**Log Analysis with Spark**

**By**

**Karthikeyan K R**

**Log Analysis with Spark**

|  |  |  |
| --- | --- | --- |
| S.No | Topic | Pg.No |
| 1 | Project Statement | 2 |
| 2 | Project Overview | 2 |
| 3 | Project Requirements | 2 |
| 3.1 | Azure Subscription | 2 |
| 3.2 | Data Sources | 2 |
| 3.3 | Azure Data Factory | 2 |
| 3.4 | Azure Storage | 2 |
| 3.5 | Azure Databricks | 2 |
| 4 | Architecture Diagram | 3 |
| 5 | Execution Overview | 3 |
| 6 | Project Implementation | 3 |
| 6.1 | Azure Account | 3 |
| 6.2 | Creating Azure Blob Account | 4 |
| 6.3 | Uploading Log File in Blob Account | 6 |
| 6.4 | Creating Azure Data Lake Storage Gen 2 | 7 |
| 6.5 | Creating Container On ADLS | 9 |
| 6.6 | Azure Data Factory Configuration | 9 |
| 6.7 | Azure Databricks Configuration | 17 |
| 6.8 | Connecting ADLS With Databricks | 21 |
| 6.9 | Read Log Files | 23 |
| 6.10 | Error Finding On Log Files | 27 |
| 6.11 | Anomaly Detection On Log Files | 29 |
| 6.12 | Unmount ADLS from Databricks | 30 |
| 7 | Conclusion | 30 |

**1)Project statement**

Utilize Azure Data Factory to ingest log data into Azure Storage, and leverage Azure Databricks for log analysis using Apache Spark for insights such as error rates, trends, and anomalies.

**2)Project Overview**

This project aims to provide a data analysis solution for Log files using Azure Databricks. Once the data is stored in Azure Storage, the project aims to leverage Azure Databricks, a cloud-based analytics platform, to perform log analysis using Apache Spark. The focus of the analysis is to extract valuable insights from the log data, such as error rates, trends, and anomalies.

**3)Project Requirements**

1. Azure Subscription
2. Data Sources
3. Azure Data Factory
4. Azure Storage
5. Azure Databricks
6. Spark-based Log Analysis
7. Visualization and Reporting

**3.1 Azure Subscription**

* You need one Azure subscription account to perform this project.

**3.2 Data Sources**

* I downloaded sample log file from one public repository. From that repository I took “Traffic Violation.csv” as “Traffic.csv”.
* Repo Link: [Click here](https://github.com/pycaret/pycaret/blob/master/datasets/Traffic%20Violaions.csv) (https://github.com/pycaret/pycaret/blob/master/datasets/Traffic%20Violaions.csv)

**3.3 Azure Data Factory**

* Azure Data Factory is Azure's cloud ETL service for scale-out serverless data integration and data transformation. It offers a code-free UI for intuitive authoring and single-pane-of-glass monitoring and management.

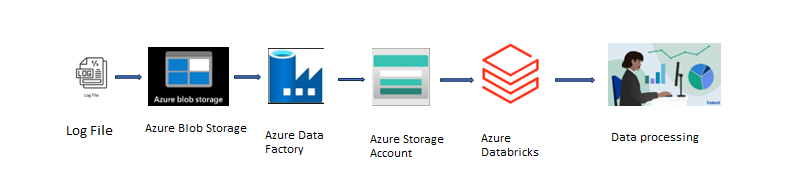
**3.4 Azure Storage**

* The Azure Storage platform is Microsoft's cloud storage solution for modern data storage scenarios. Azure Storage offers highly available, massively scalable, durable, and secure storage for a variety of data objects in the cloud.

**3.5 Azure Databricks**

* Azure Databricks provides the latest versions of Apache Spark and allows you to seamlessly integrate with open source libraries. Spin up clusters and build quickly in a fully managed Apache Spark environment with the global scale and availability of Azure.
* Log analysis and Visualization will be done on Azure Databricks.

**4)Architecture diagram**



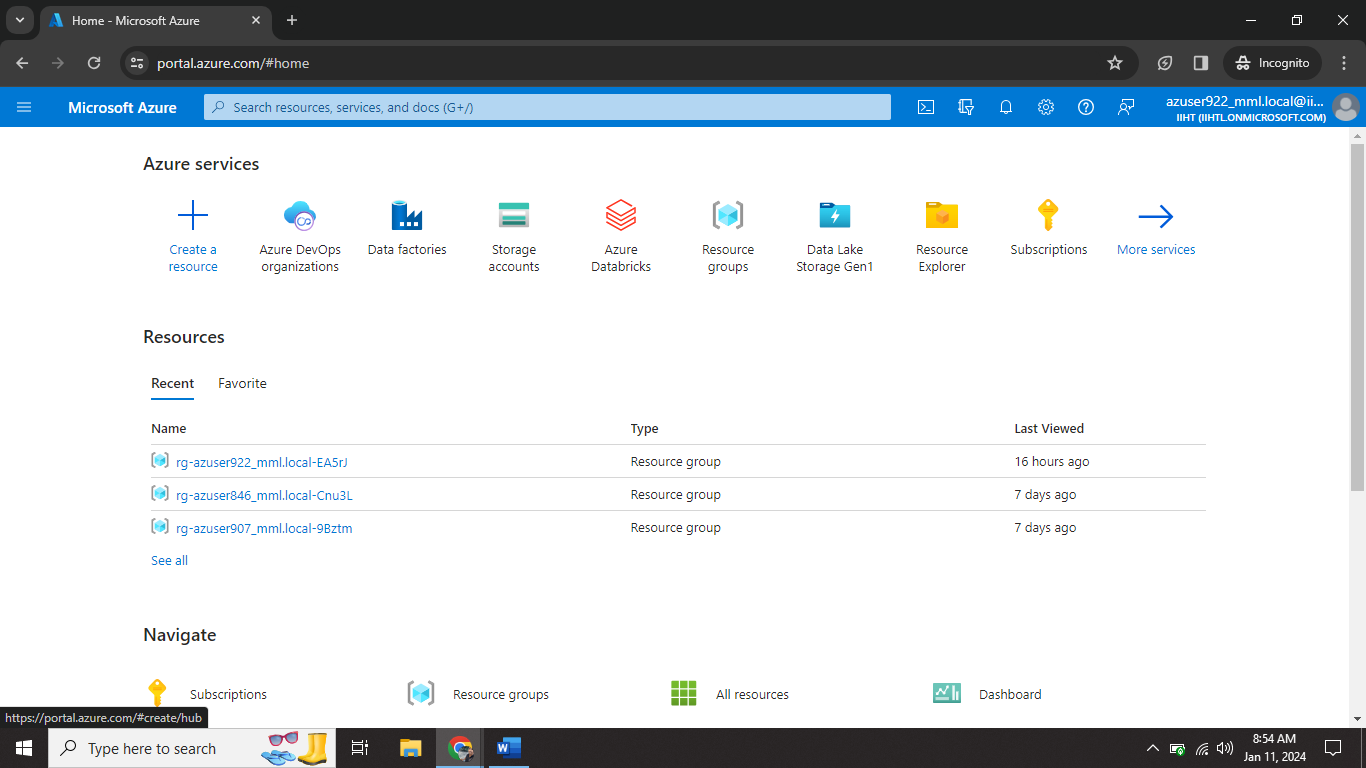
**5)Execution Overview**

* Azure Data Factory (ADF) is responsible for the execution of Azure Datarbicks notebooks as well as monitoring them. We import data from Azure Blob Storage to Azure Data Lake Storage Gen2 (ADLS). The raw data is stored in the container at **Bronze zone** (landing zone).
* Data in the Bronze zone is ingested using Azure Databricks NoteBook. This level is called **Silver Zone** (standardization zone).
* Ingested data in Silver Zone is used for performance on Azure Databricks NoteBook. This level is called **Gold Zone** (analytical zone).

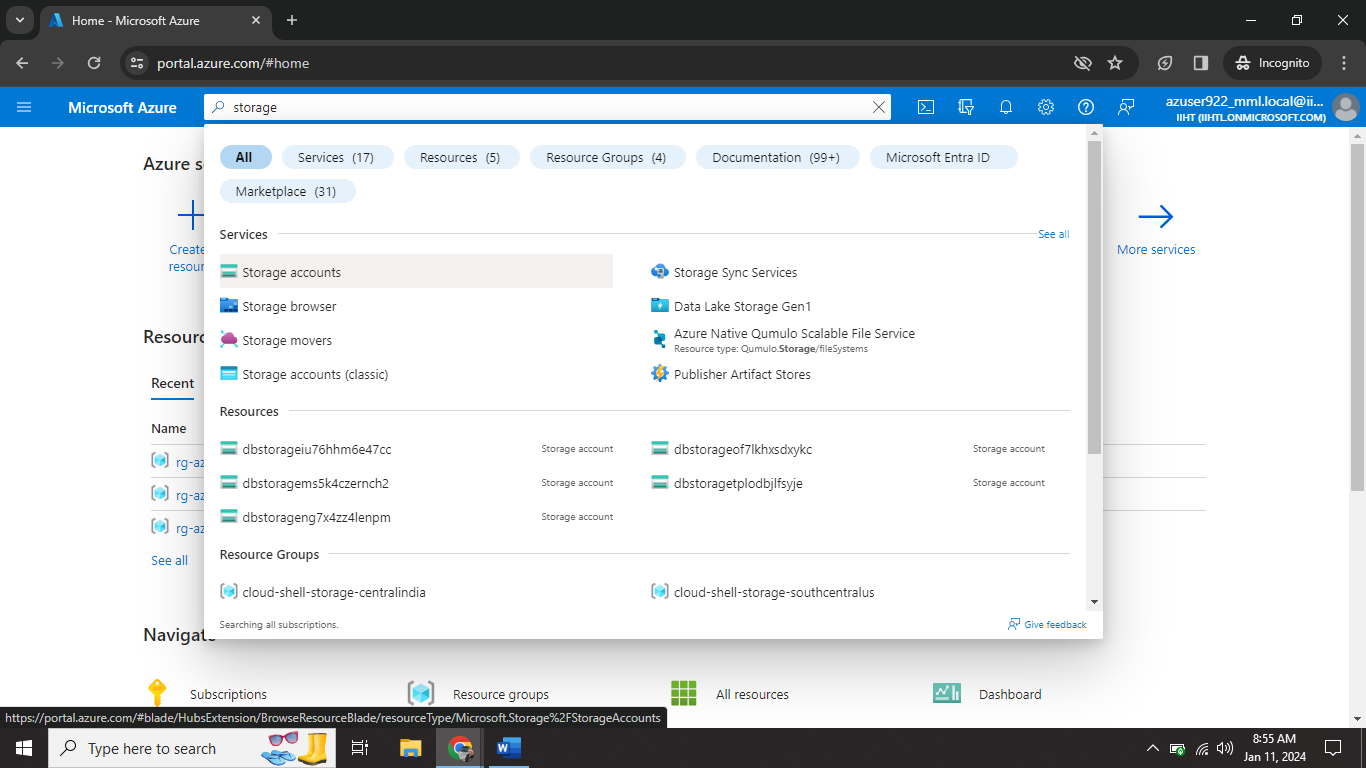
**6)Project Implementation**

**6.1 Azure Account**

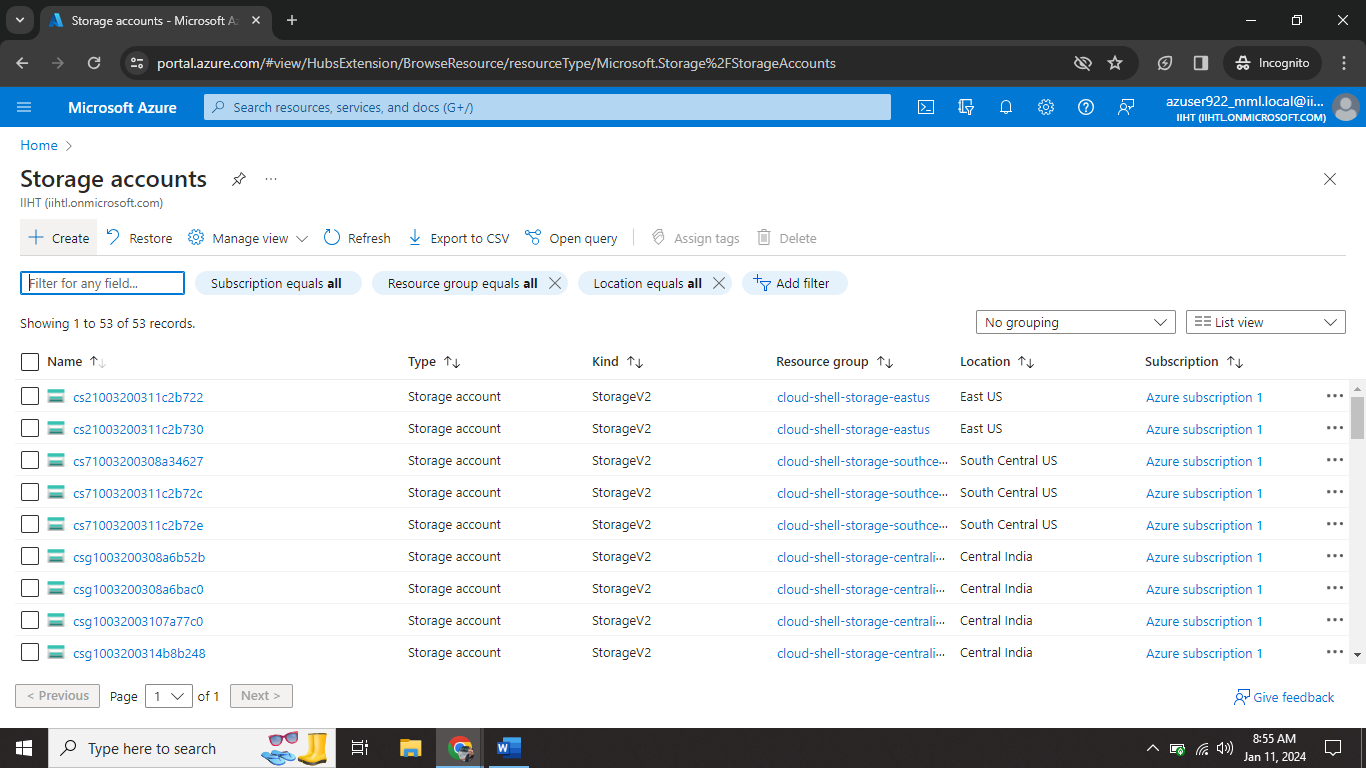
* Log in into Azure Subscription Account



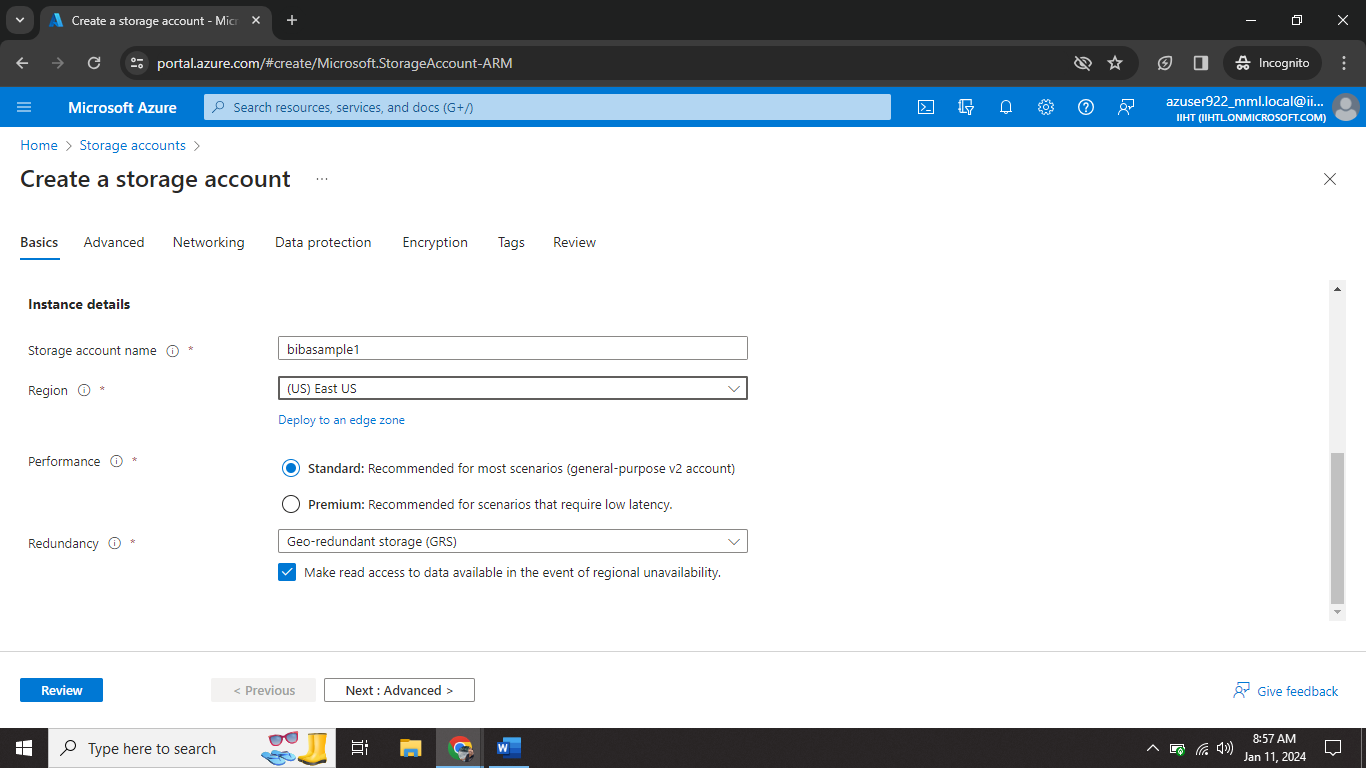
**6.2 Creating Azure Blob Account**



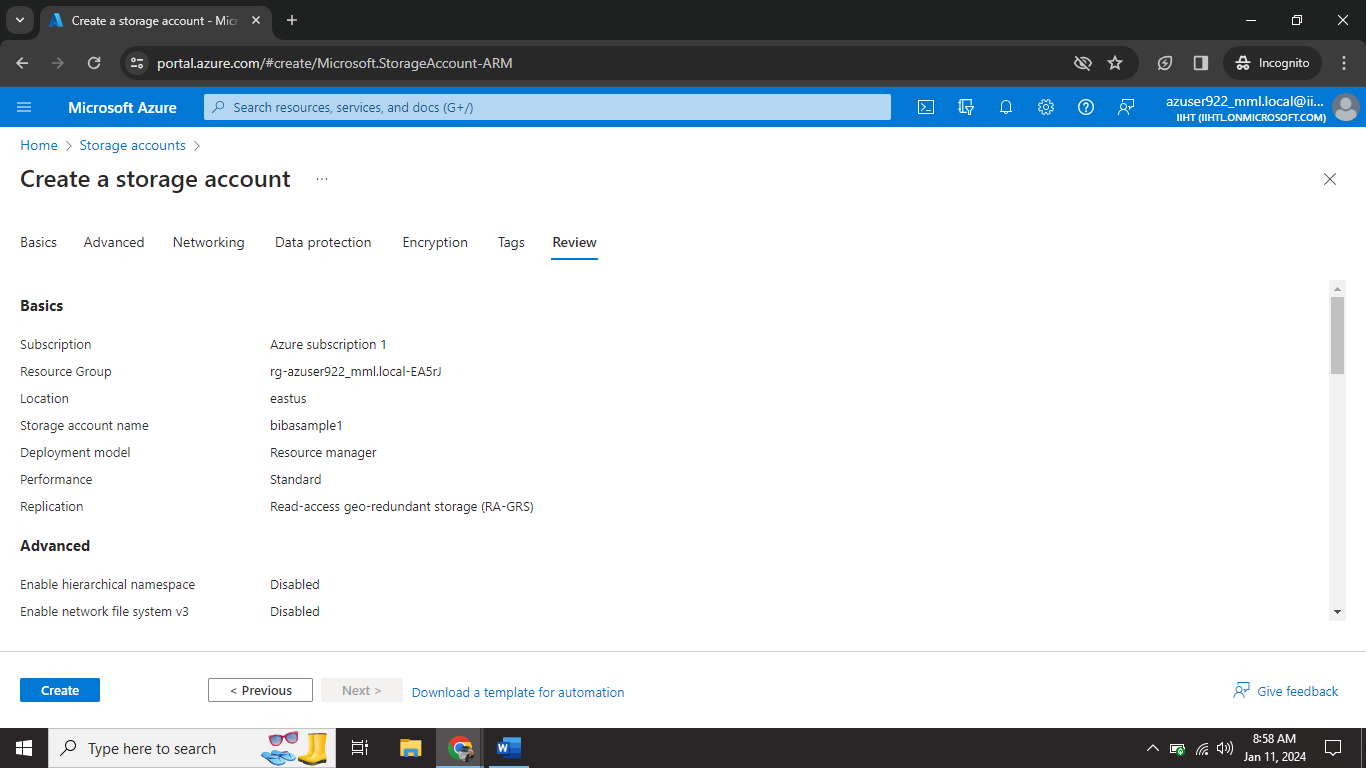
Step1🡺 Go to search bar. Search **Storage Account**.



