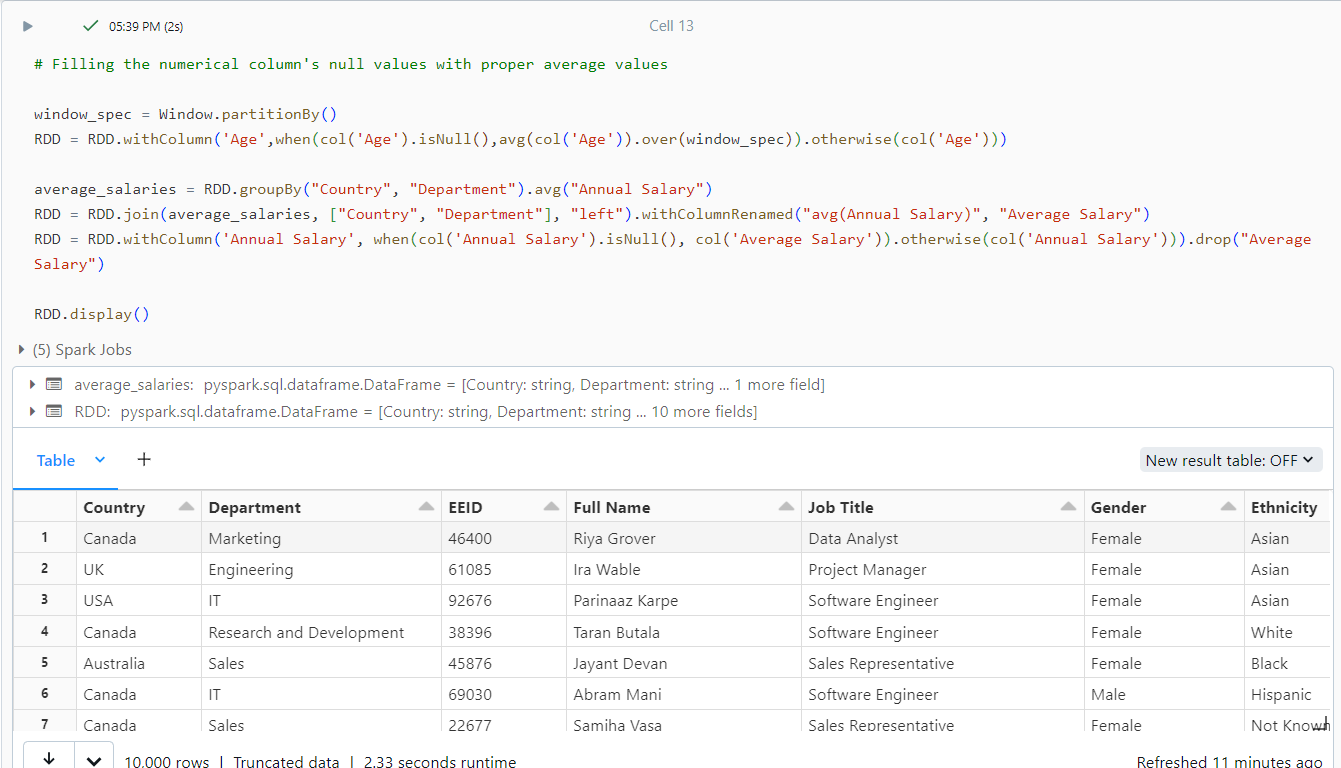
* Removing Extra spaces form the data



* Filling null values with proper messages and data



