|  |  |
| --- | --- |
| Name | Enayathulla A |
| Date | 04/01/2024 |
| Type | Daily Assignment |
| Topic | Creating pipeline for ETL Process |

**Data Pipeline:**

A data pipeline implements the steps required to move data from source systems, transform that data based on requirements, and store the data in a target system. A data pipeline includes all the processes necessary to turn raw data into prepared data that users can consume. For example, a data pipeline might prepare data so data analysts and data scientists can extract value from the data through analysis and reporting.

An extract, transform, and load (ETL) workflow is a common example of a data pipeline. In ETL processing, data is ingested from source systems and written to a staging area, transformed based on requirements (ensuring data quality, deduplicating records, and so forth), and then written to a target system such as a data warehouse or data lake.

**Extract:**

The first stage in the ETL process is to extract data from various sources such as transactional systems, spreadsheets, and flat files. This step involves reading data from the source systems and storing it in a staging area.

**Transform:**

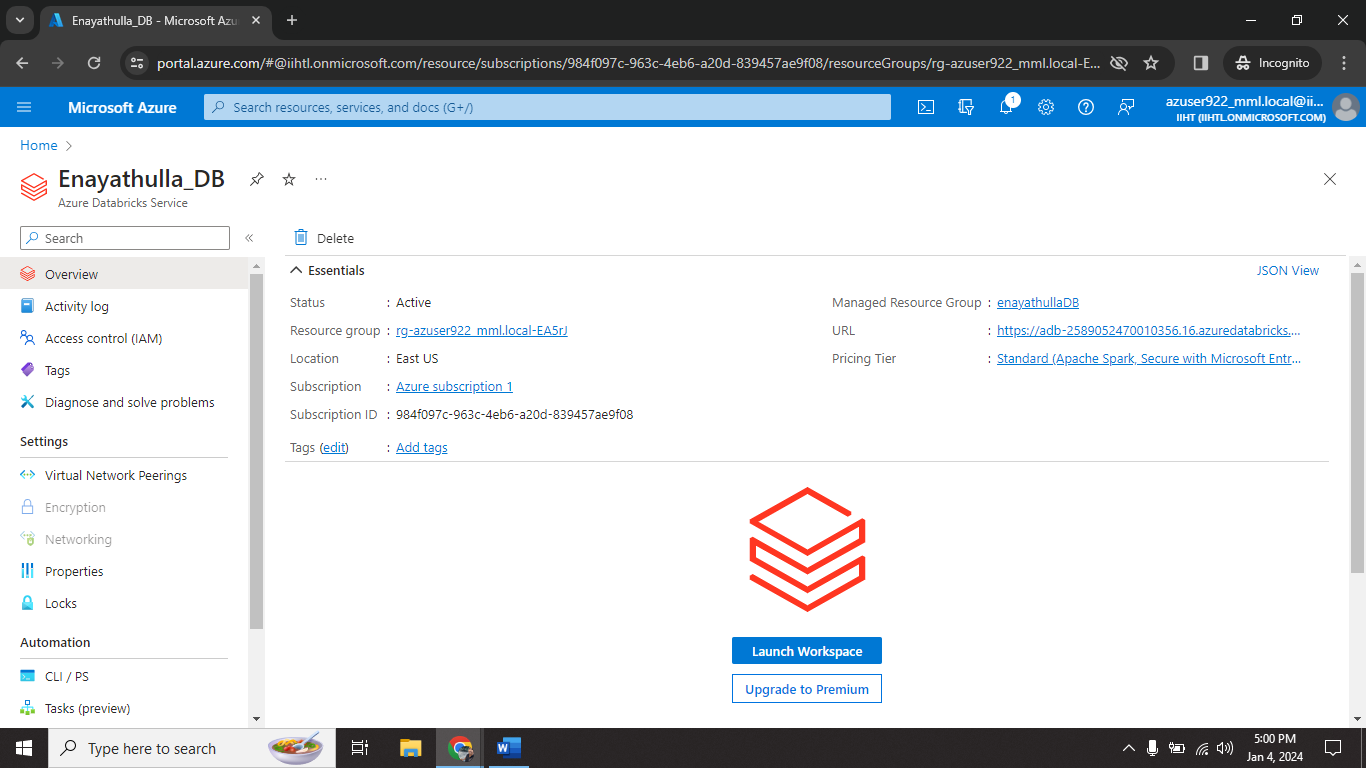
In this stage, the extracted data is transformed into a format that is suitable for loading into the data warehouse. This may involve cleaning and validating the data, converting data types, combining data from multiple sources, and creating new data fields.