Step2🡺 Click on **create** option.

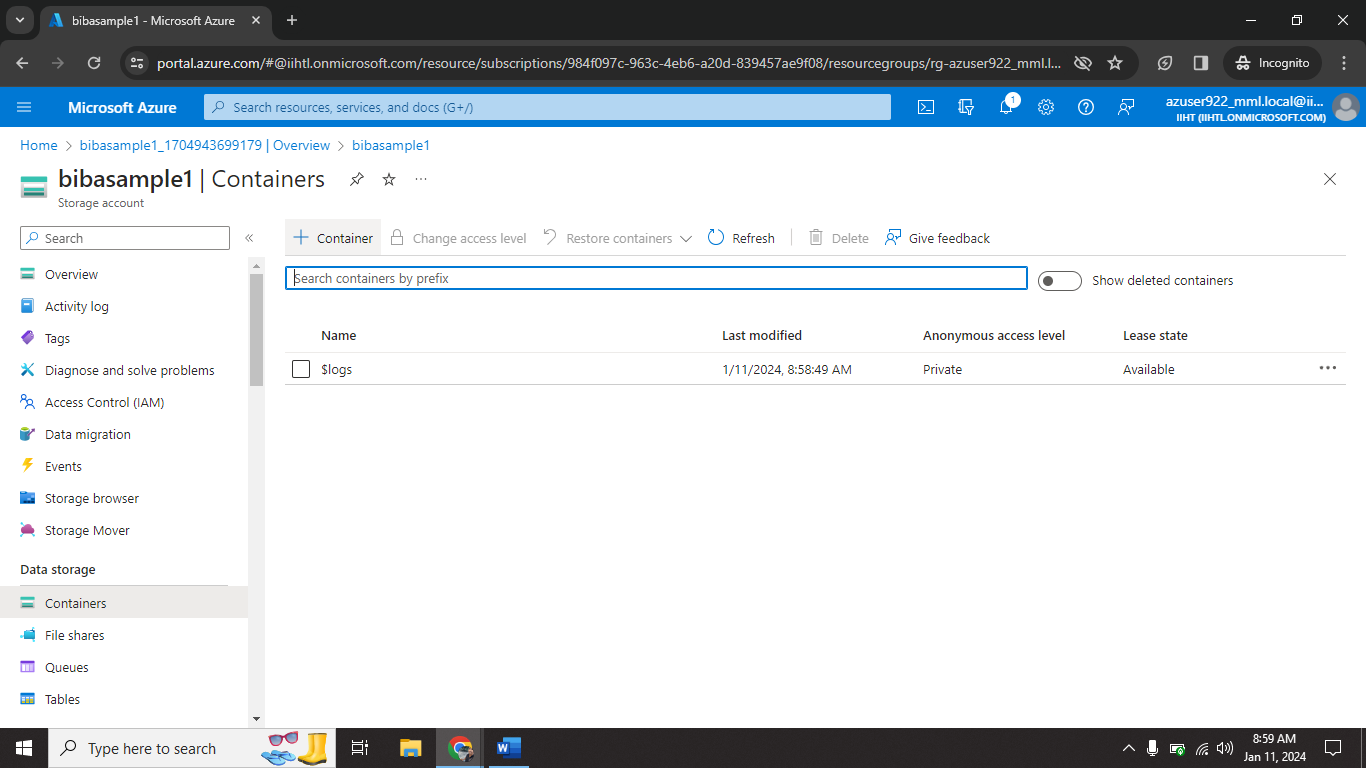


Step3🡺 Give storage account name (**bibasample1**).



Step4🡺 After review the details click on **create**.

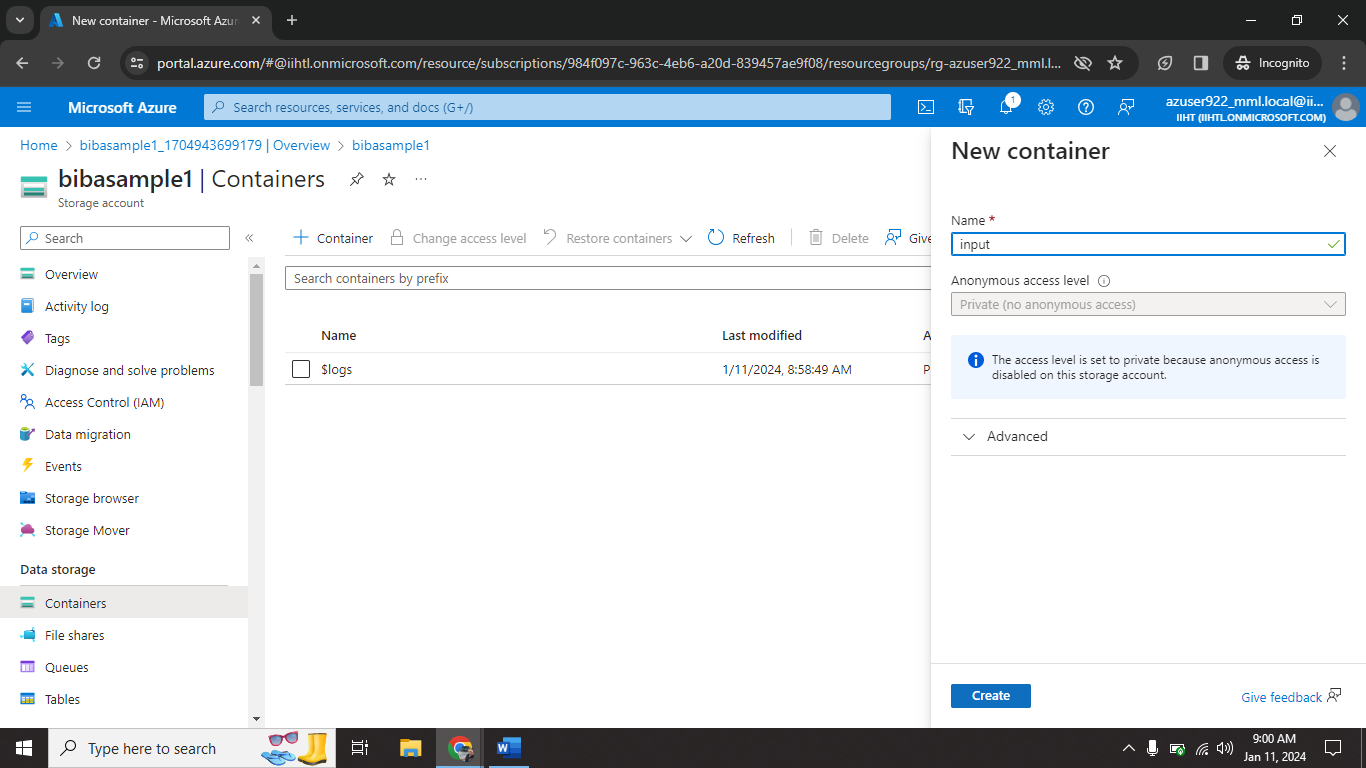
**6.3 Uploading Log file in Blob Account**



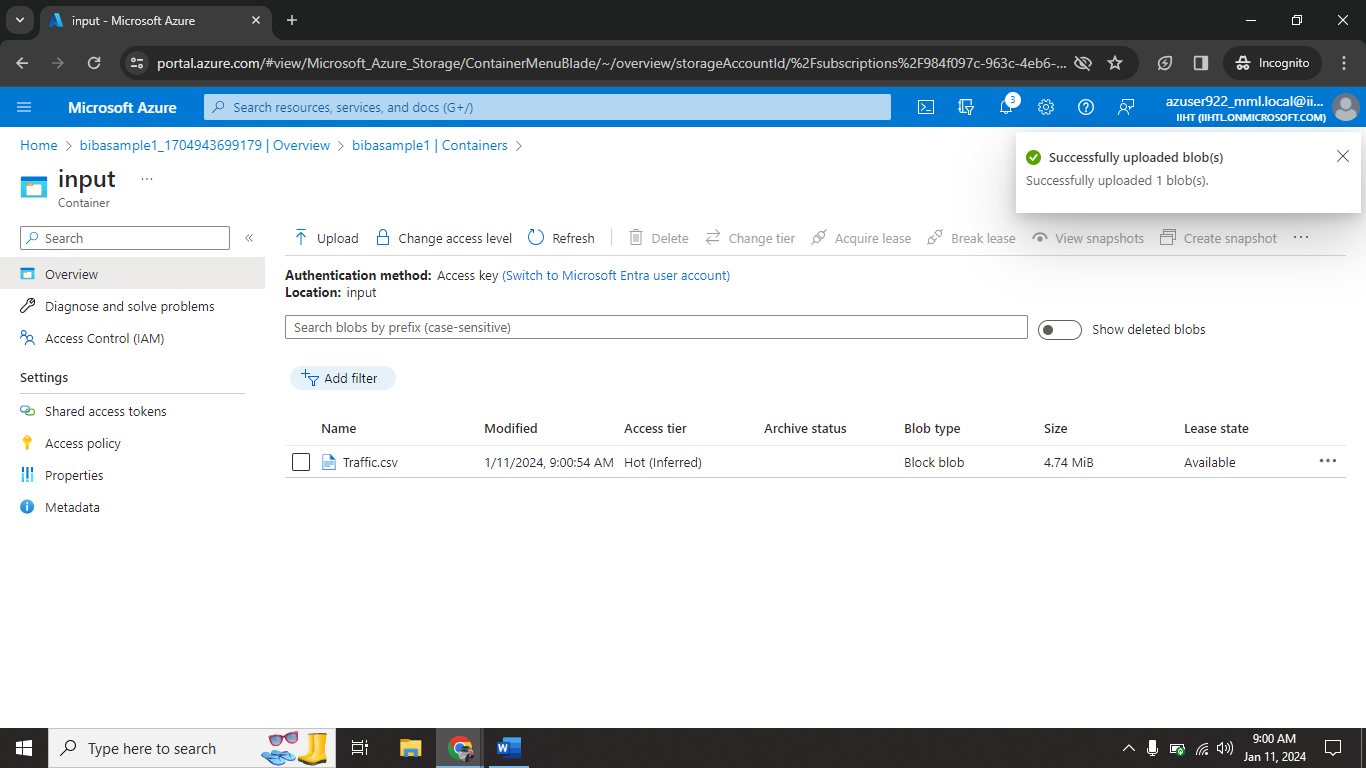
Step1🡺 Go inside the Blob Account.

Step2🡺 Click on Container.

Step3🡺 Click on **create**.

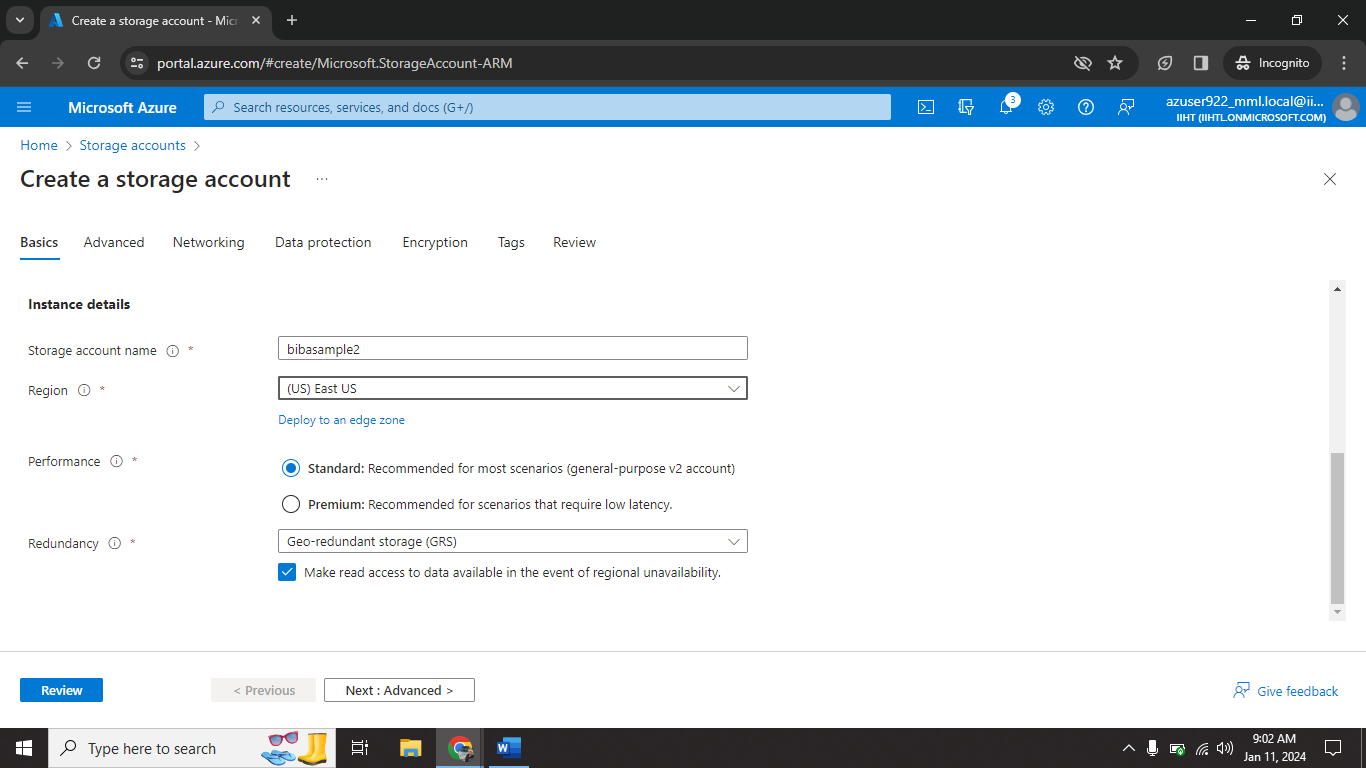


Step4🡺 Give container name (**input**) and click on **create.**



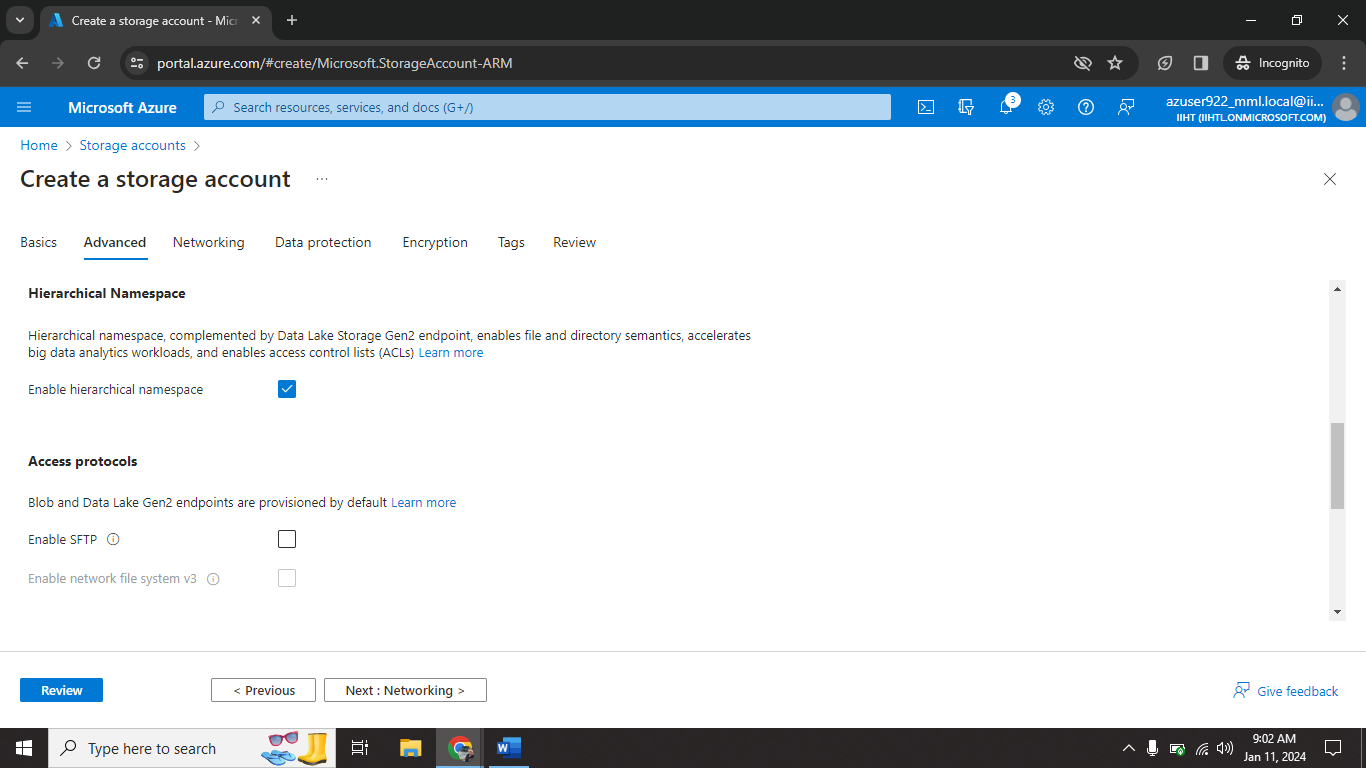
Step5🡺 Upload log file (**Traffic.csv**) from local storage. (Log file already downloaded from public git repository)

**6.4 Creating Azure Data Lake Storage Gen 2**

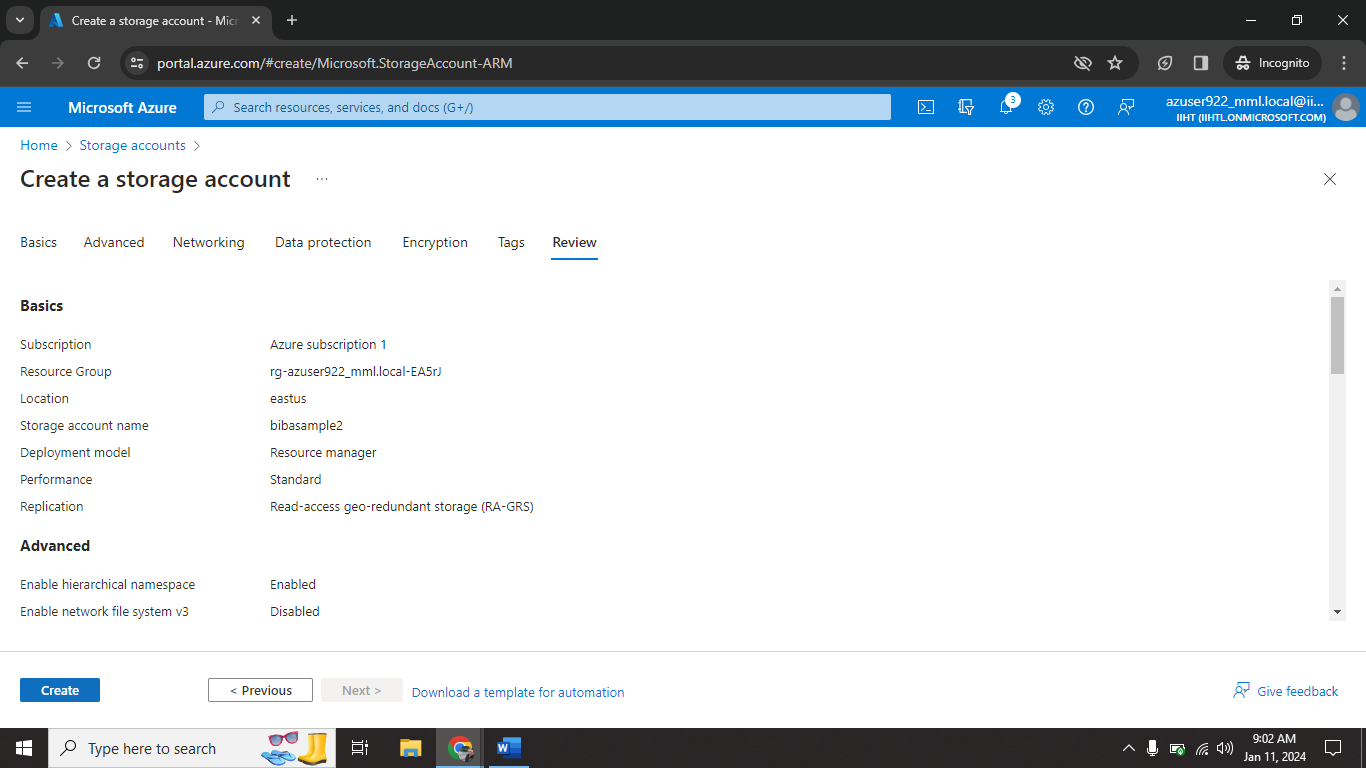


Step1🡺 Go to storage account click on create.