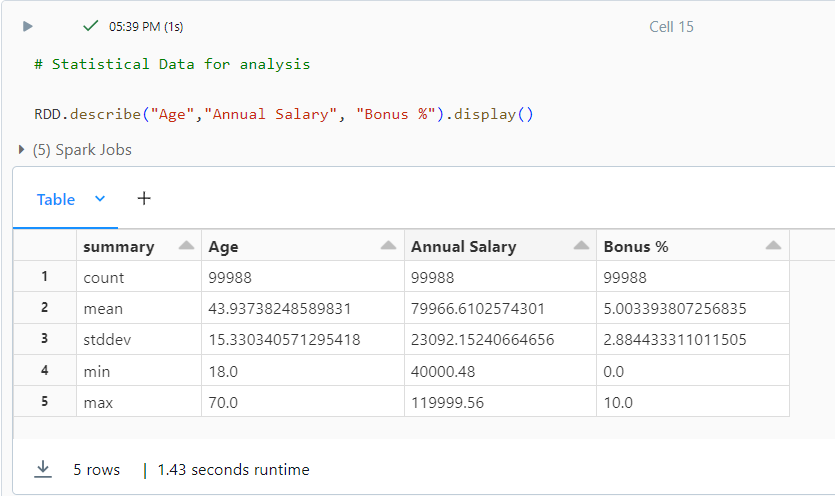




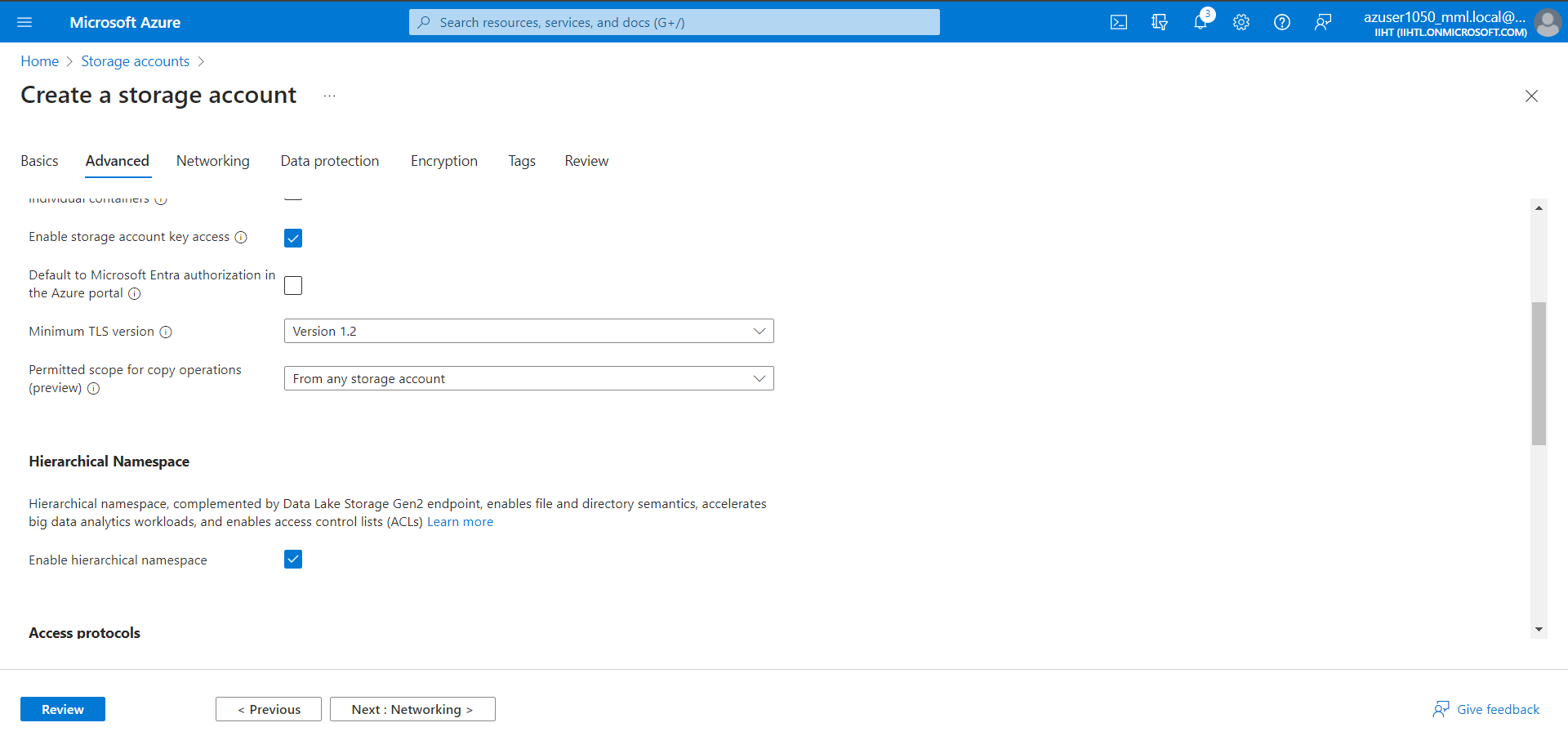
* Renaming USA to US to make dataset consistent