**Load:**

After the data is transformed, it is loaded into the data warehouse. This step involves creating the physical data structures and loading the data into the warehouse.

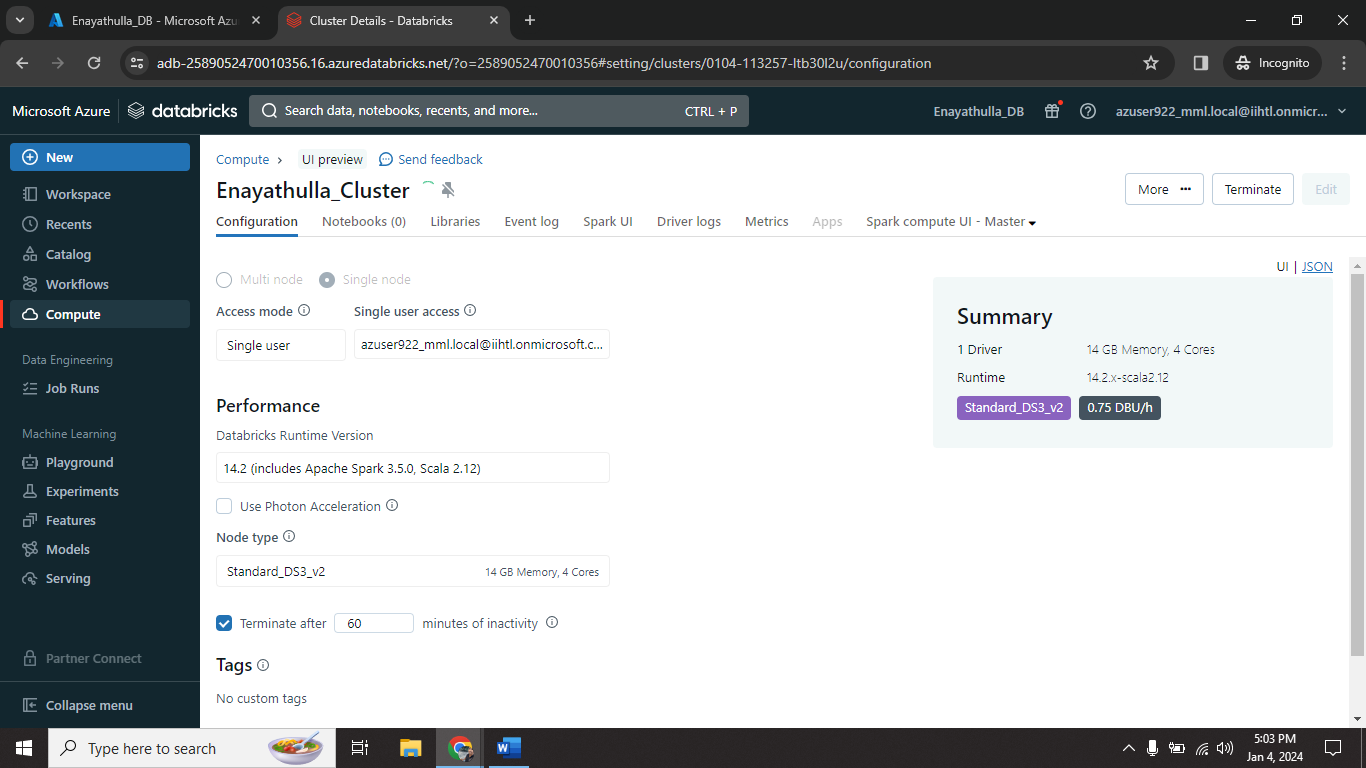
**Creating Data pipeline for ETL Process:**