Step2🡺 Give name for ADLS (**bibasample2**)

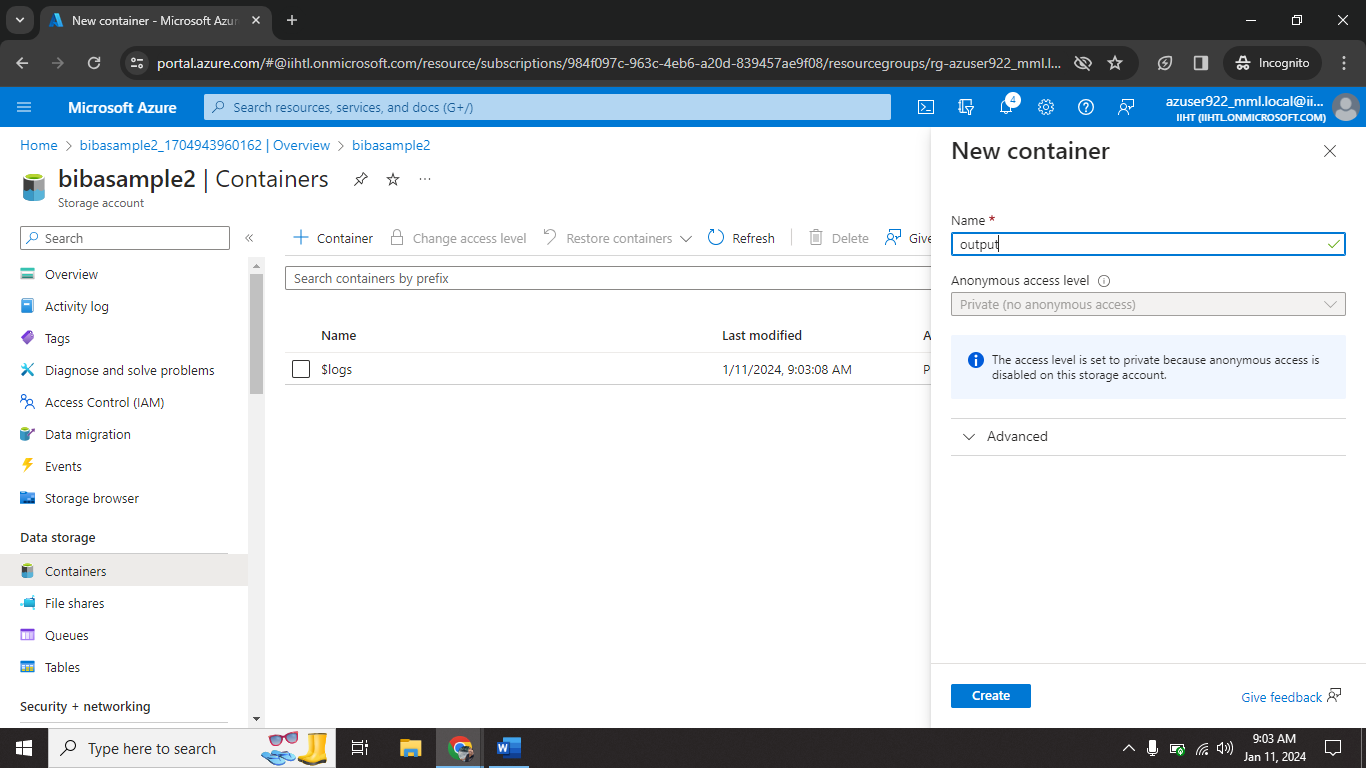


Step3🡺 click on **Hierarchical Namespace** on advanced tab. (Important step)



Step4🡺 After review the details click on **create**.

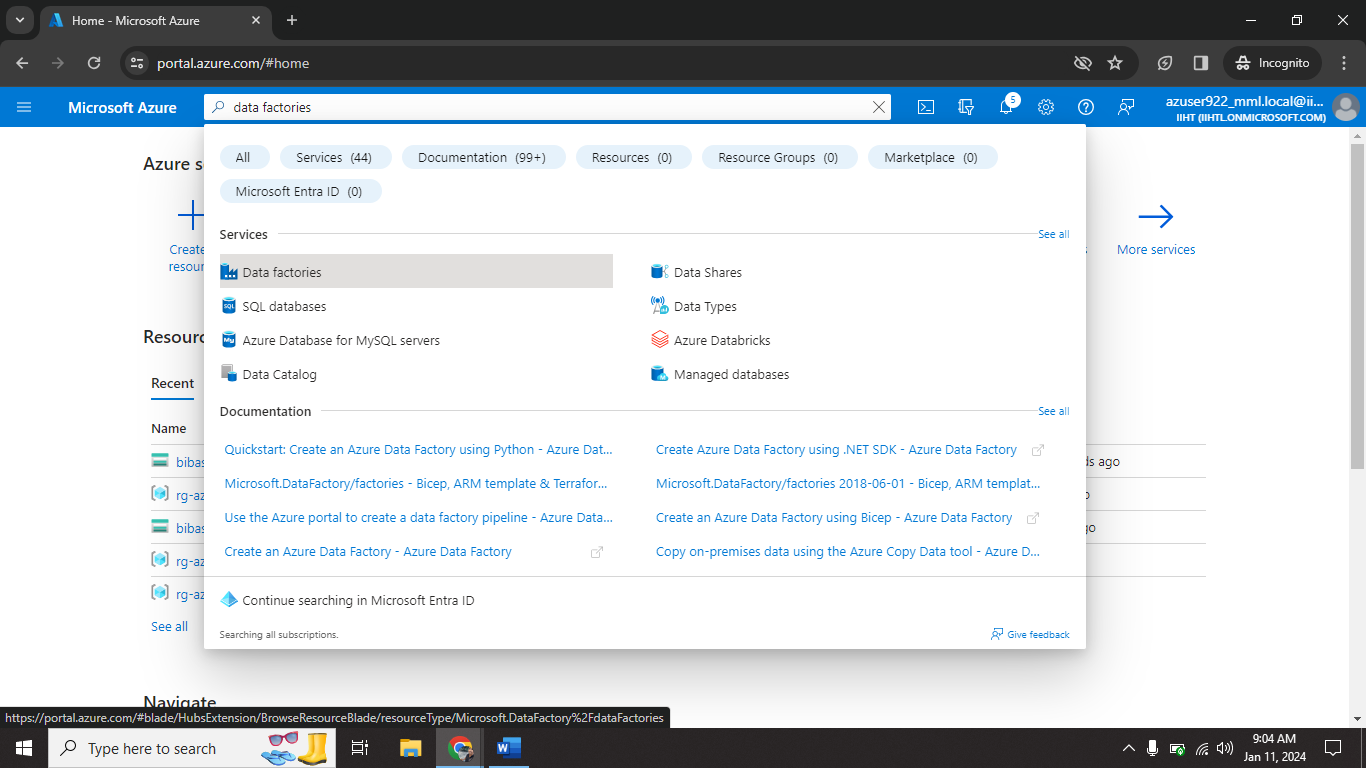
**6.5 Creating Container on ADLS**



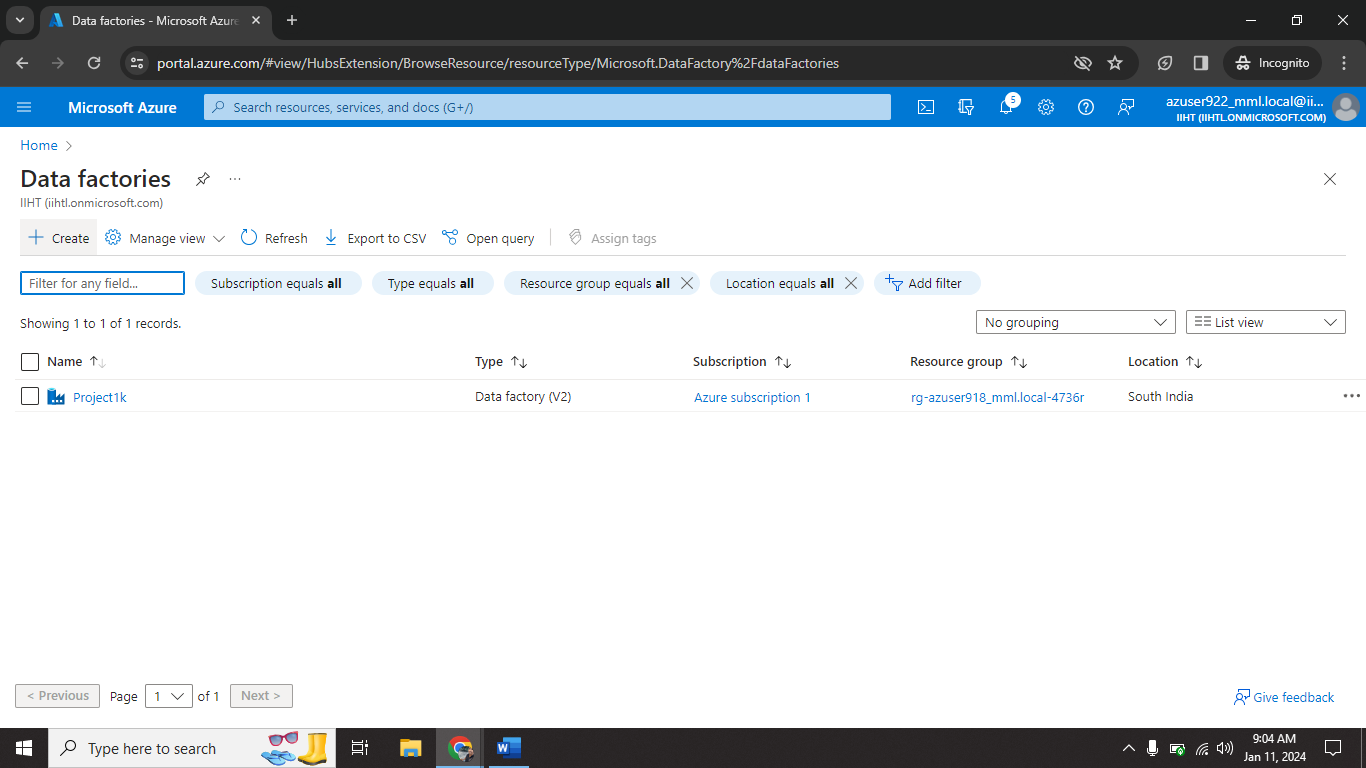
Step1🡺 Go to ADLS

Step2🡺 click on Containers and create new container (**output**) for receive log file.

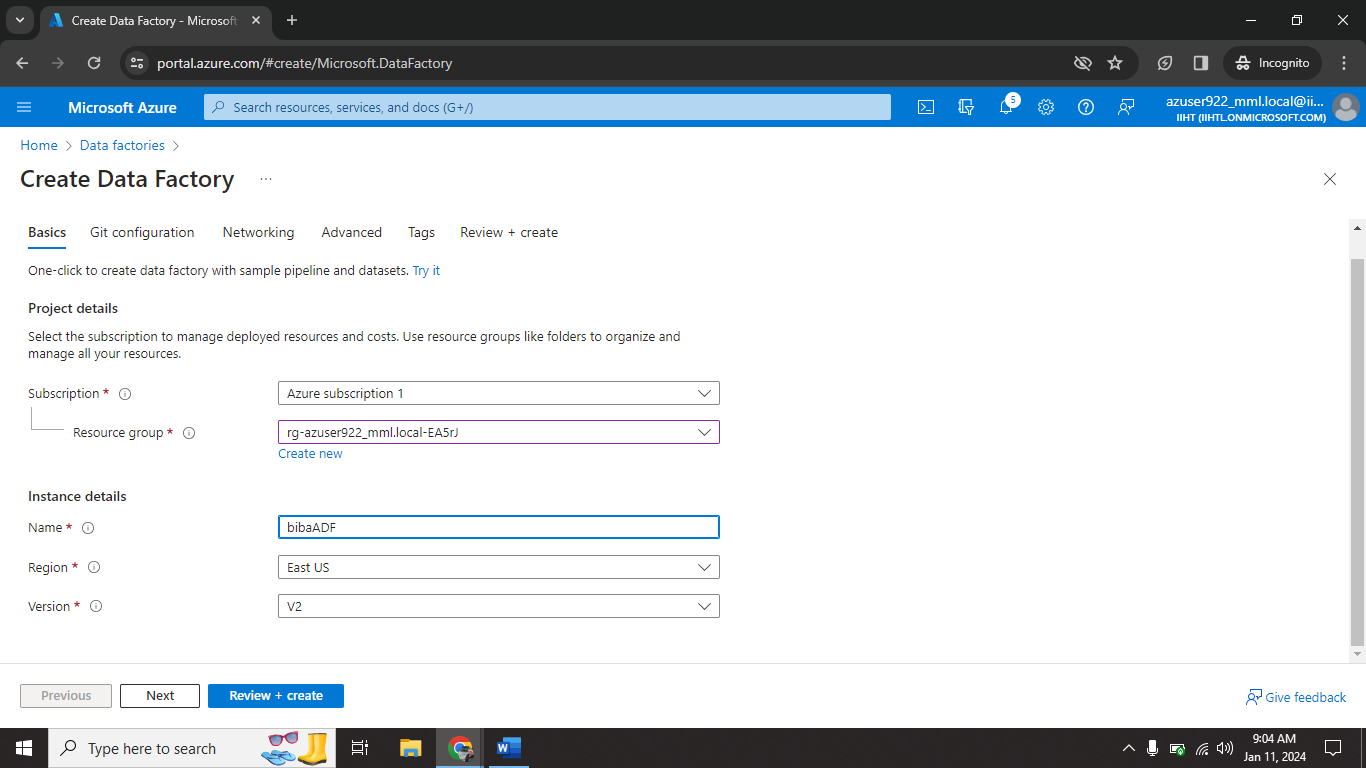
**6.6 Azure Data Factory Configuration**



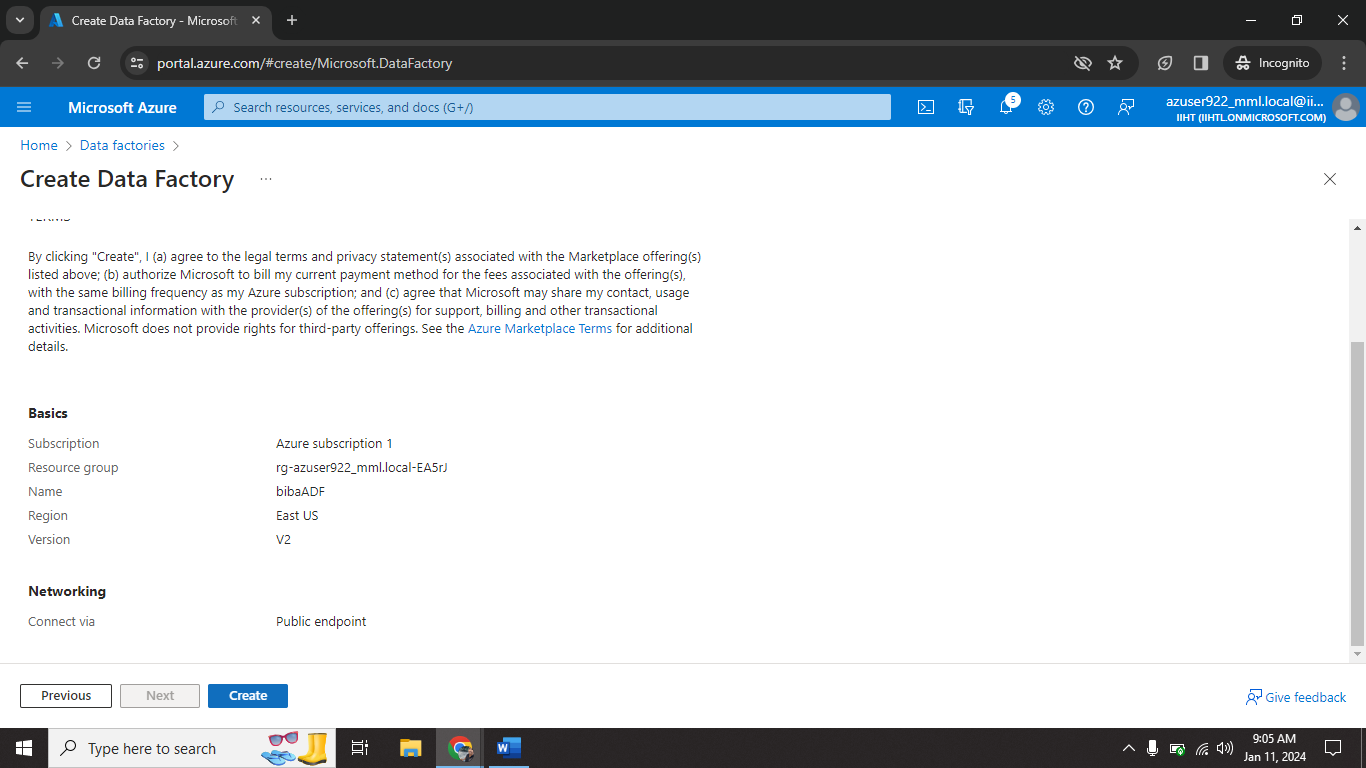
Step1🡺 Go to search bar type **data factory.**



Step2🡺 click on **create**.



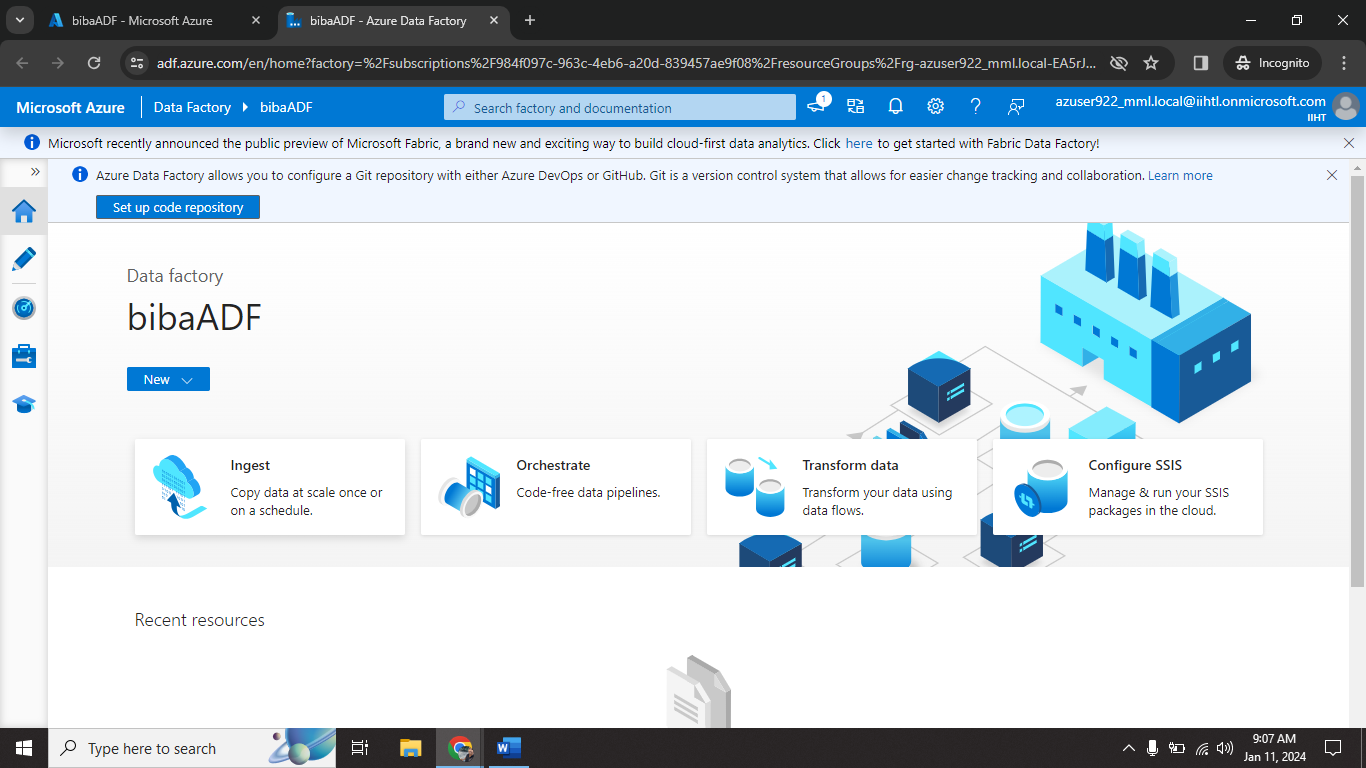
Step3🡺 Give ADF name (**bibaADF**)



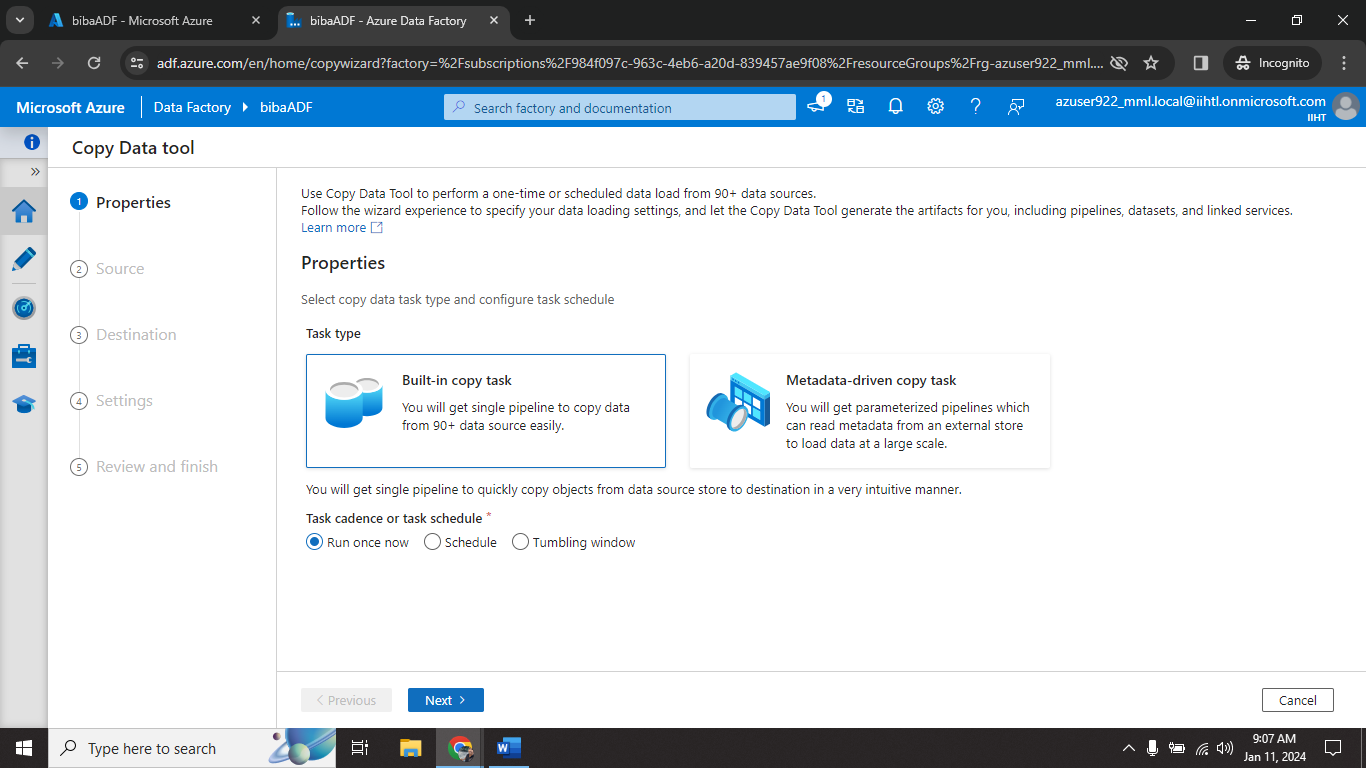
Step4🡺 After review all the details click on **create.**



Step5🡺 click on **Launch studio.**



Step6🡺 click on **Ingest** option.