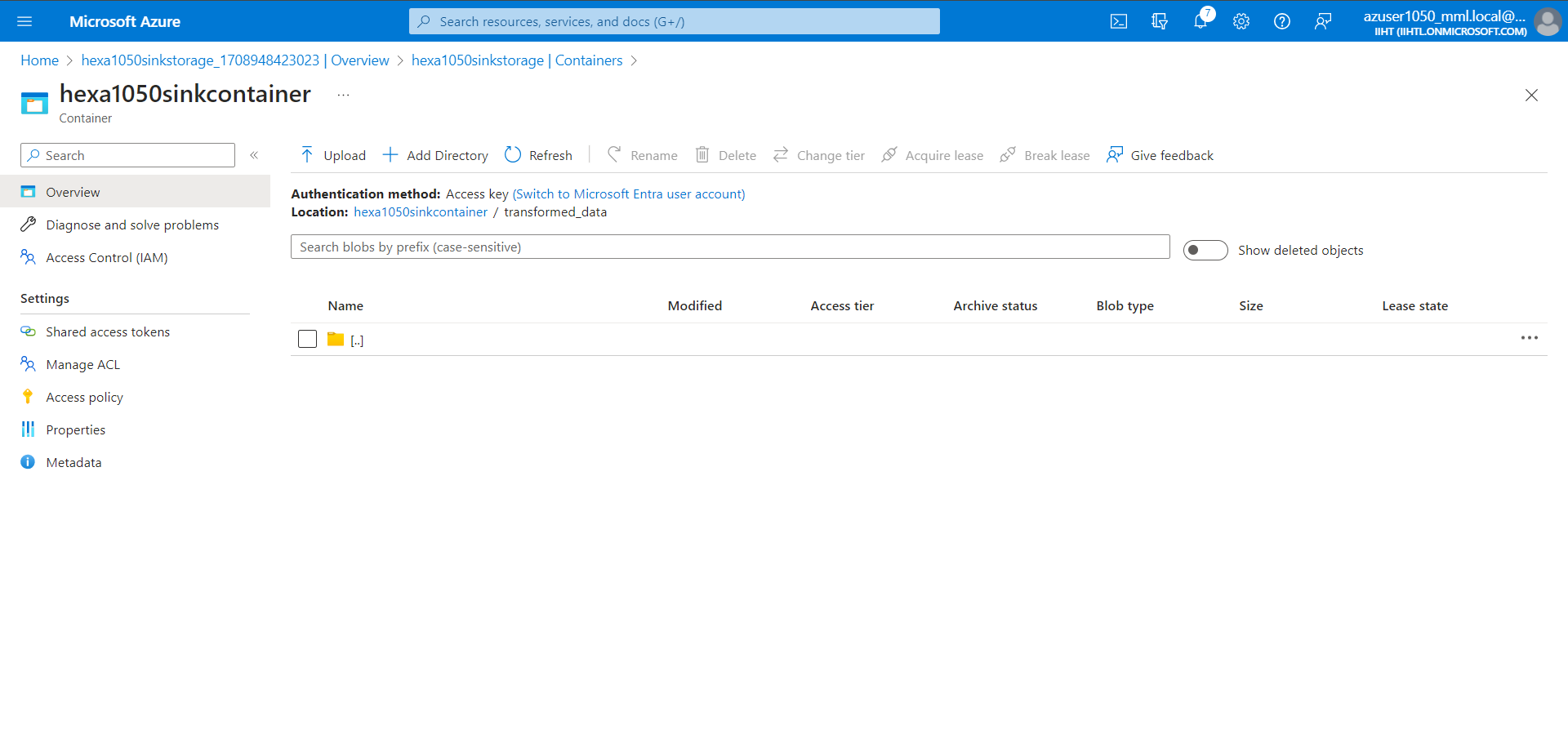


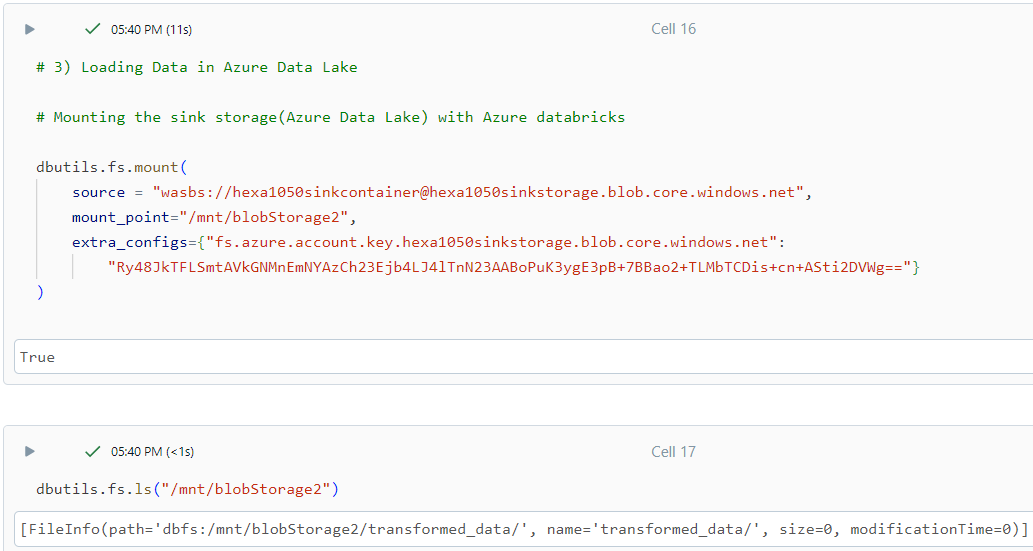
* Statistical data

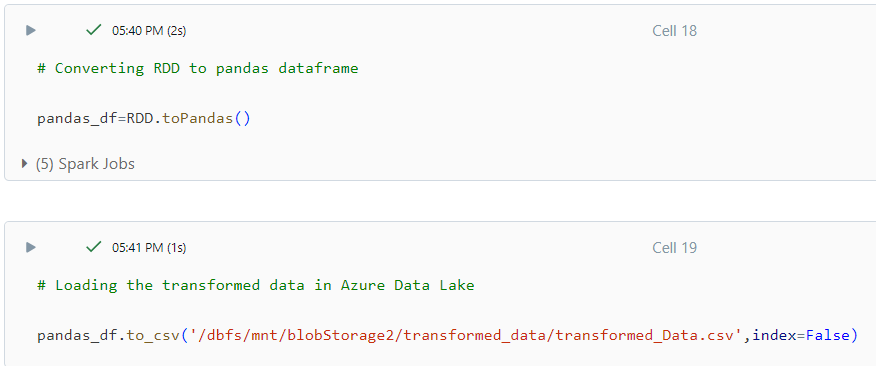


1. Loading Data into Sink Storage
   * To store the transformed data we need to create sink storage (Azure Data Lake) and Container
   * Connecting data source (ADLS) by mounting it to the Databricks notebook to load the data