**Databricks creation:**

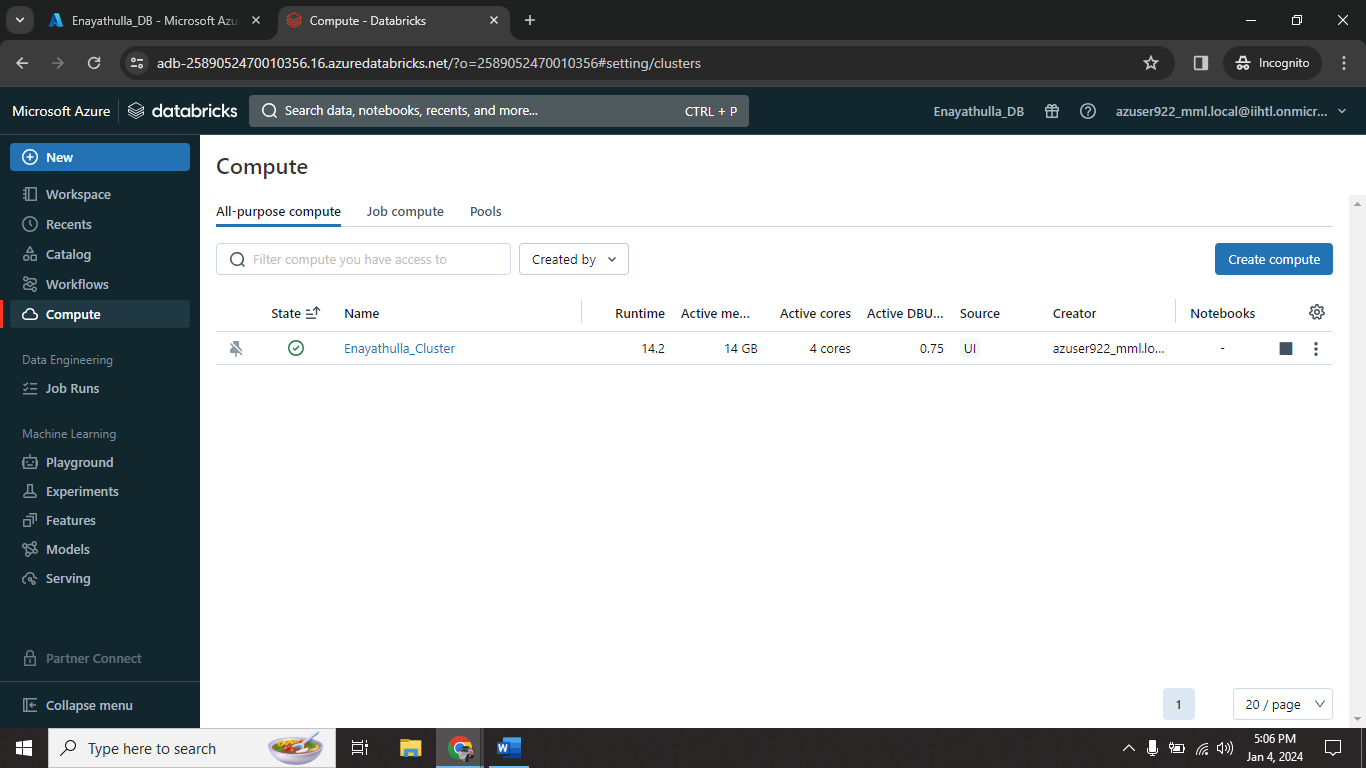


Step1🡺 create databricks resource on azure.