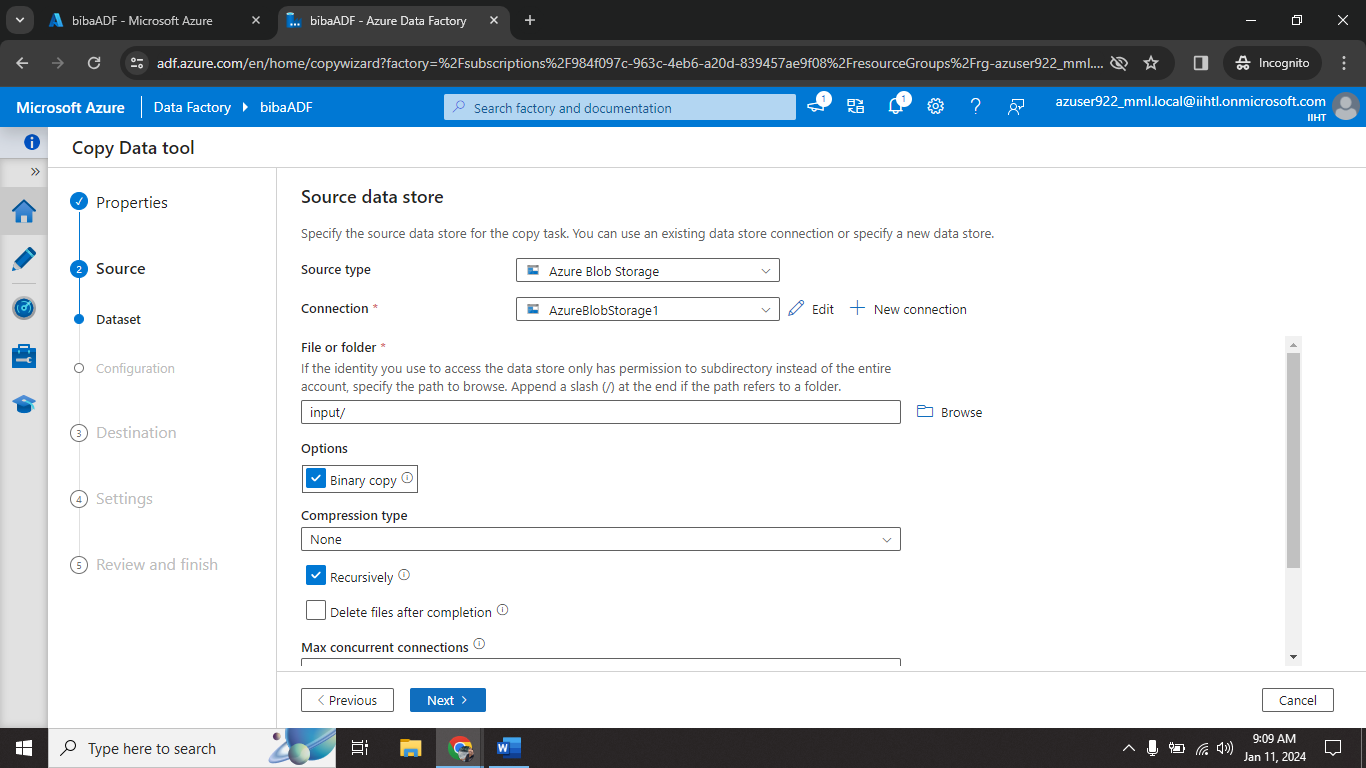


Step7🡺 click on **Built-in-copy task.**

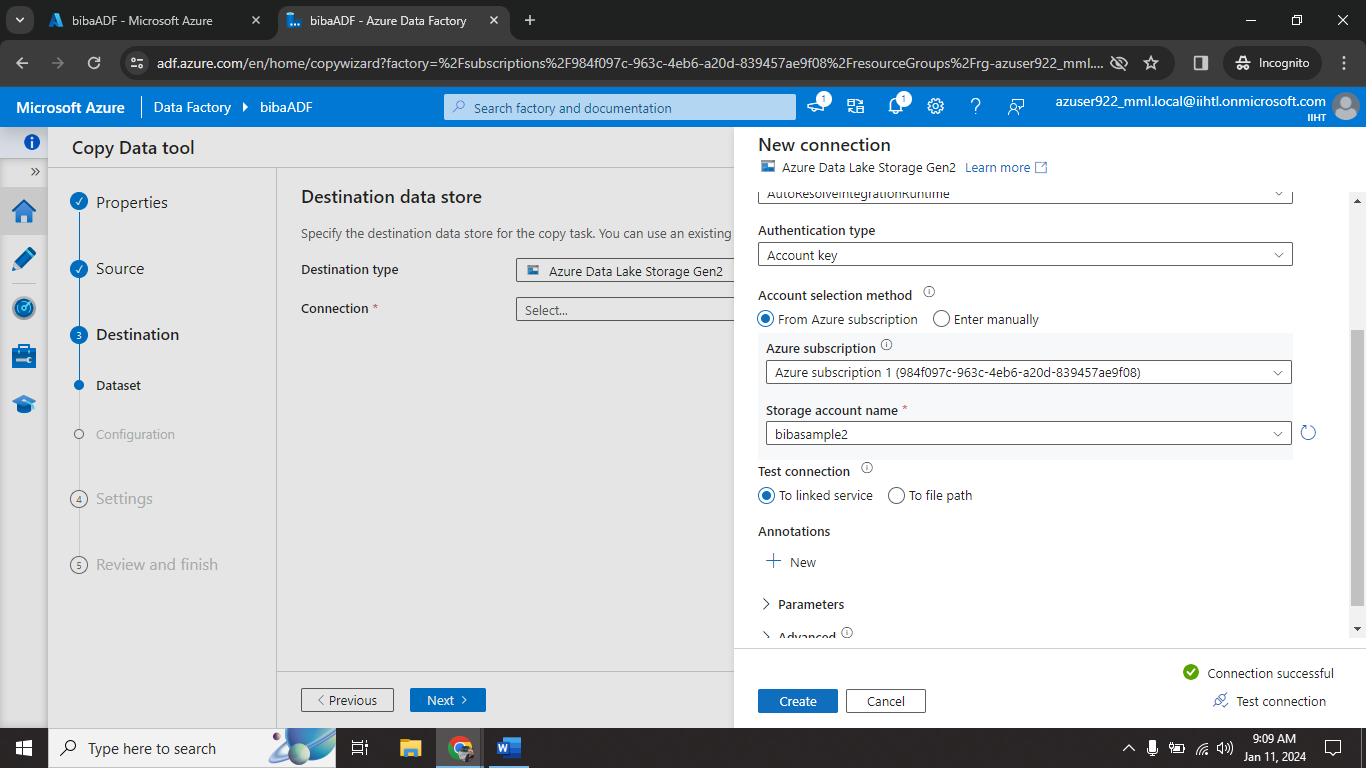


Step8🡺 click on **New connection.** Fill all information(blob storage details) and click on **Test connection.**

Step9🡺 After connection successfully click on **create.**

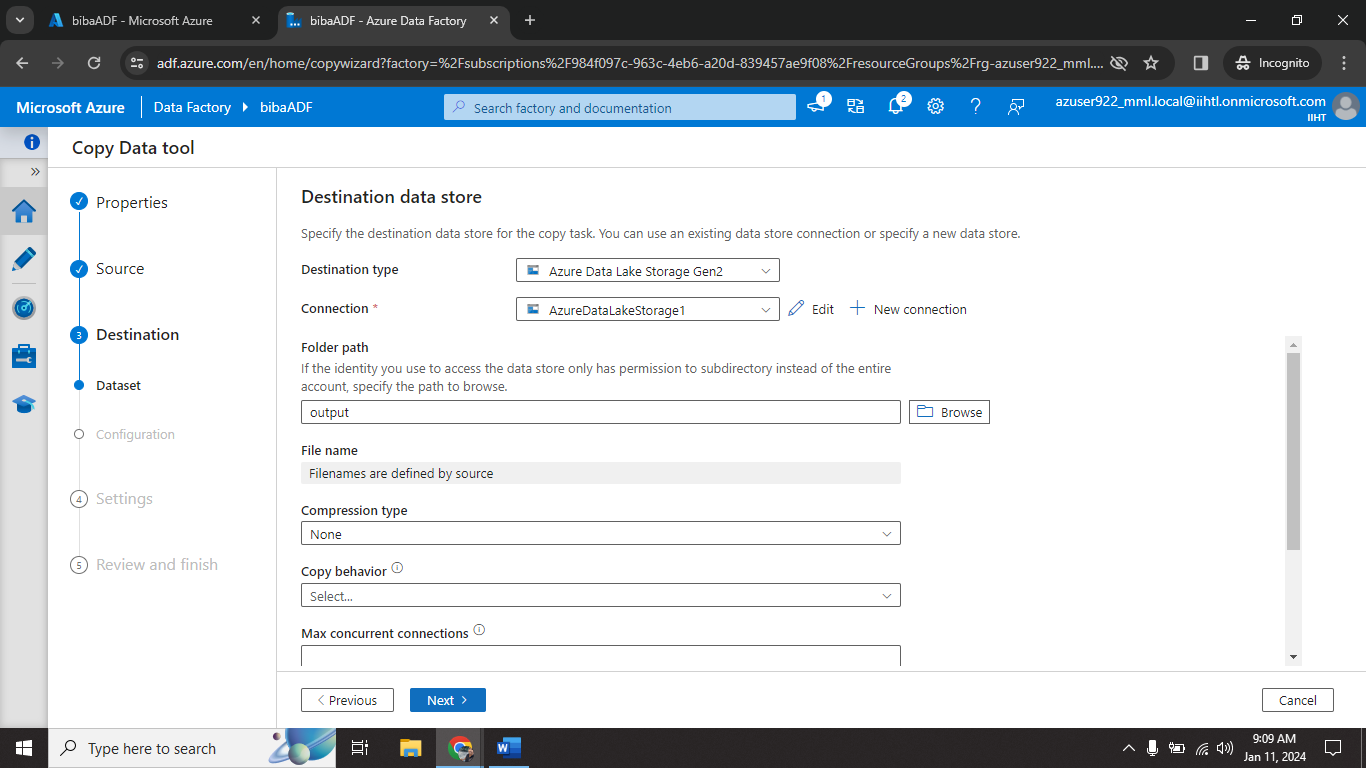


Step10🡺 Browse log file location. Click on **Binary code**. Click **Next.**

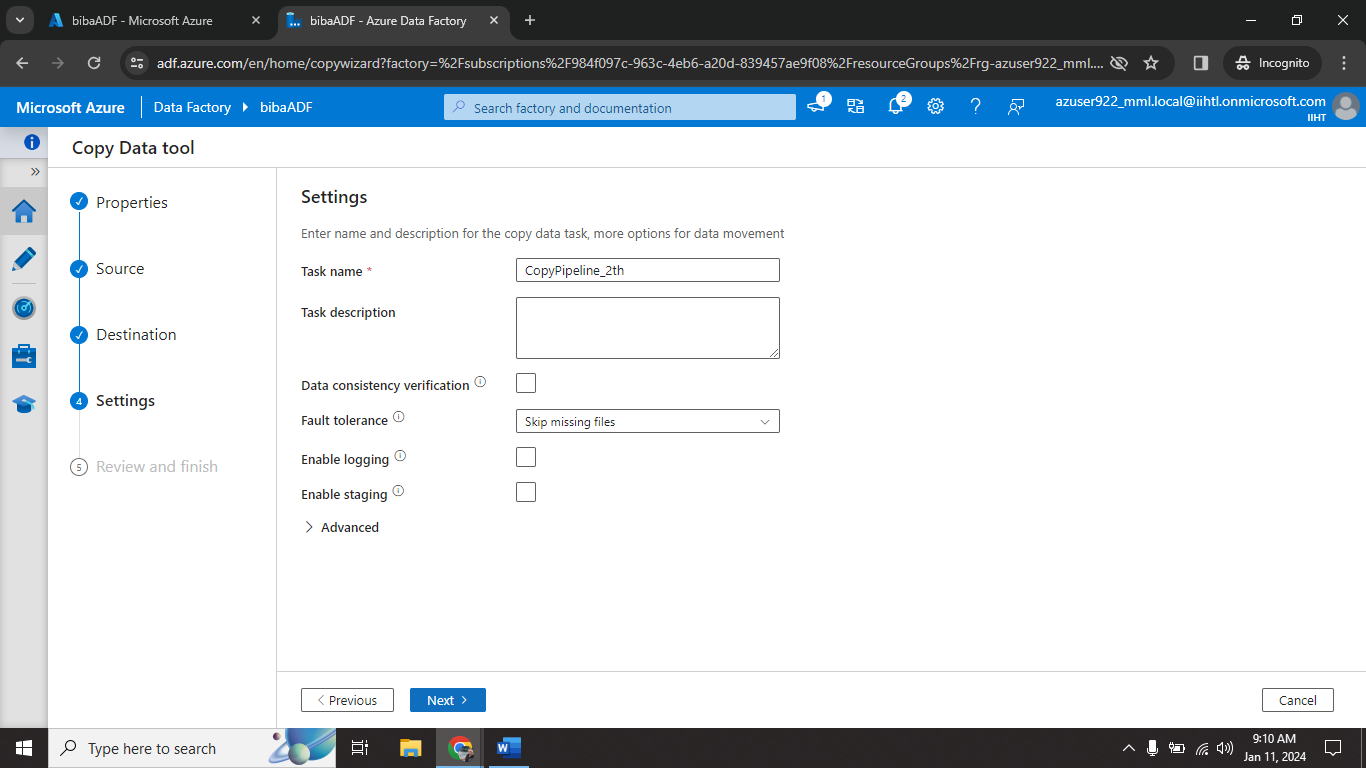


Step11🡺 click on New connection. Fill all information (ADLS) and click on **Test connection.**

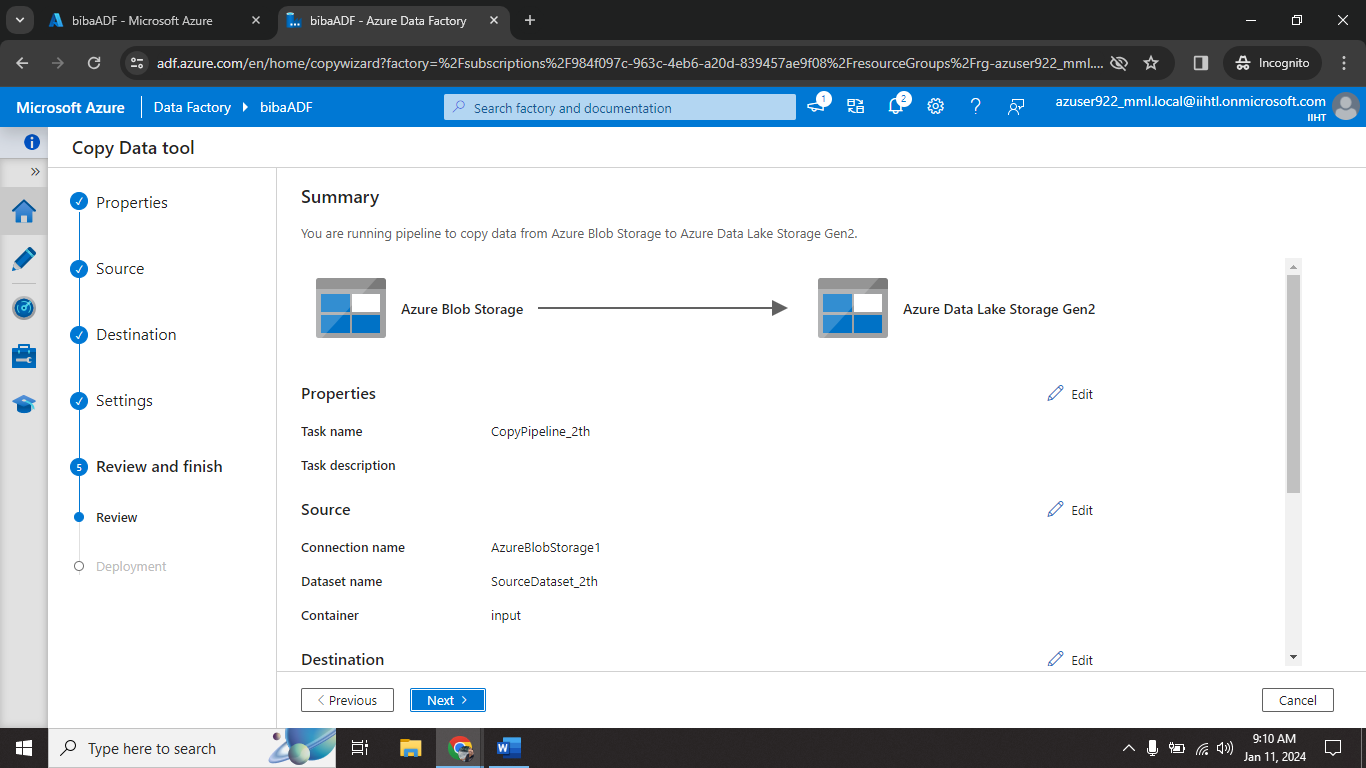
Step12🡺 After connection successful click on **create.**



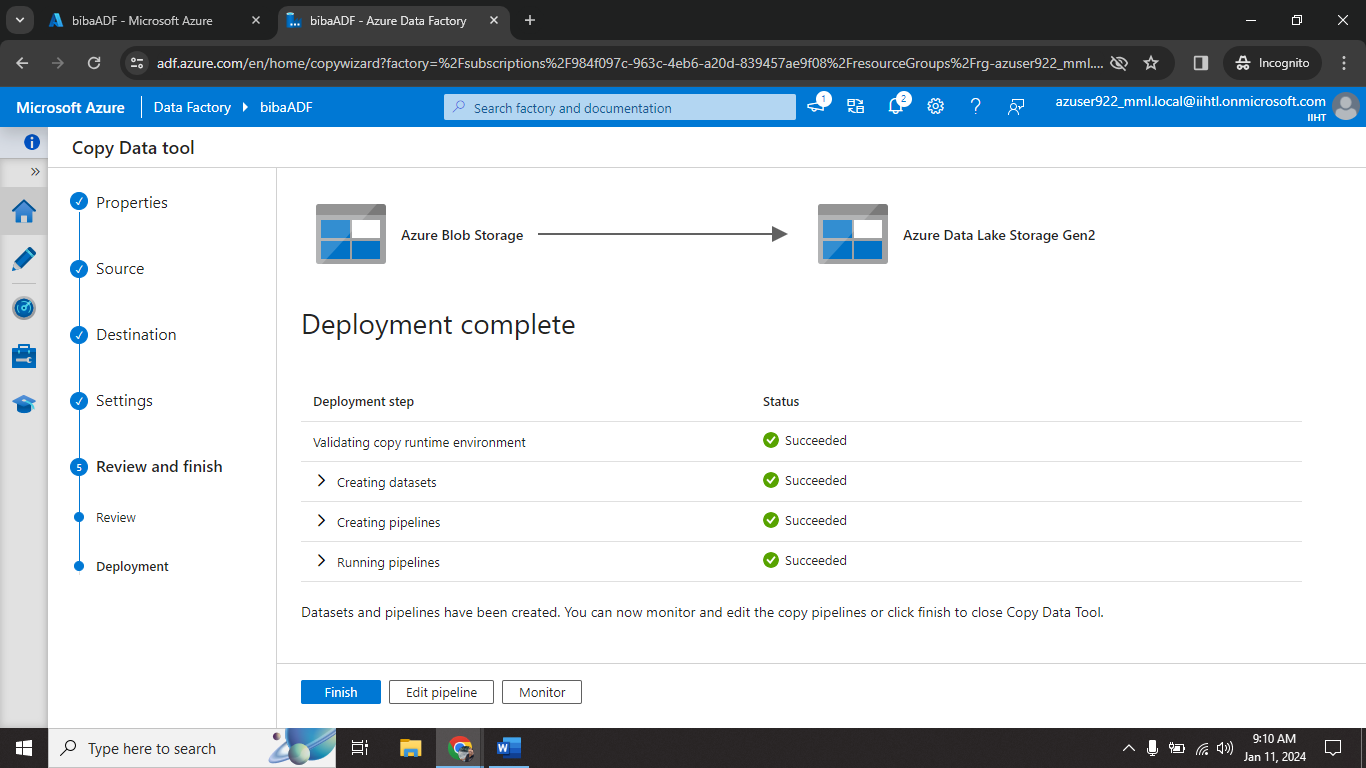
Step13🡺 Browse destination container (**output**). Click on **Next.**