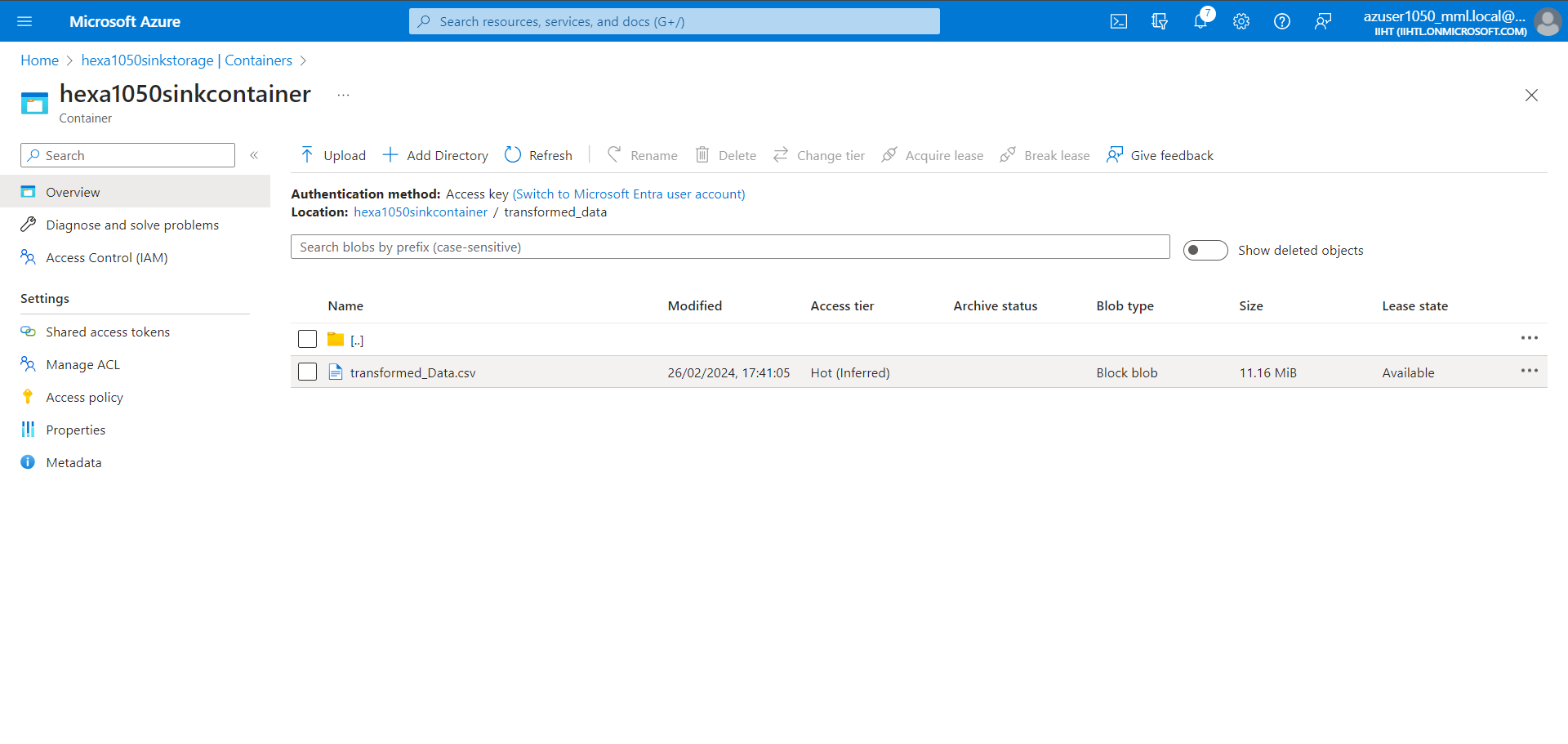




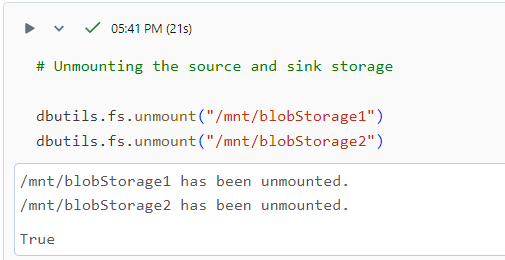




* + Data Successfully Loaded

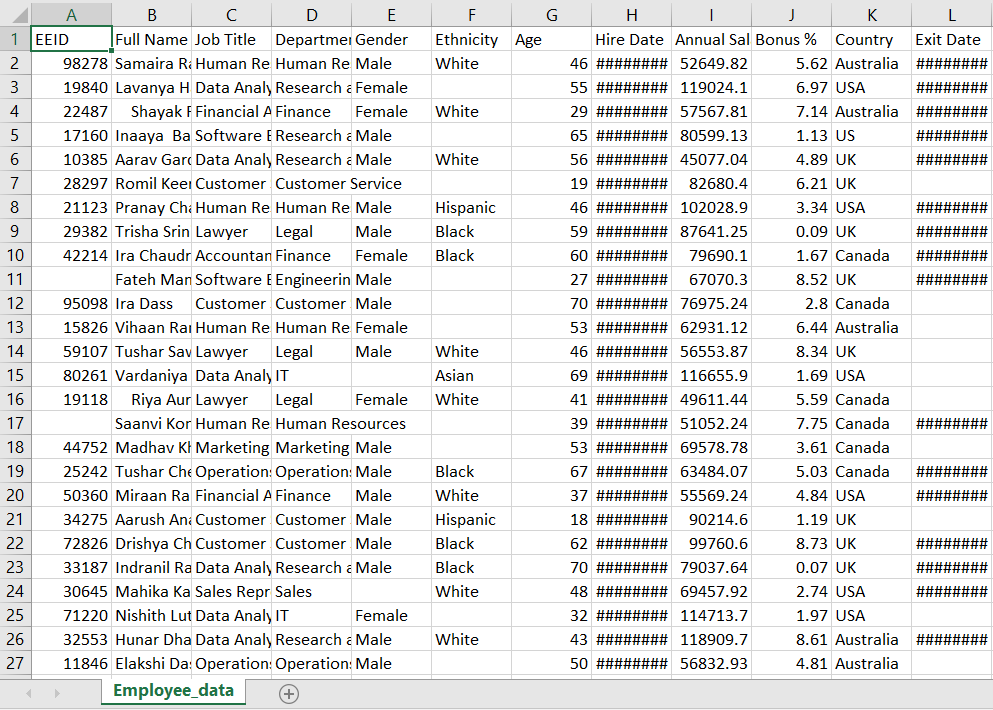


# Unmounting the Source and Sink storage



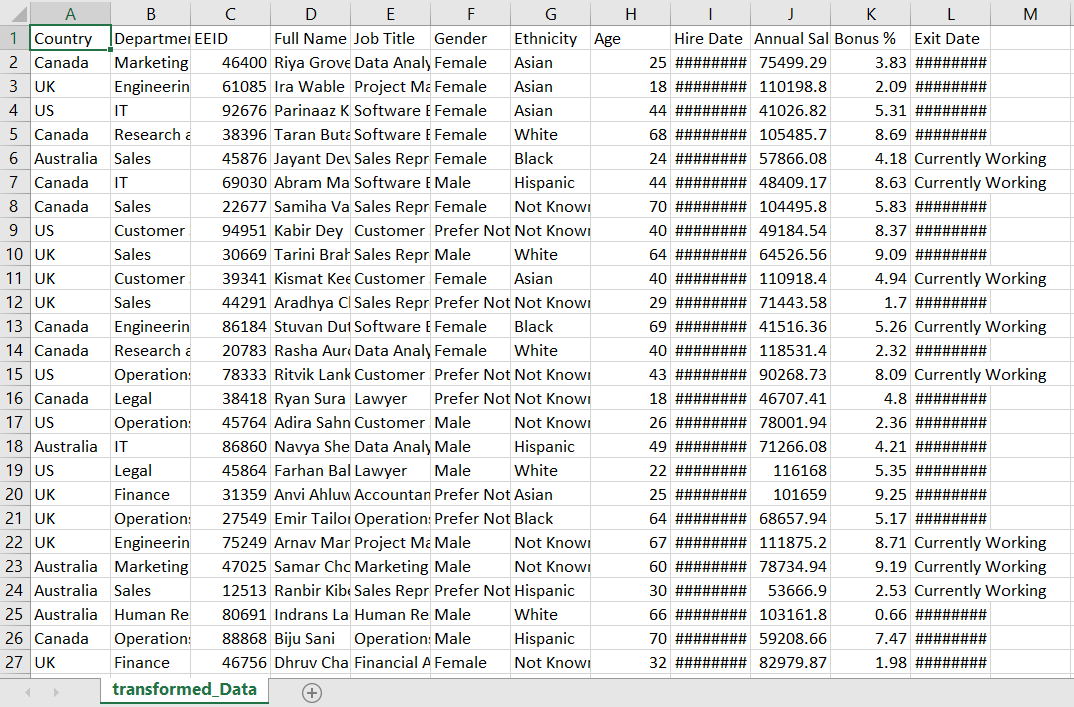


Raw Data





Transformed Data





Conclusion

In conclusion, this project successfully demonstrated the utilization of Azure Databricks and PySpark for large-scale data processing on a sample CSV file containing employee data. Through the implementation of Extract, Transform, Load (ETL) operations, the dataset was cleaned, transformed, and prepared for further analysis or reporting. Overall, this project serves as a practical demonstration of how Azure Databricks and PySpark can be effectively utilized for large-scale data processing.