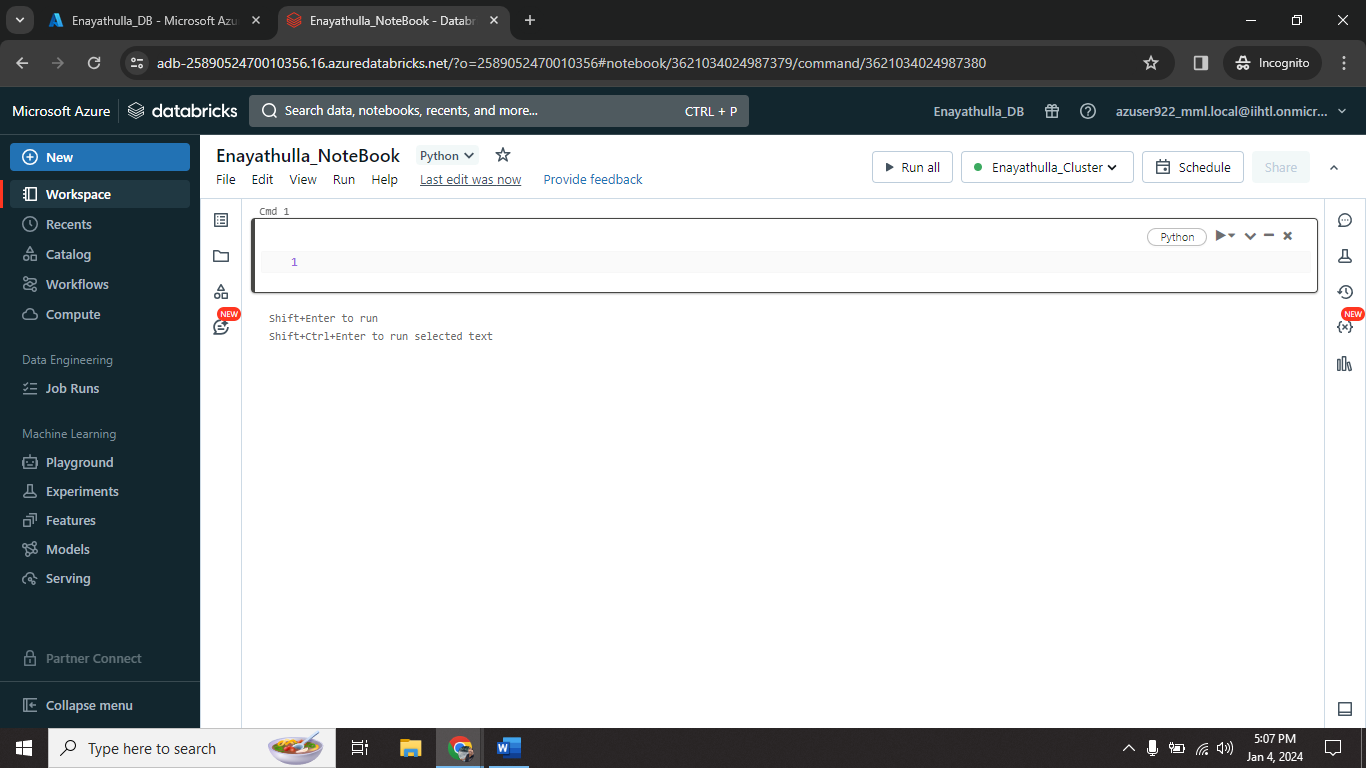
**Cluster creation:**



Step2🡺 create cluster on databricks.



**NoteBook Creation:**



Step3🡺 create NoteBook on that cluster.

Successfully created NoteBook.

**Sample code for enabling ETL pipeline on notebook:**

# Import functions

from pyspark.sql.functions import col, current\_timestamp

# Define variables used in code below

file\_path = "/databricks-datasets/structured-streaming/events"

username = spark.sql("SELECT regexp\_replace(current\_user(), '[^a-zA-Z0-9]', '\_')").first()[0]

table\_name = f"{username}\_etl\_quickstart"

checkpoint\_path = f"/tmp/{username}/\_checkpoint/etl\_quickstart"

# Clear out data from previous demo execution

spark.sql(f"DROP TABLE IF EXISTS {table\_name}")

dbutils.fs.rm(checkpoint\_path, True)

# Configure Auto Loader to ingest JSON data to a Delta table

(spark.readStream

.format("cloudFiles")

.option("cloudFiles.format", "json")

.option("cloudFiles.schemaLocation", checkpoint\_path)

.load(file\_path)

.select("\*", col("\_metadata.file\_path").alias("source\_file"), current\_timestamp().alias("processing\_time"))