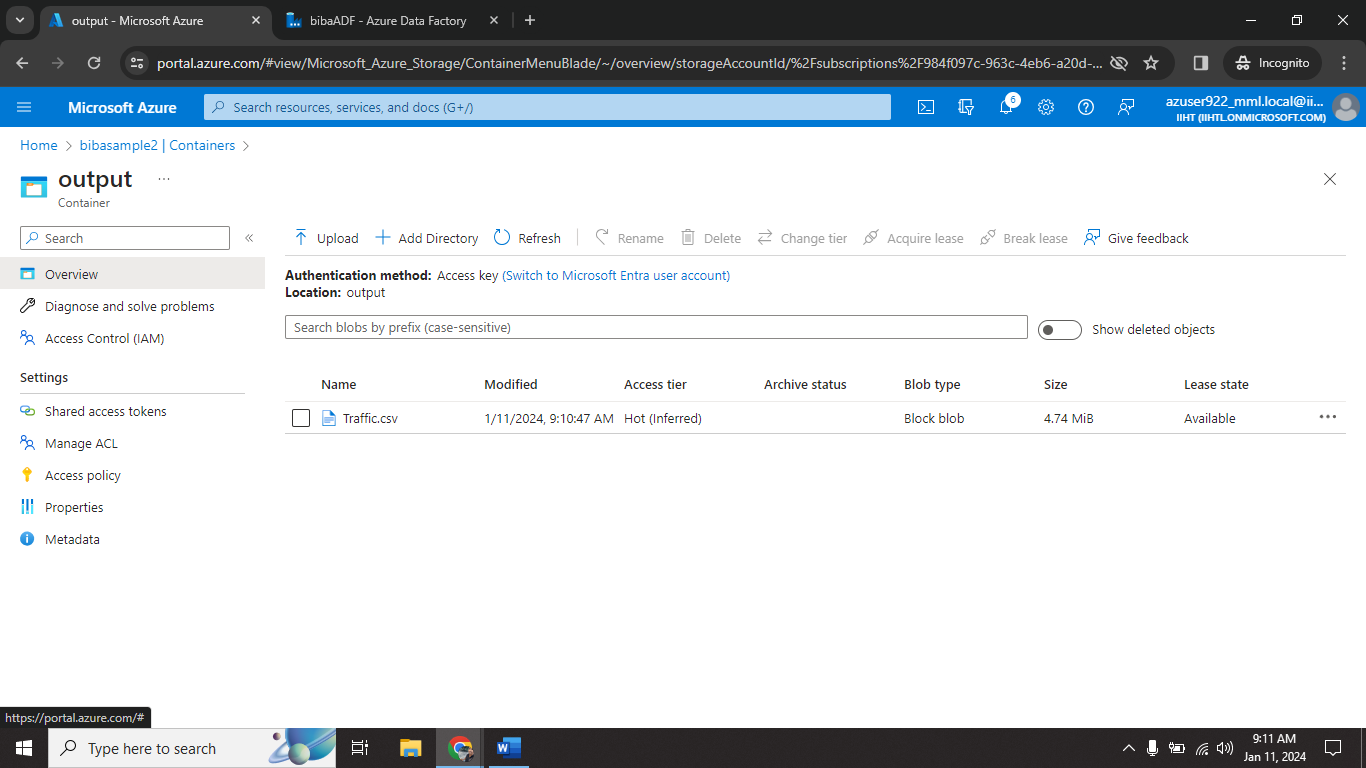
Step14🡺 Give Task name and click on **Next.**



Step15🡺 Review all details carefully. Click on **Next.**

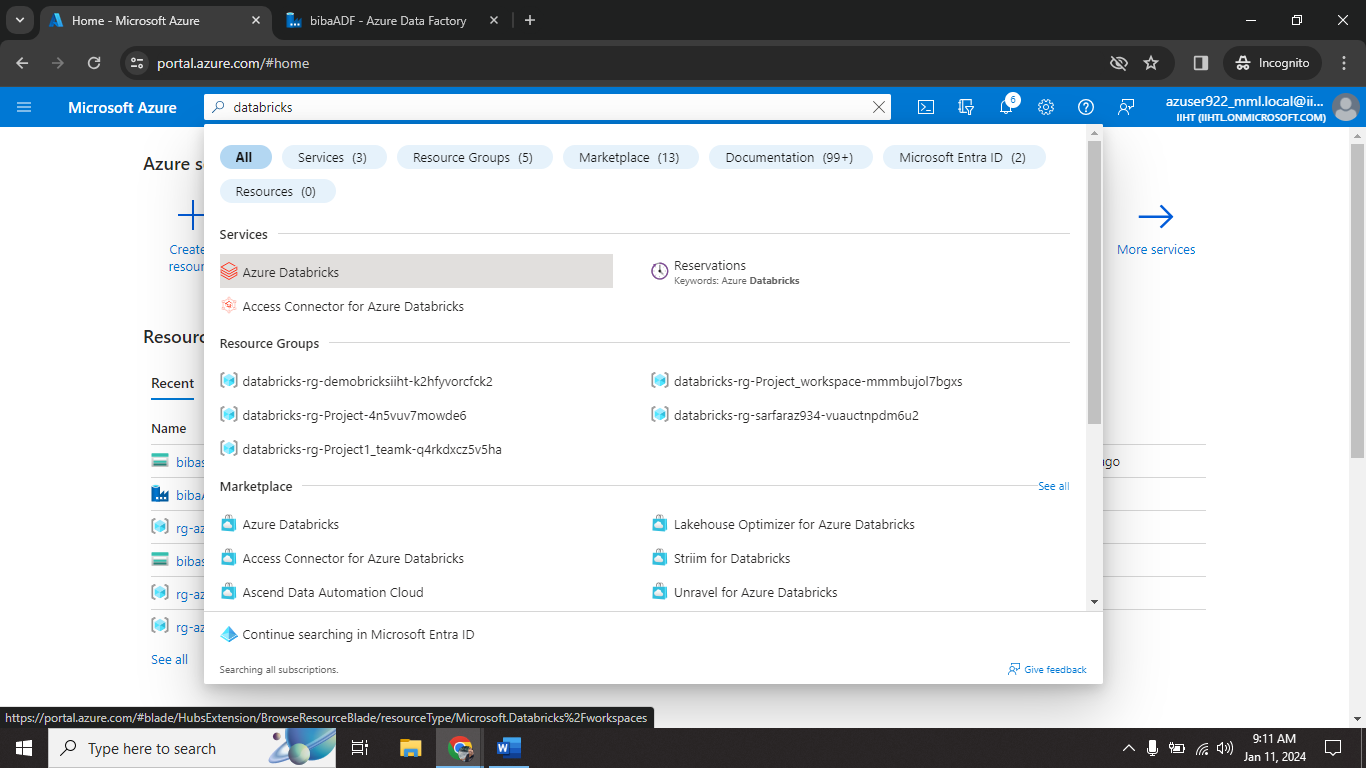


Step16🡺 Log file successfully transferred to ADLS. Click on **Finish.**



Step17🡺 Now you will see “Traffic.csv” file display on ADLS account.

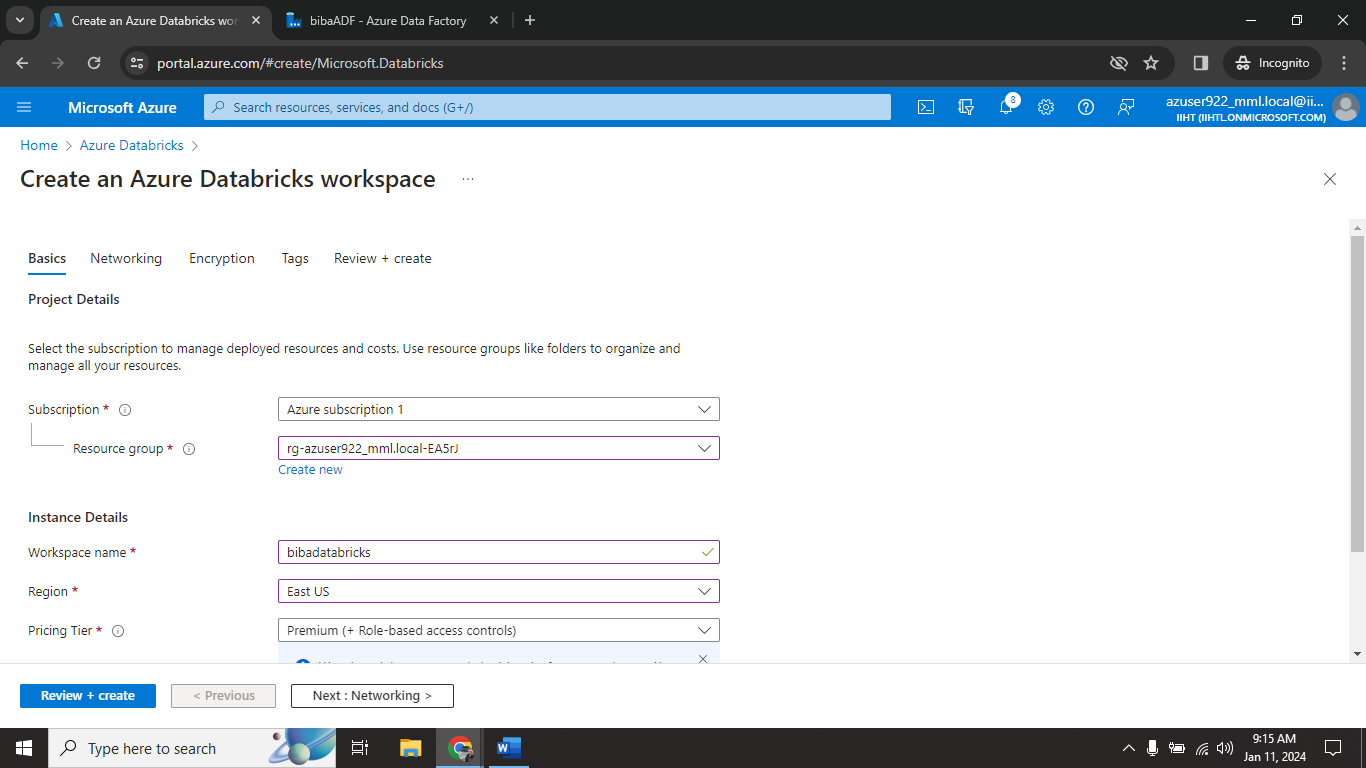
**6.7 Azure Databricks Configuration**



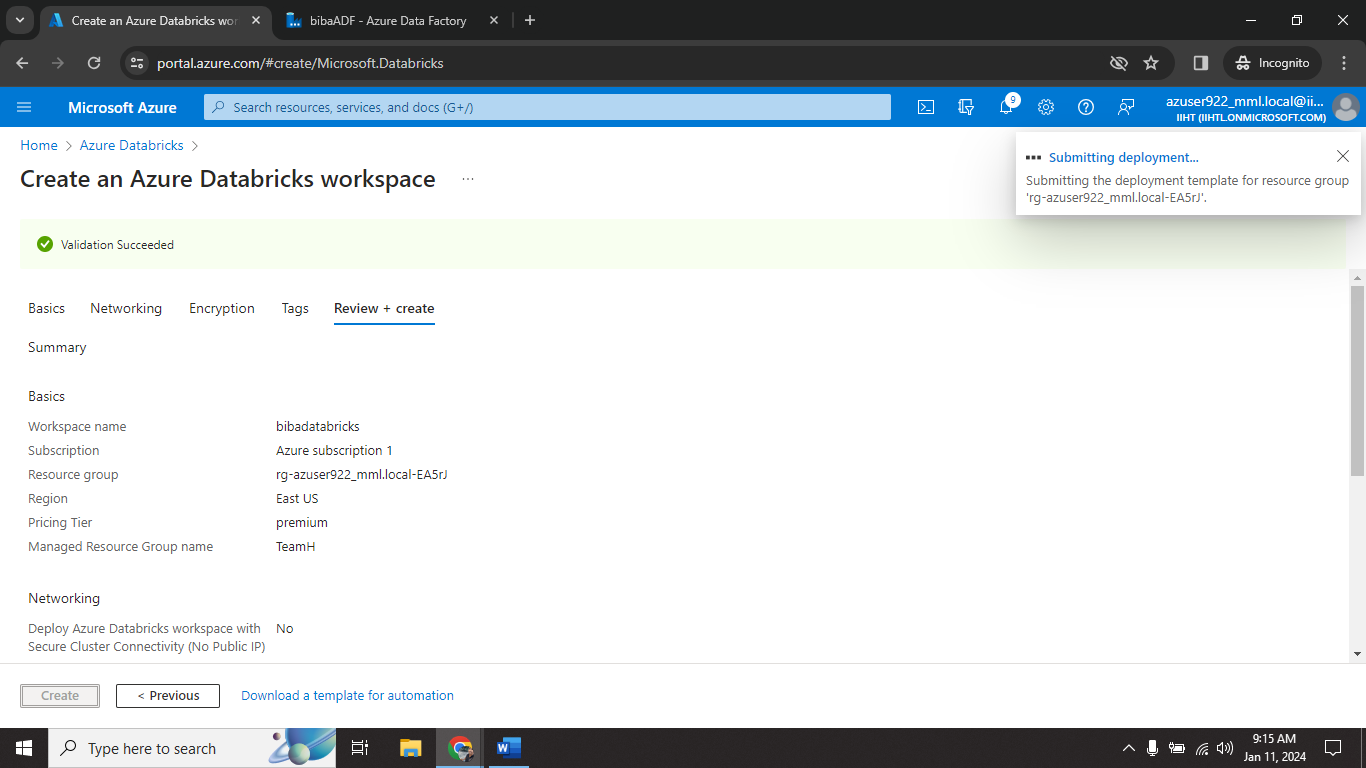
Step1🡺 go to search bar. Type databricks.



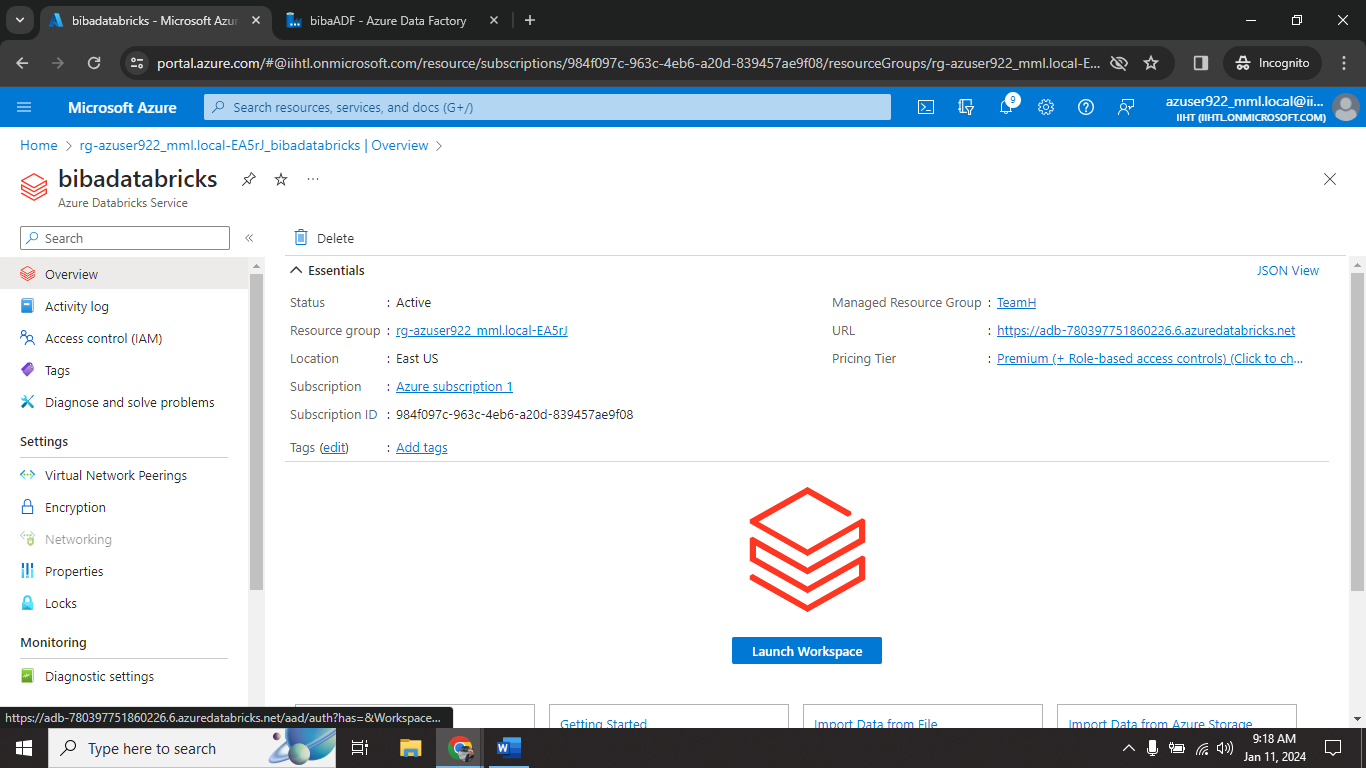
Step2🡺 click on **create**.



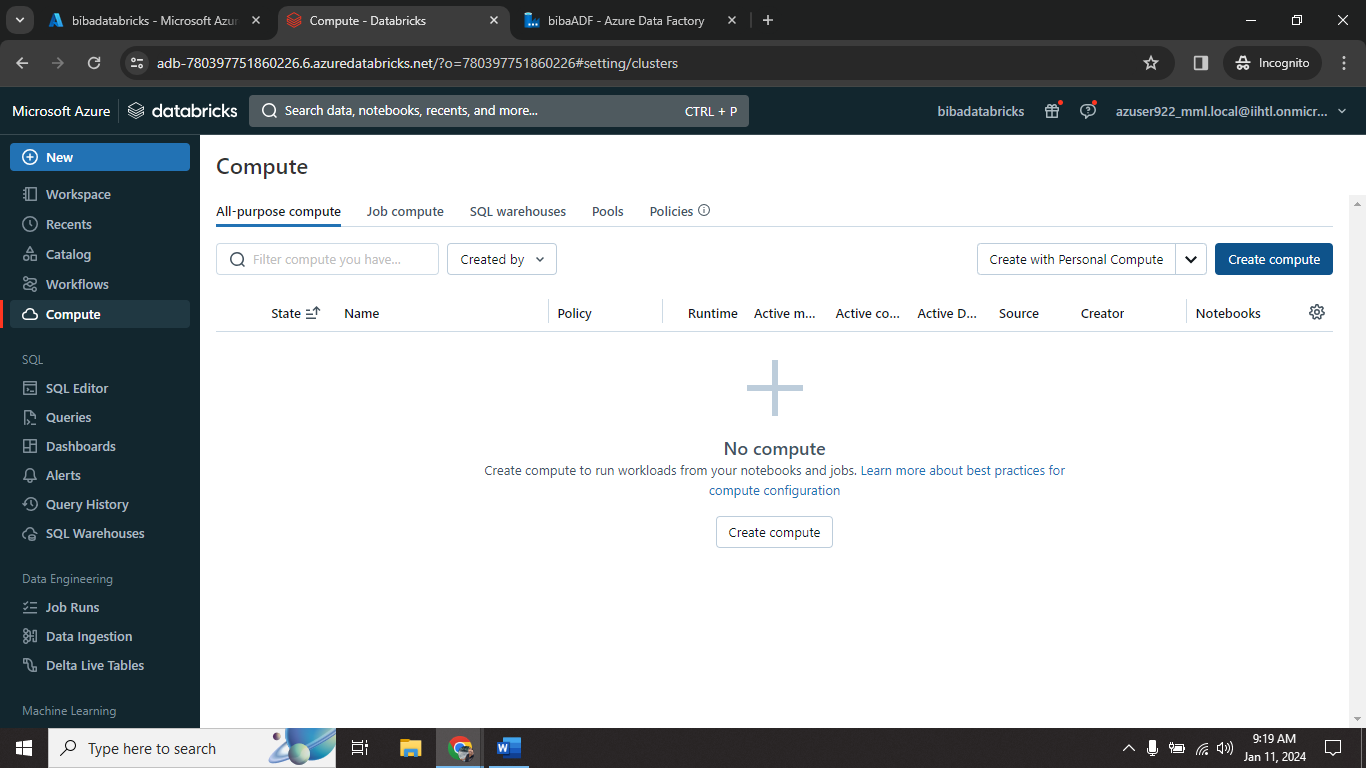
Step3🡺 give workspace name (**bibadatabricks**) and fill all requirements.



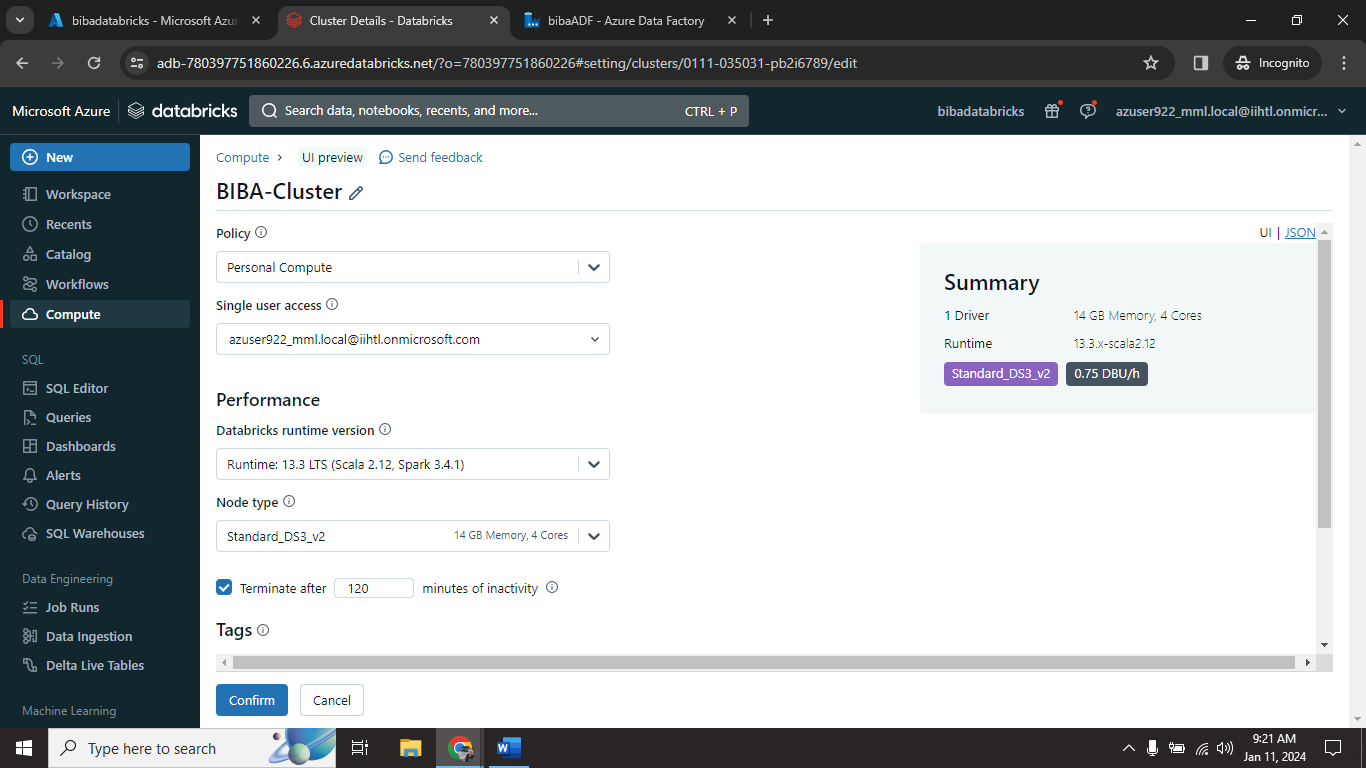
Step4🡺 click on **create**.



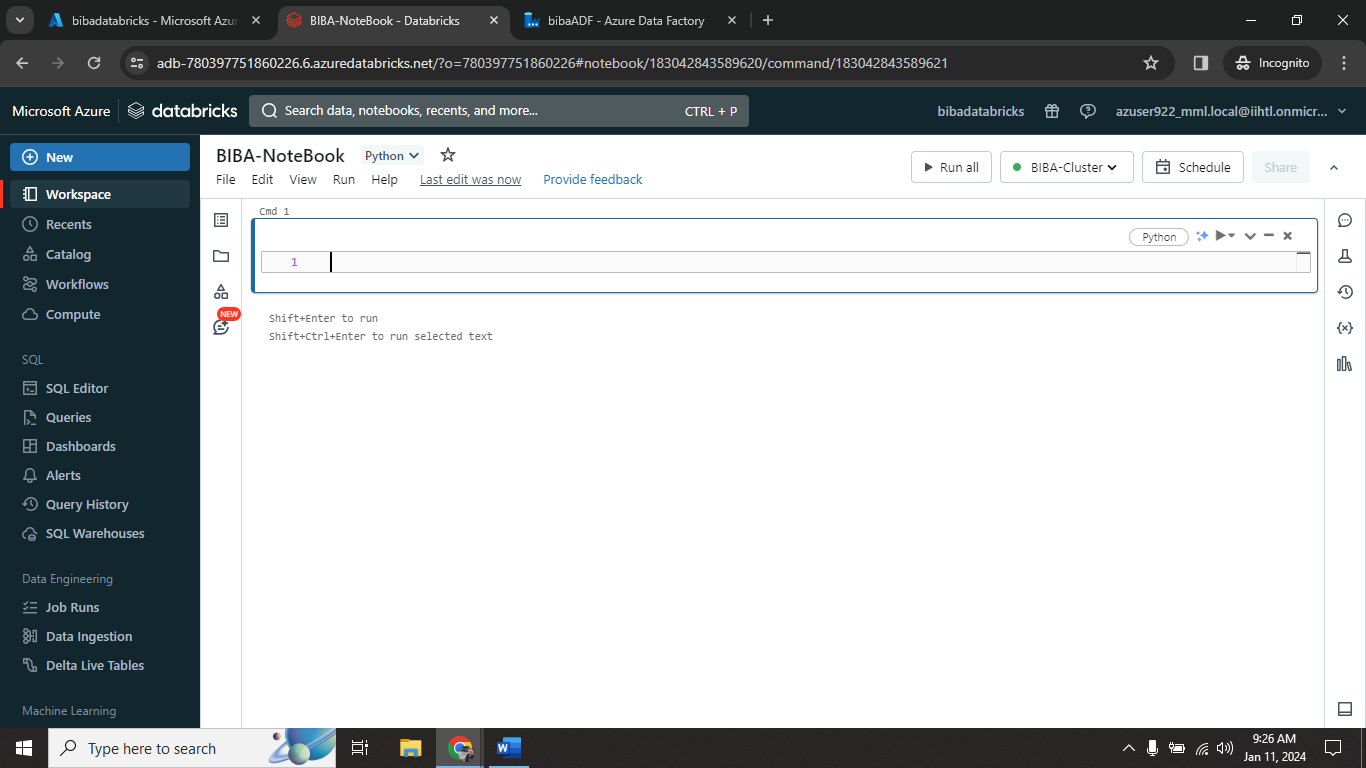
Step5🡺 click on **Launch Workspace**.



Step6🡺 click on **Compute.** Click on **create compute.**

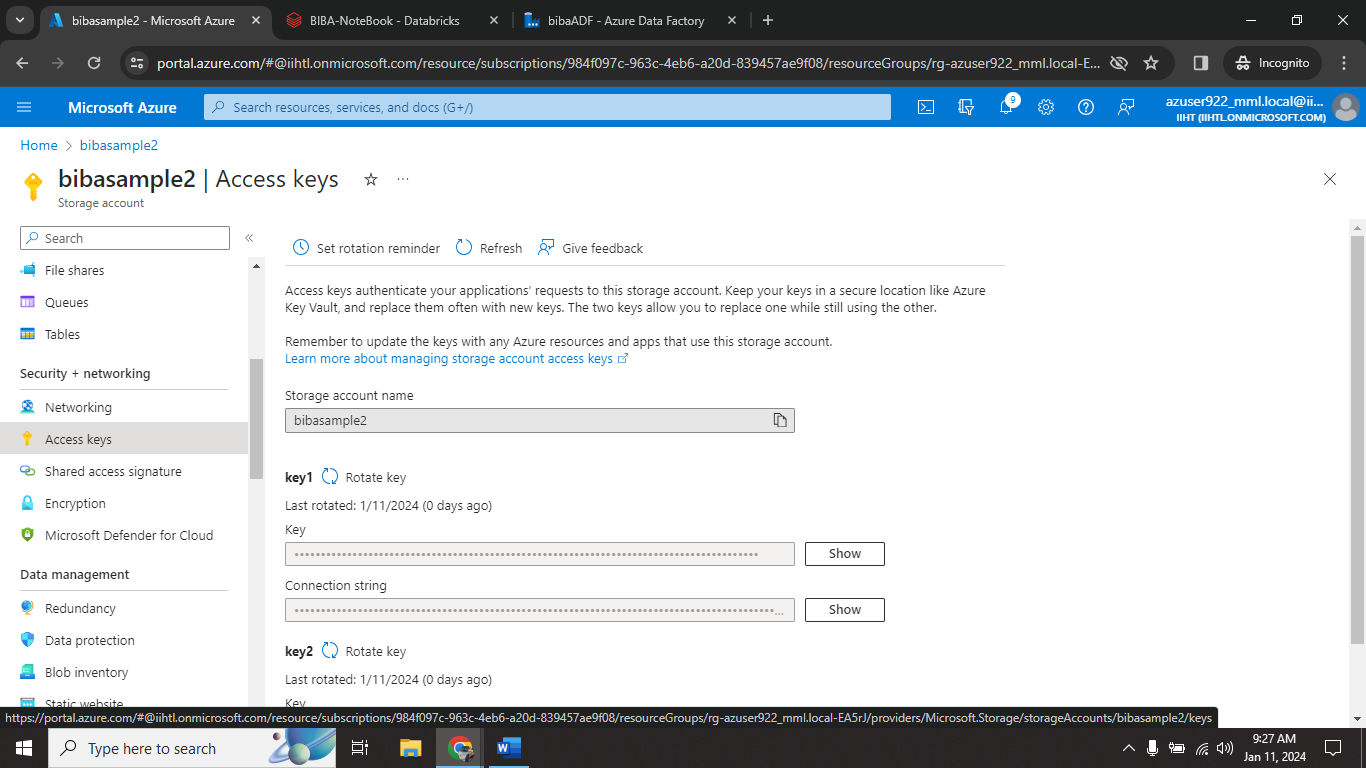


Step7🡺 give cluster name(**BIBA-Cluster**) and click on **create.**



Step8🡺 create NoteBook (**BIBA-NoteBook**)

**6.8 Connecting ADLS with Databricks**



Step1🡺 go to ADLS account and click on **Access Keys**. Copy key 1’s key.

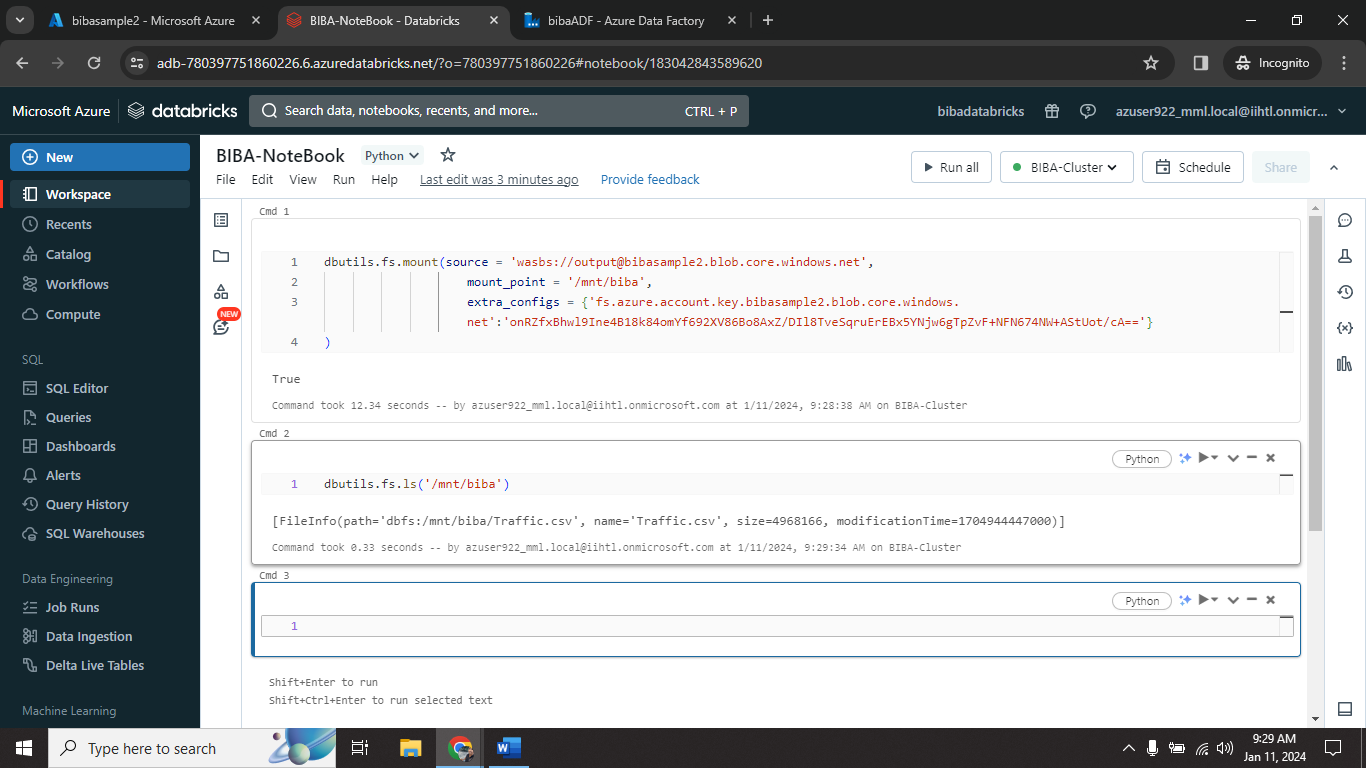


Syntax for access ADLS account from Databericks:

Dbutils.fs.mount.(source=’wasbs://<containername>@<storageaccountname>.blob.core.windows.net’, mount\_point=’/mnt/<anyname>’, extra\_configs=(‘fs.azure.account.key.<storageaccountname>.blob.core.windows.net’:’<copied key>’))

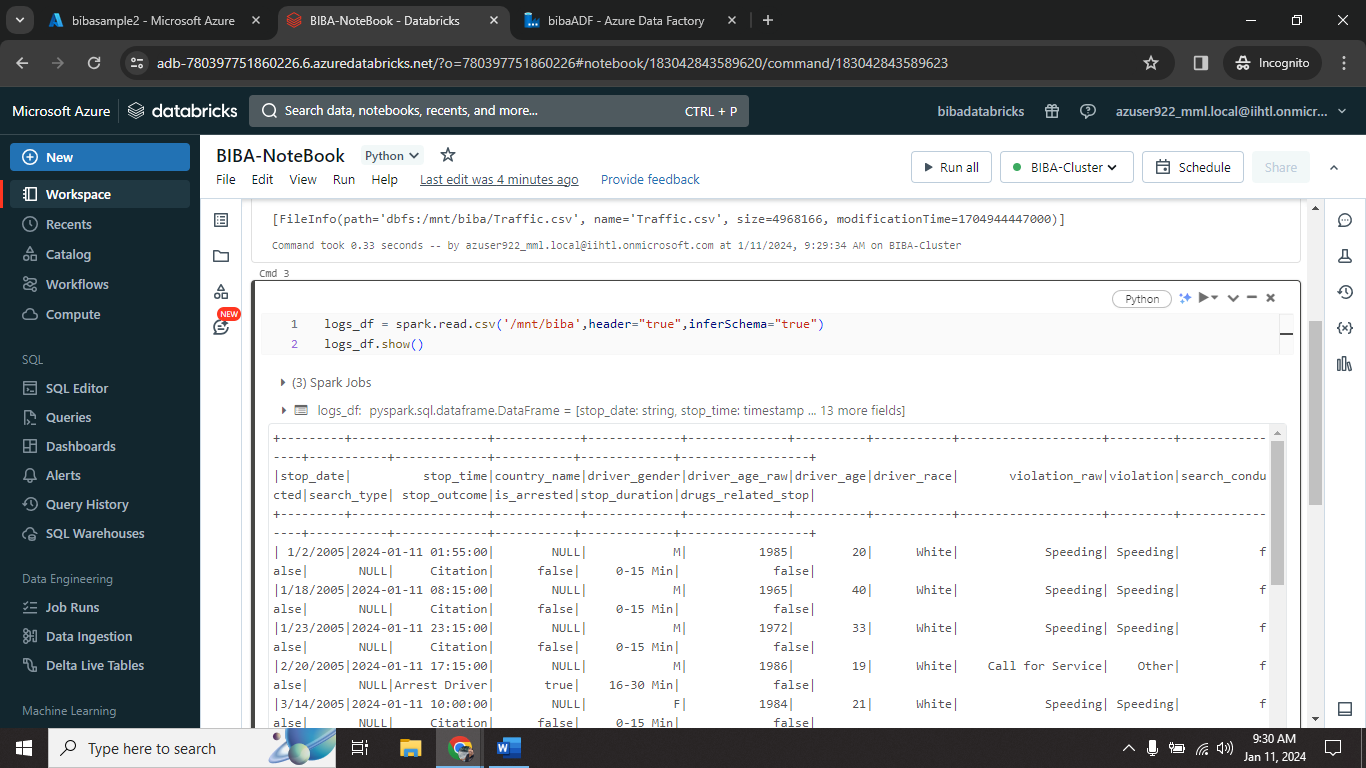
Step2🡺 create mount for access log file.



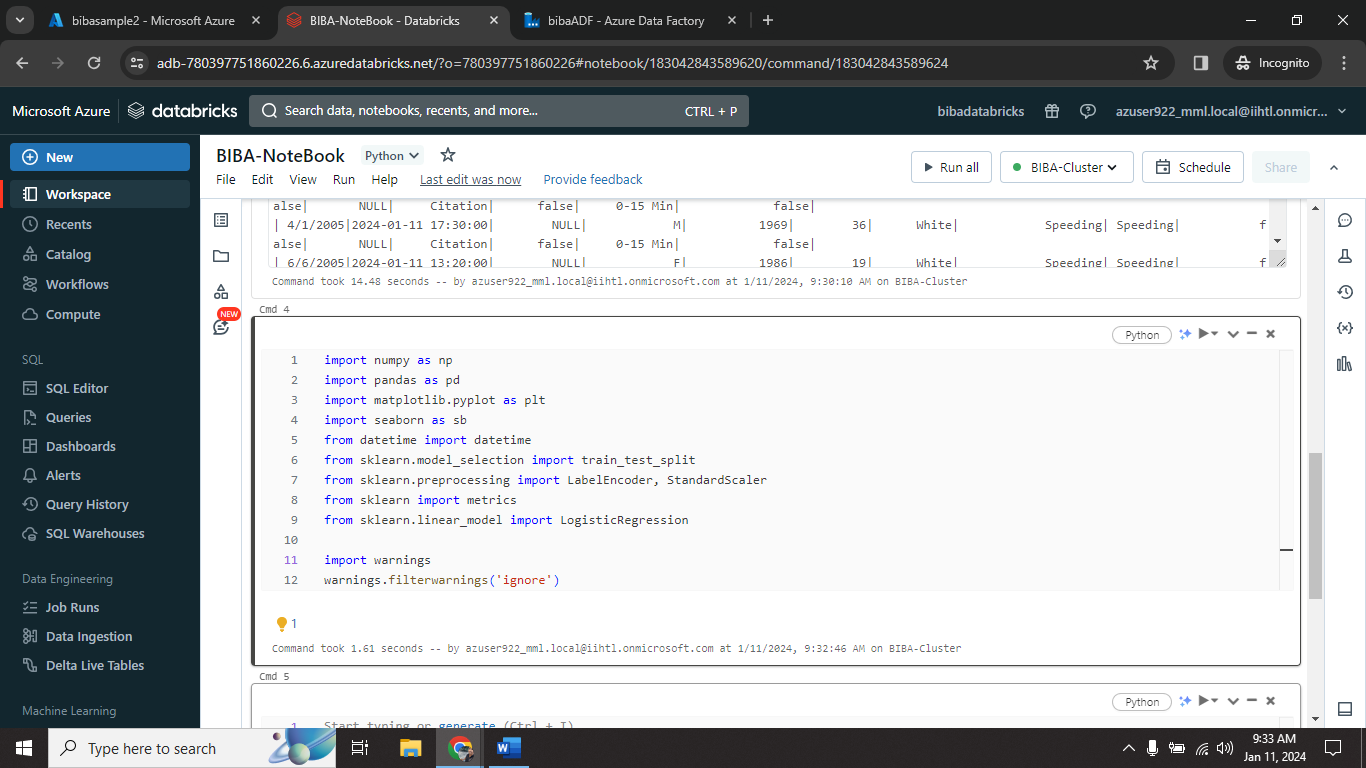
Step3🡺 mount successfully created.