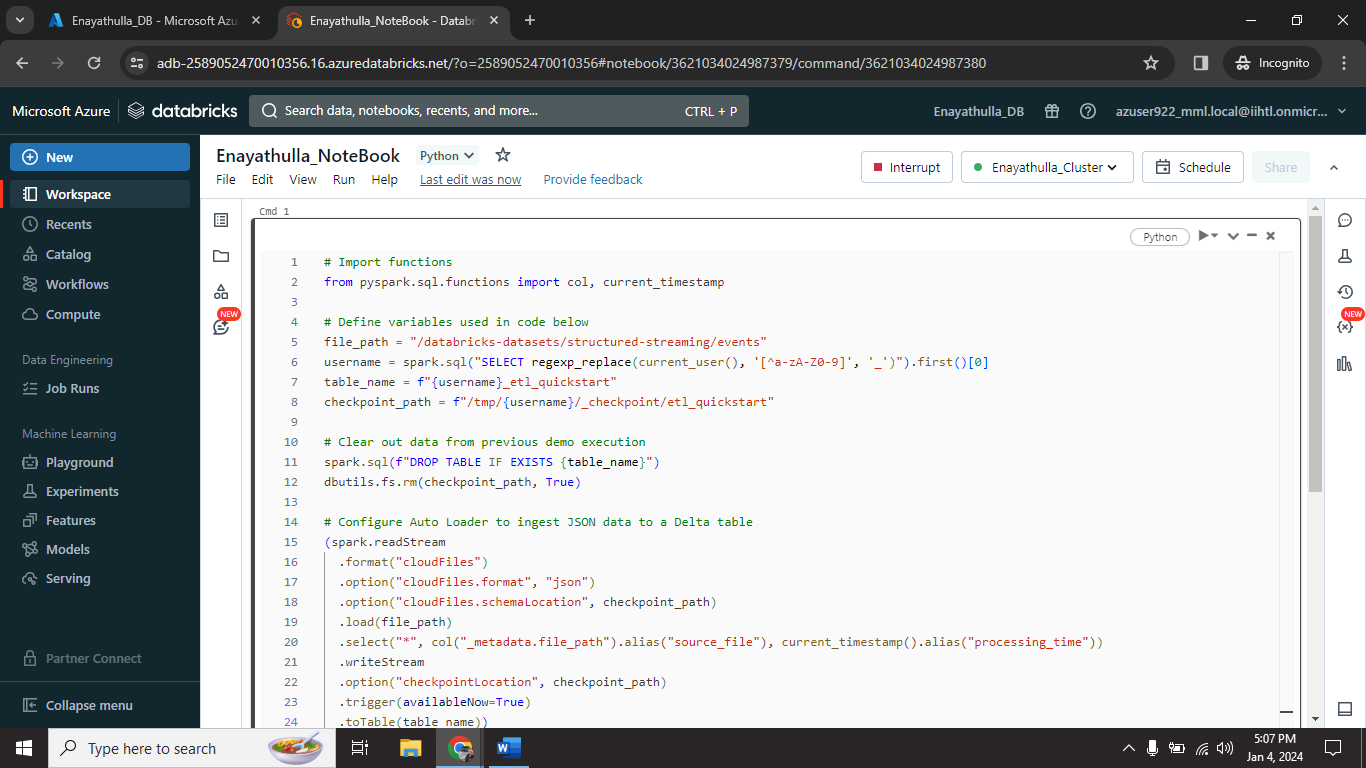
.writeStream

.option("checkpointLocation", checkpoint\_path)

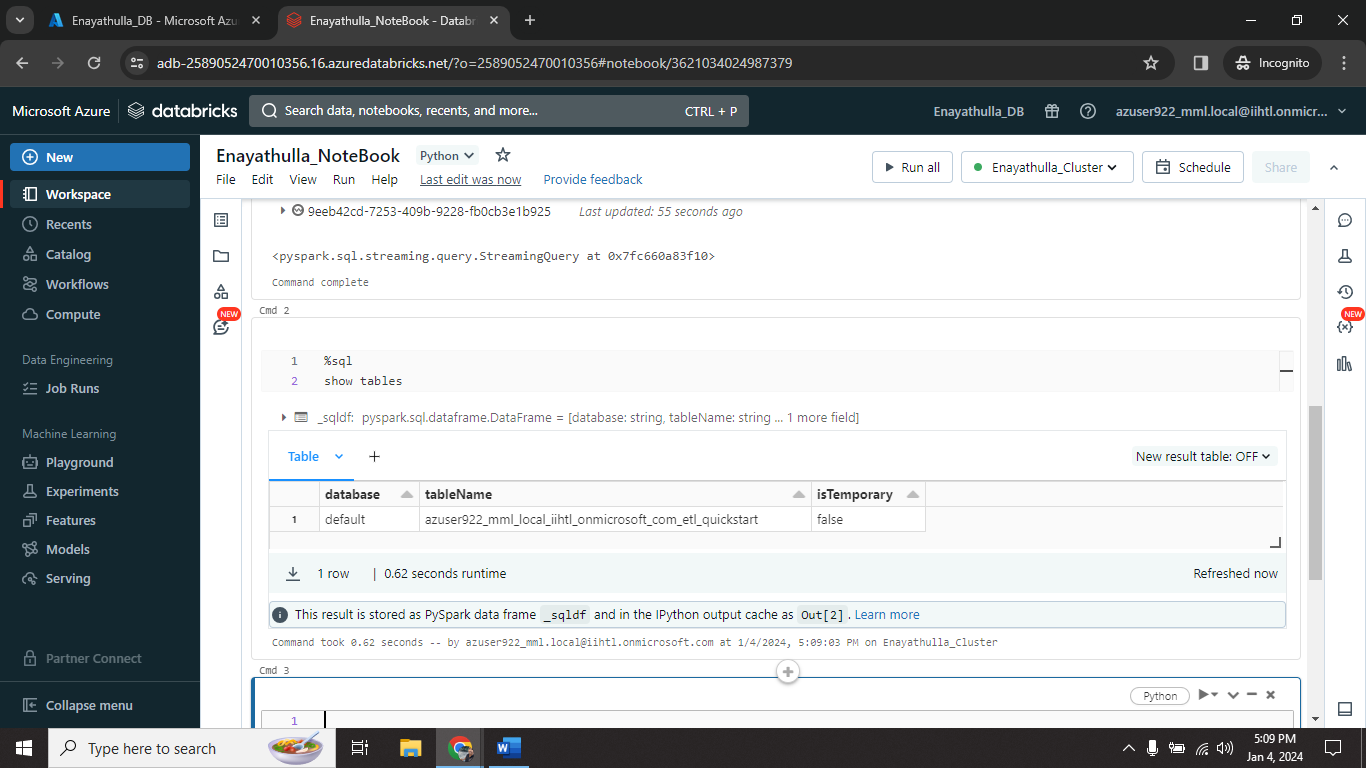
.trigger(availableNow=True)