Step4🡺 display available log files in mounted directory.

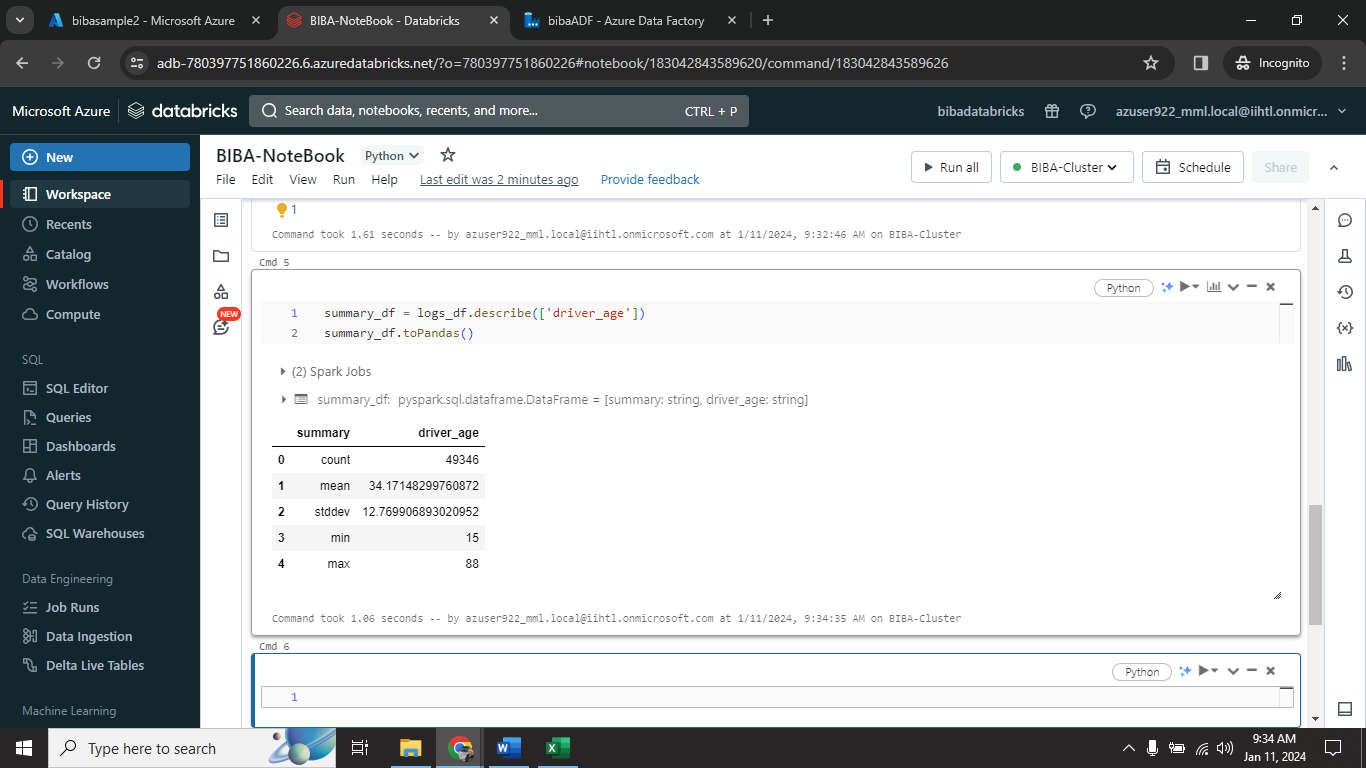
**6.9 Read Log Files**



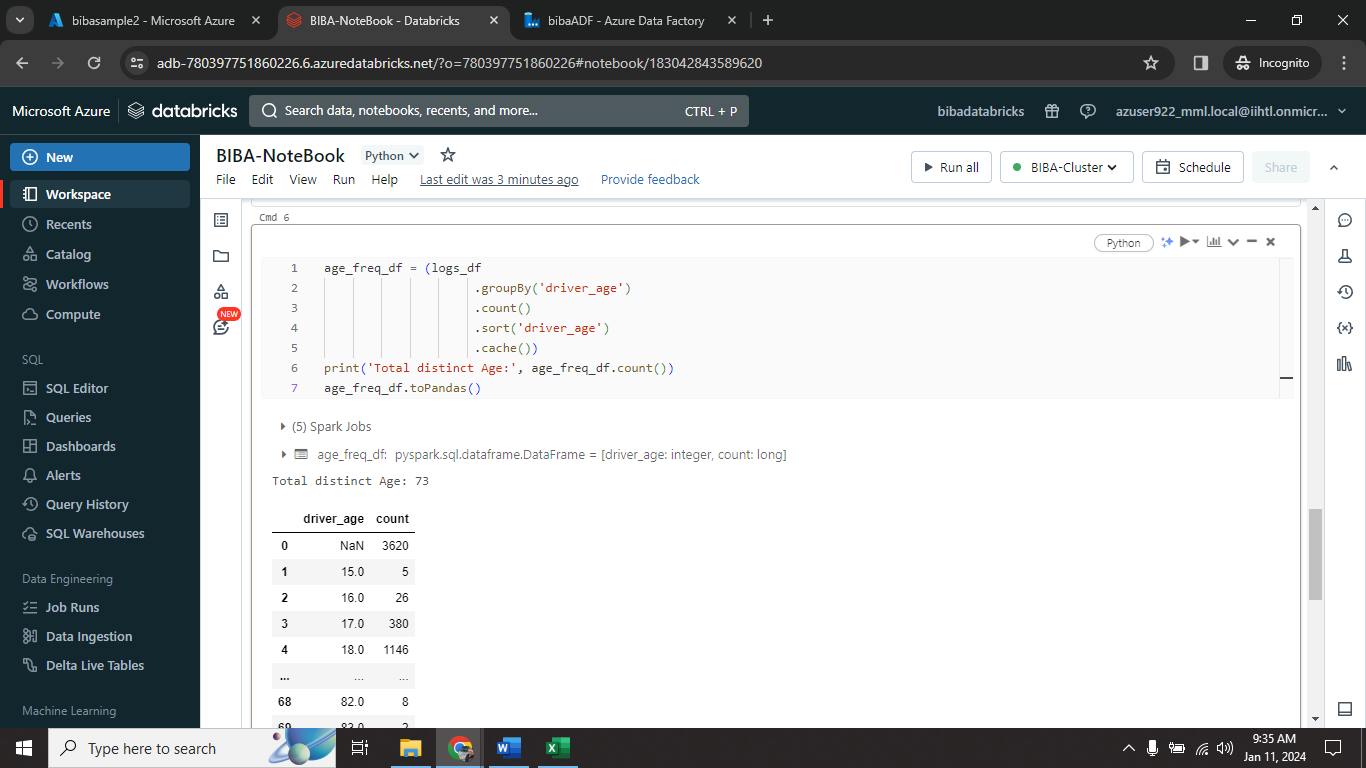
Step1🡺 read log file in notebook store in **logs\_df** variable.



Step2🡺 Initialize all needed modules.

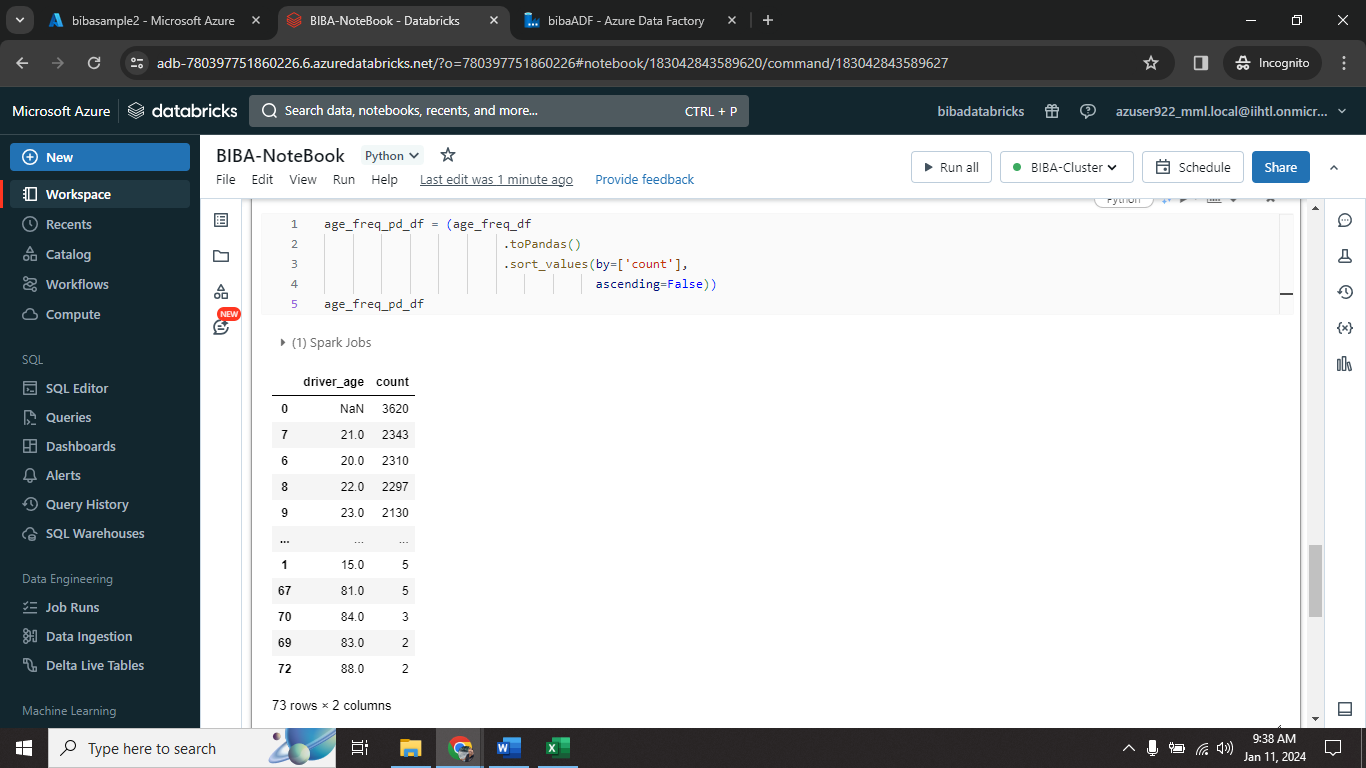


Step3🡺 summarize log file based on **driver\_age** column.

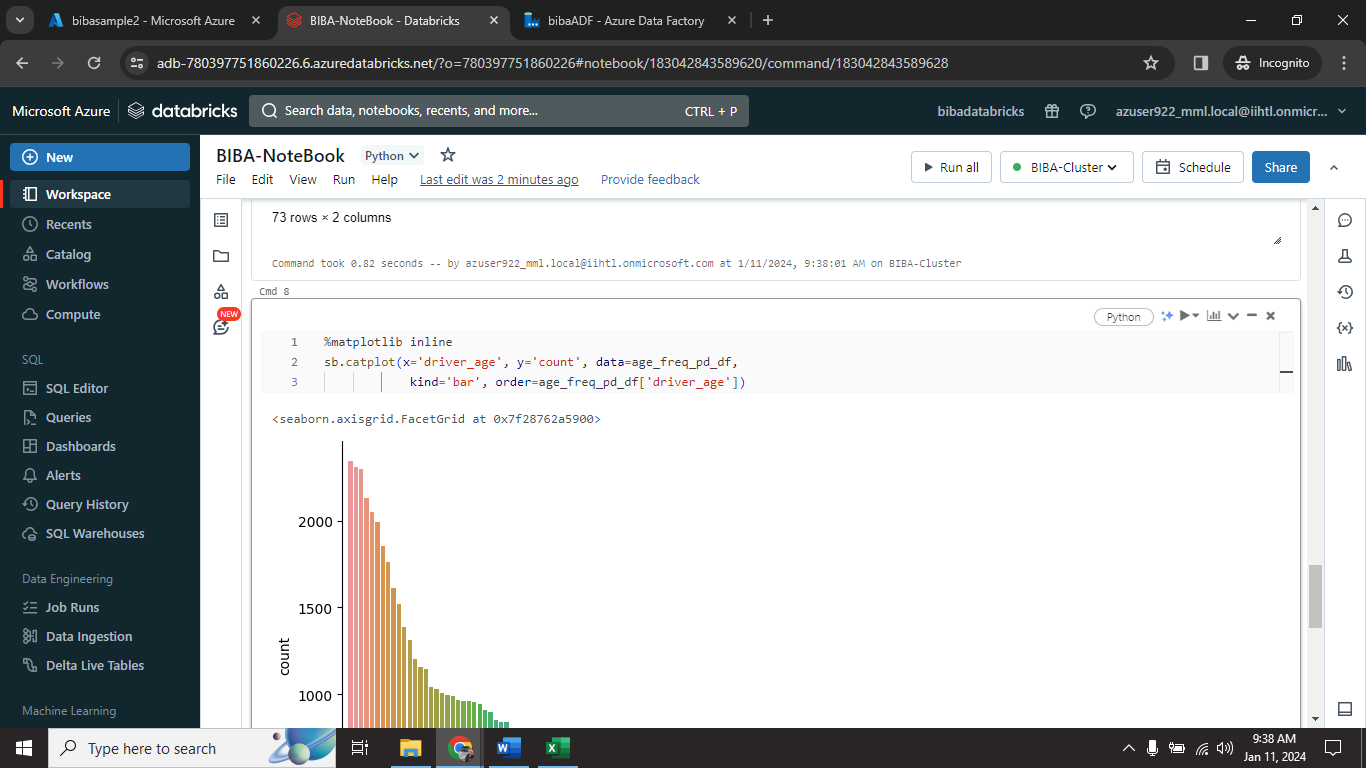


Step4🡺 display log data frame group by **driver\_age** with **count.**

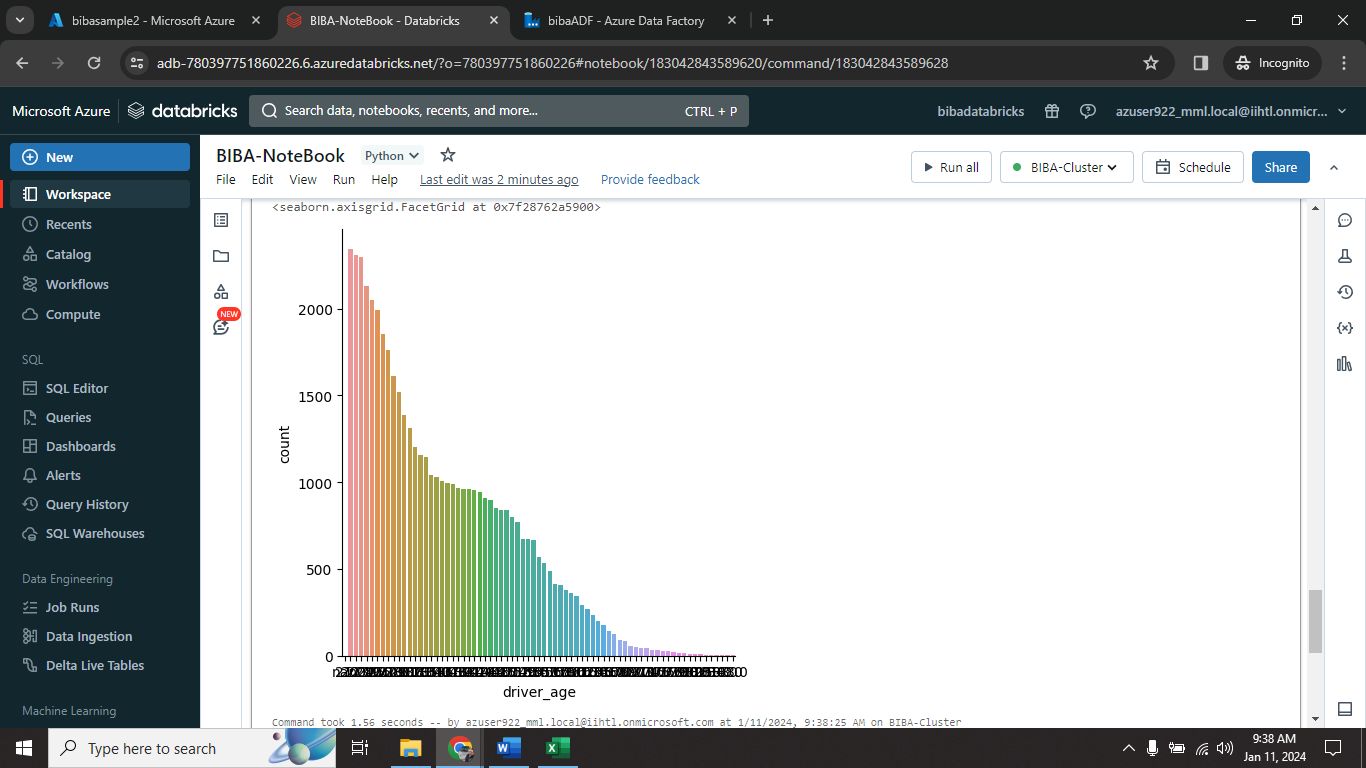




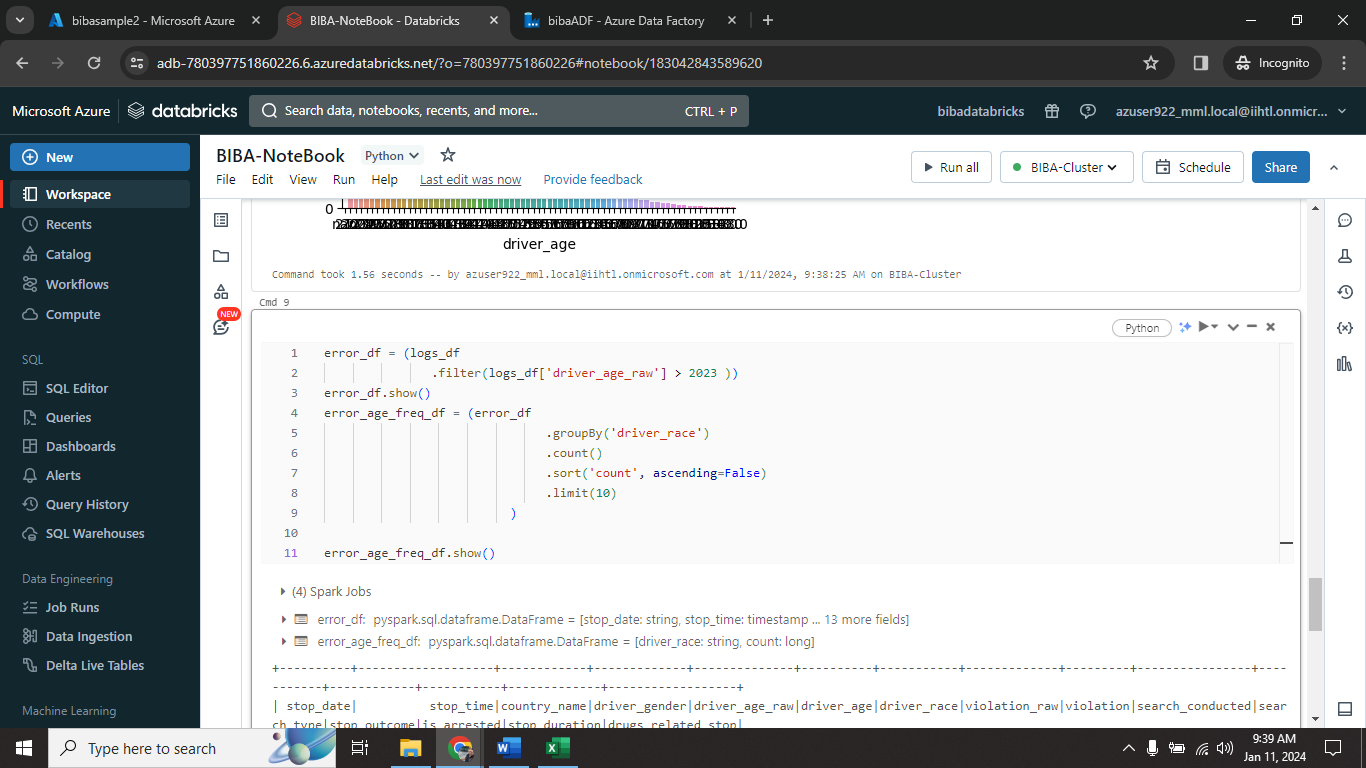
Step5🡺 create age based dataframe (**age\_freq\_pd\_df**)



Step6🡺 Visualize age\_freq\_pd\_df in bar visualization.