.toTable(table\_name))

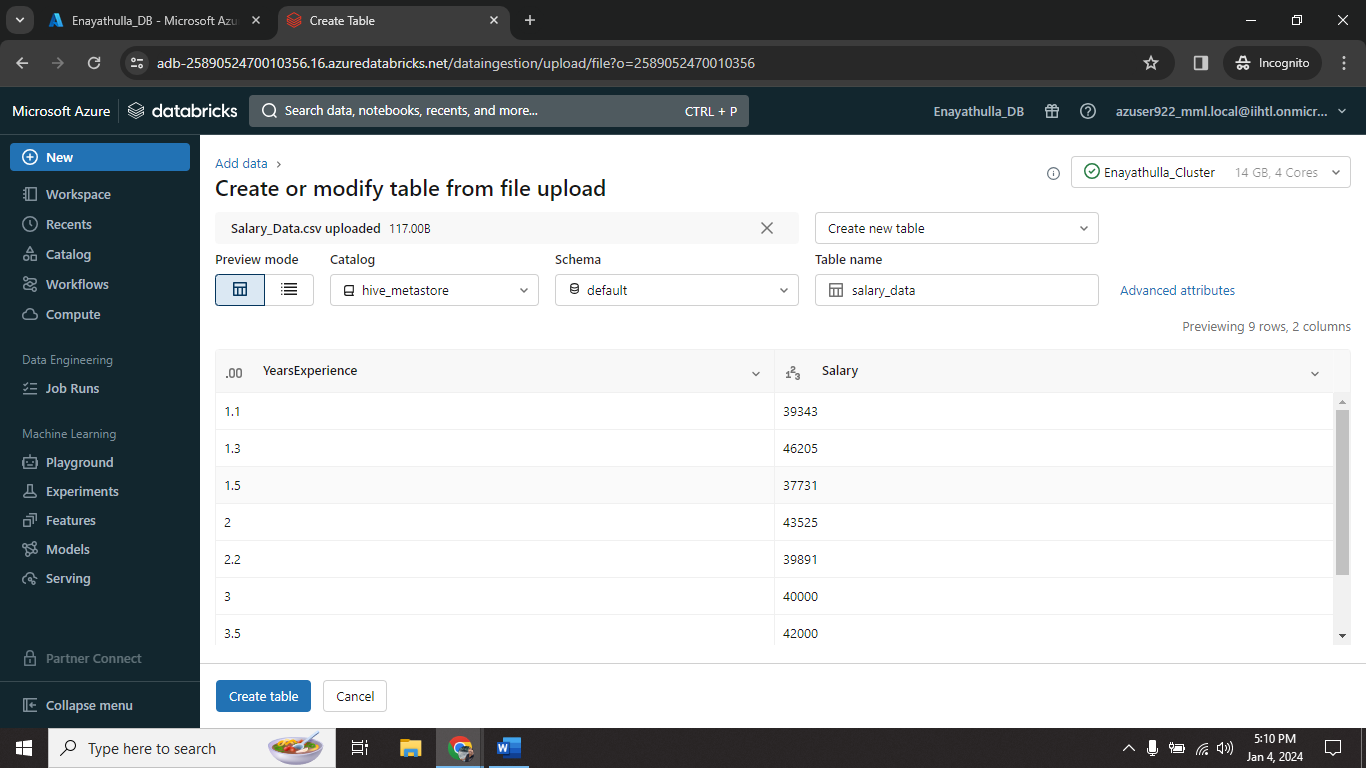
**Configure Auto loader to ingest data to data lake:**