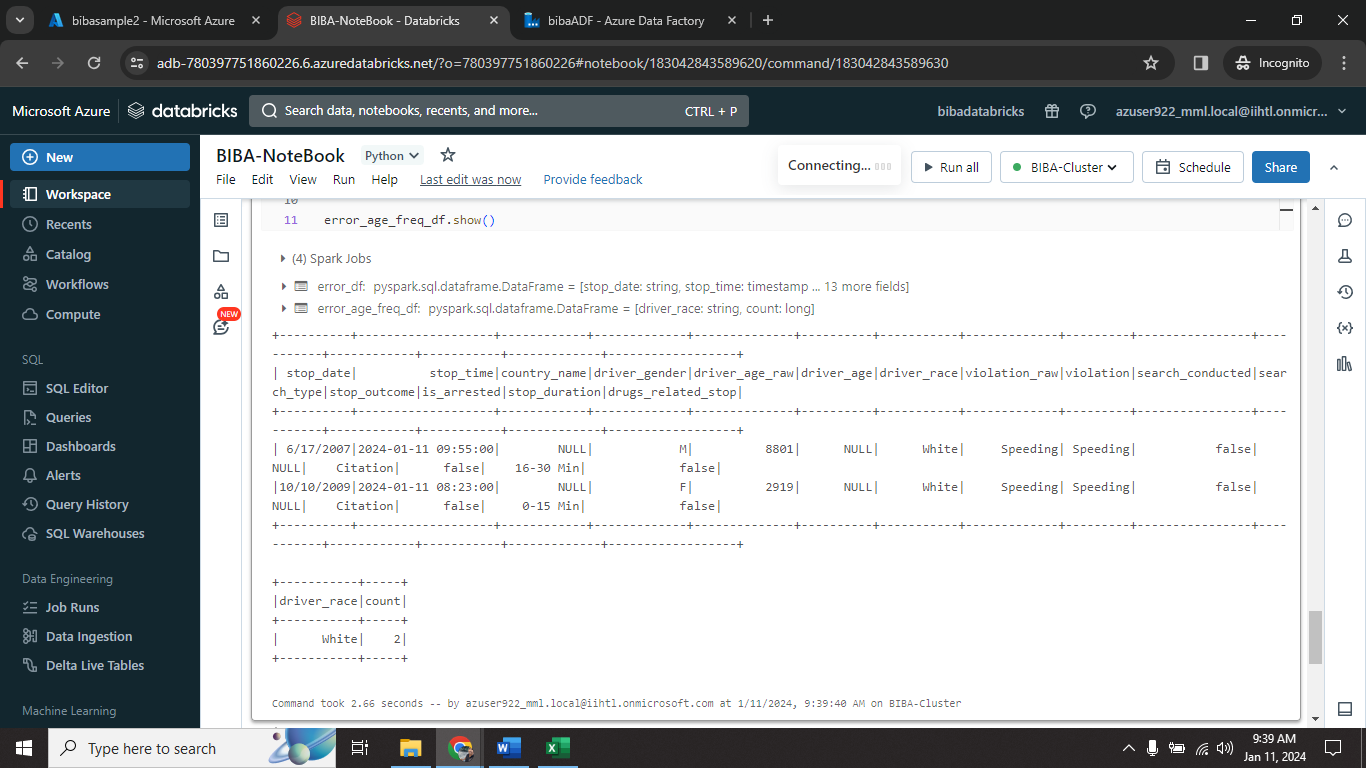


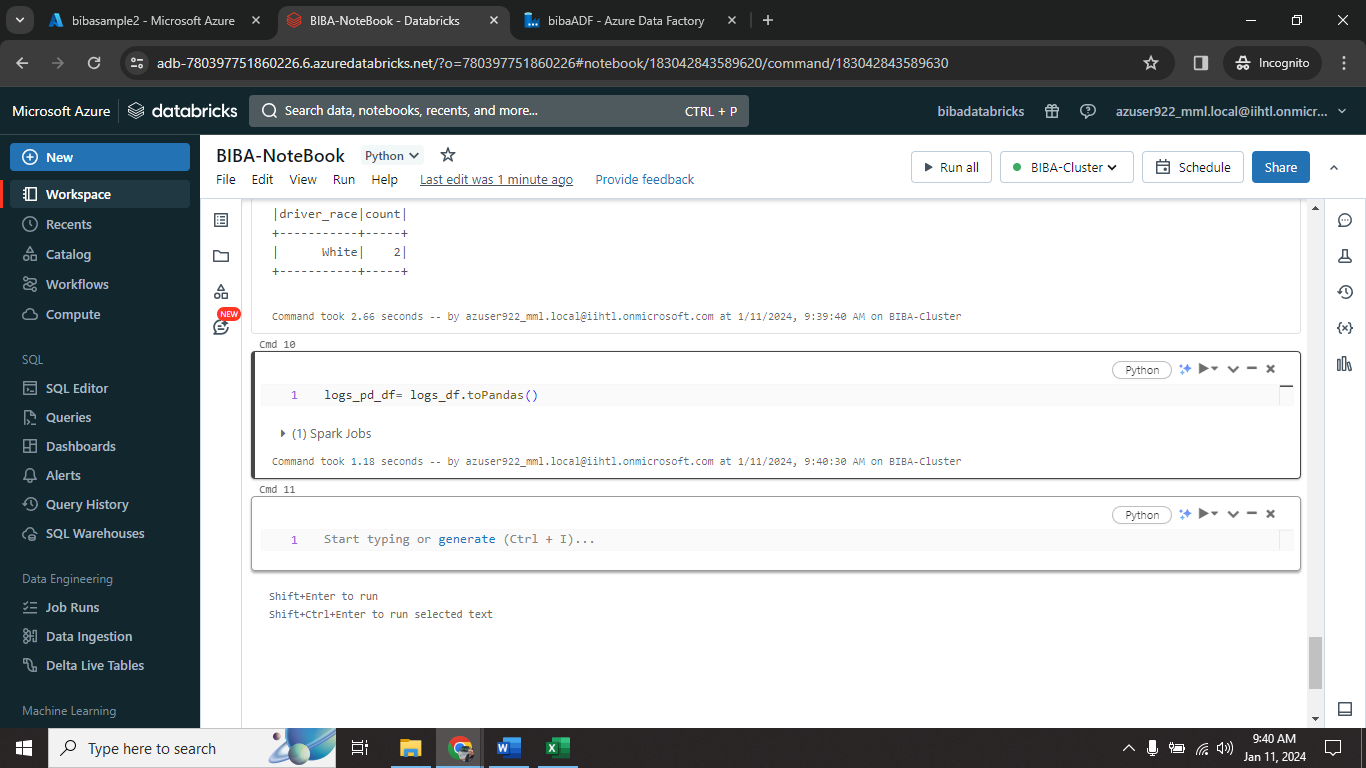
**6.10 Error Finding on Log files**



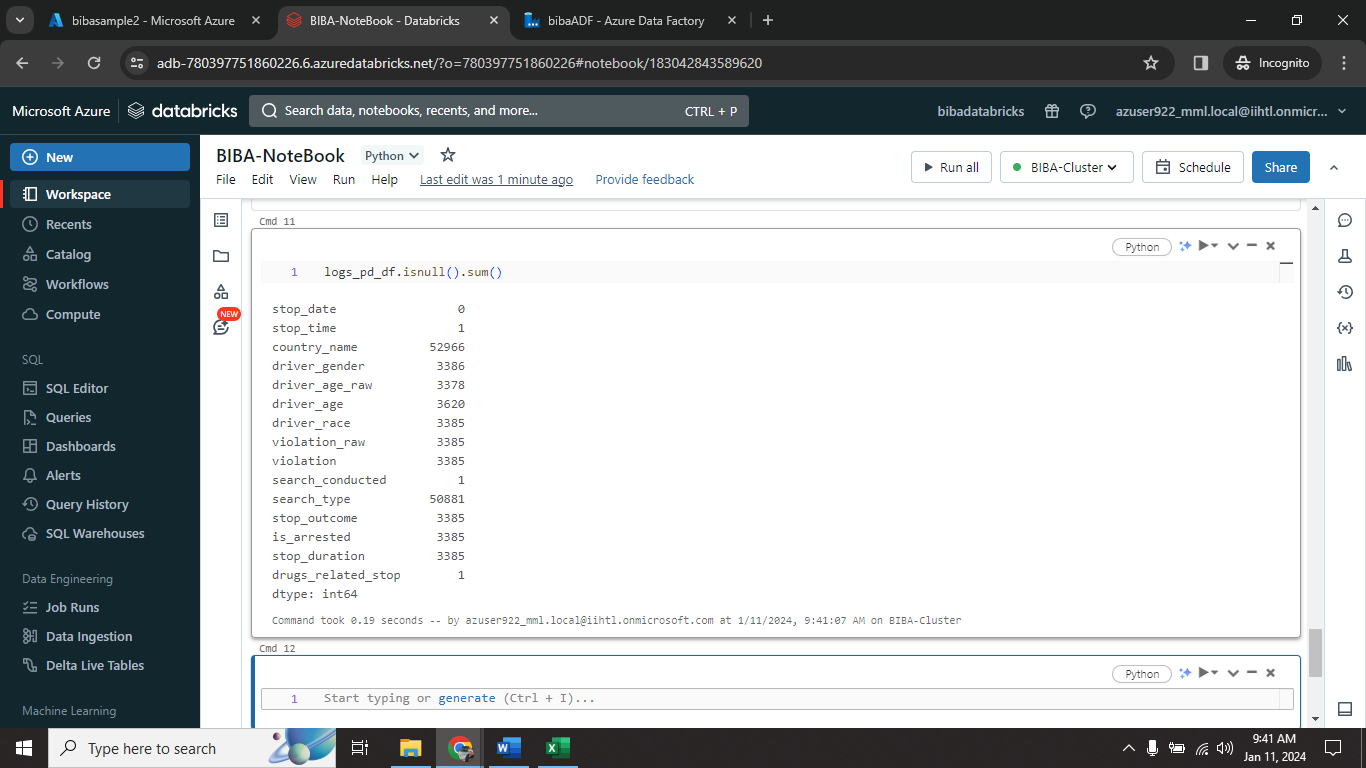
Step1🡺 display error data and save into **error\_df** variable.

This data frame display year error in log file (**driver\_age\_raw > 2023**)



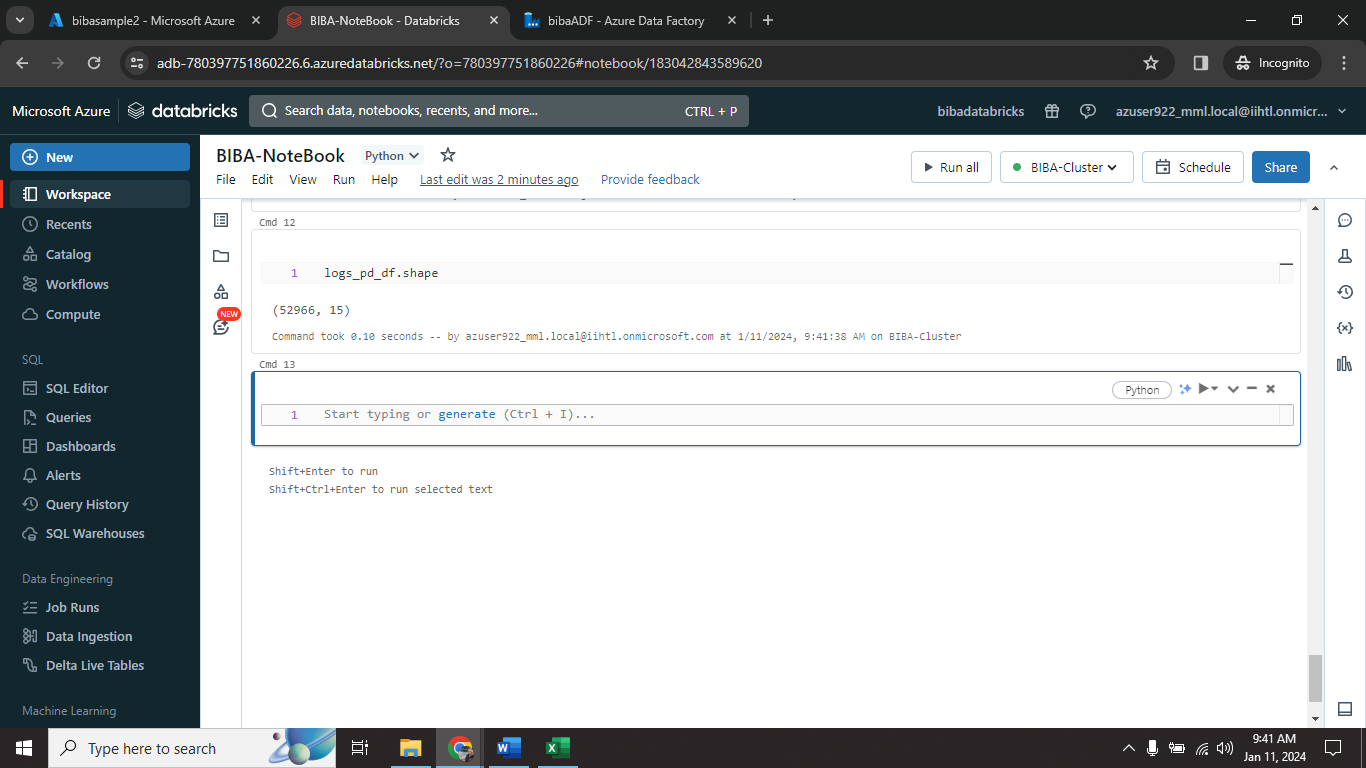


Step2🡺 create dataframe for **logs\_df** as **logs\_pd\_df.**



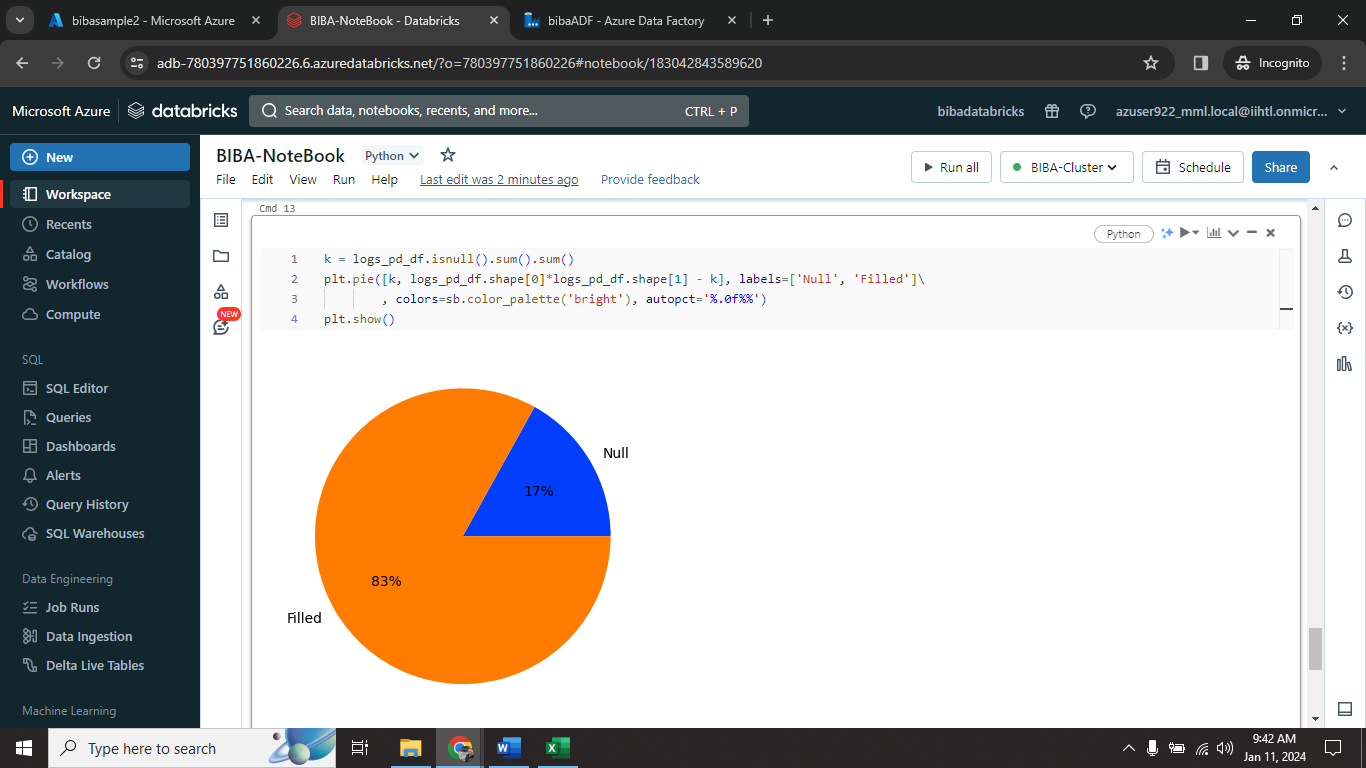
Step3🡺 Finding Null and empty values in log file. [logs\_pd\_df.isnull().sum()]

**6.11 Anomaly detection On Log Files**



Step1🡺 shape our dataframe to understand how many rows and columns their in log file.

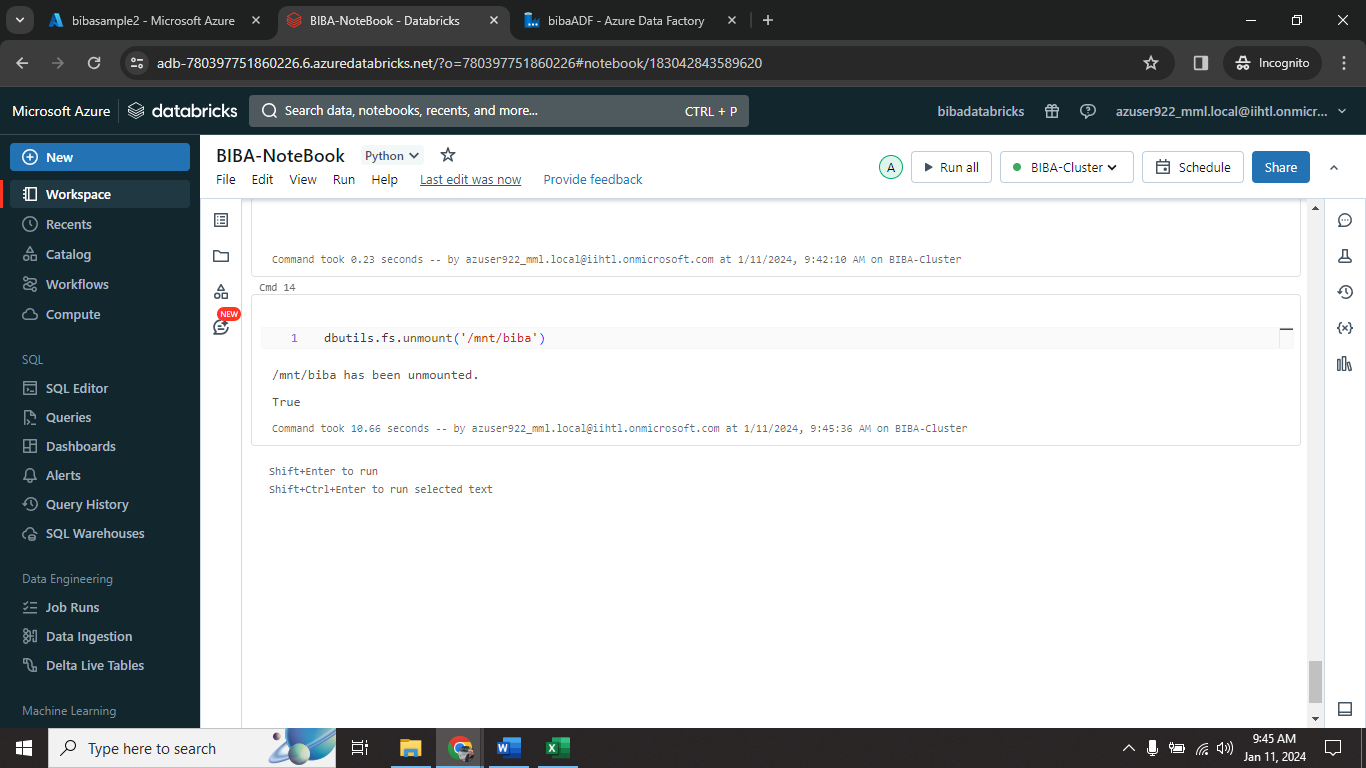
Rows = 52966 , Column = 15



Step2🡺 Find anomaly using pie chart representation.

Filled = 83% , Null = 17%

**6.12 Unmount ADLS from Databricks**



Step1🡺 Run dbutils.fs.unmount(‘/mnt/biba’) to unmount ADLS from Databricks.

**7)Conclusion**

* In conclusion, this project presents a comprehensive solution for log data management and analysis using Azure Data Factory and Azure Databricks. By leveraging these Azure services, the project aims to streamline the process of ingesting log data from diverse sources into Azure Storage and subsequently harnessing the power of Apache Spark for insightful analytics.
* The utilization of Azure Data Factory ensures efficient and scalable data movement, with the ability to handle incremental updates, while Azure Storage serves as a centralized repository for storing the ingested log data. The integration of Azure Databricks brings the advanced analytics capabilities of Apache Spark, enabling the extraction of valuable insights such as error rates, trends, and anomalies from the log data.