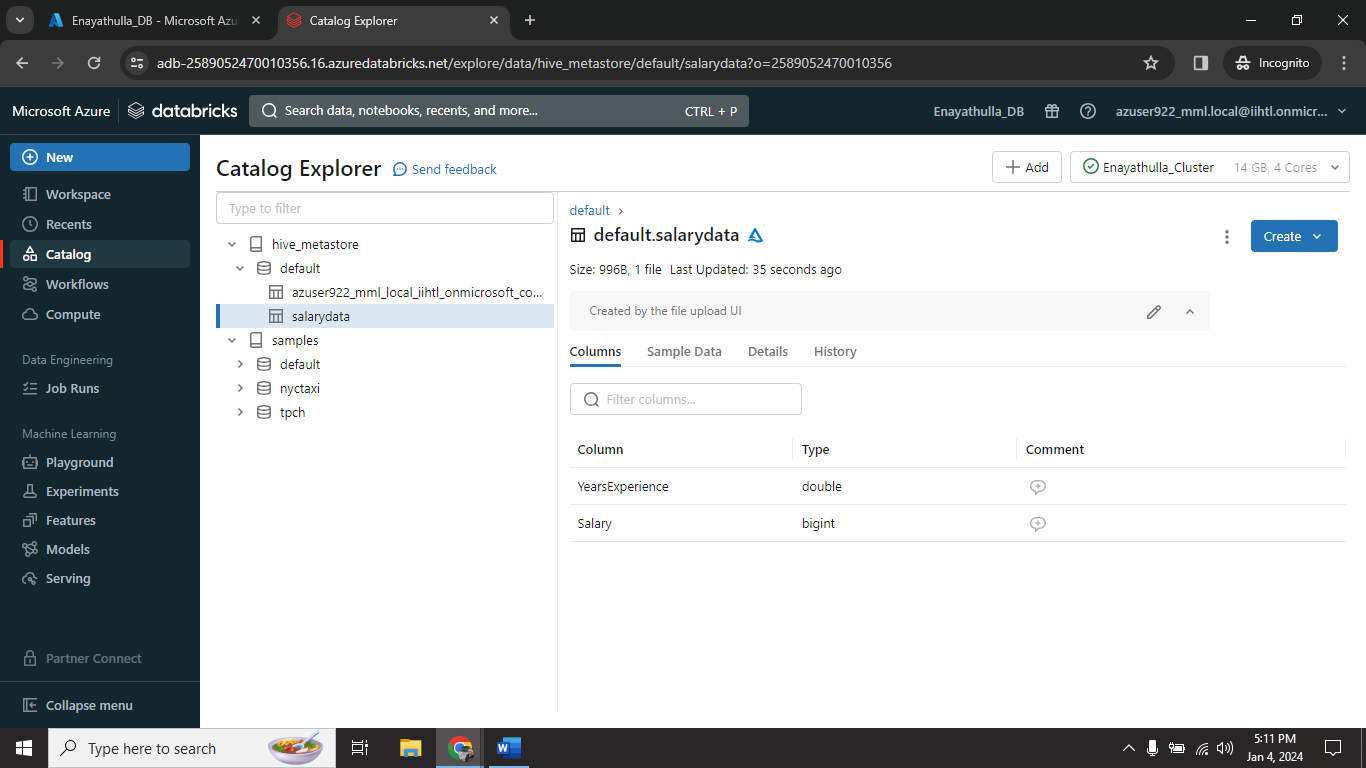
Step4🡺 Run the code into notebook.



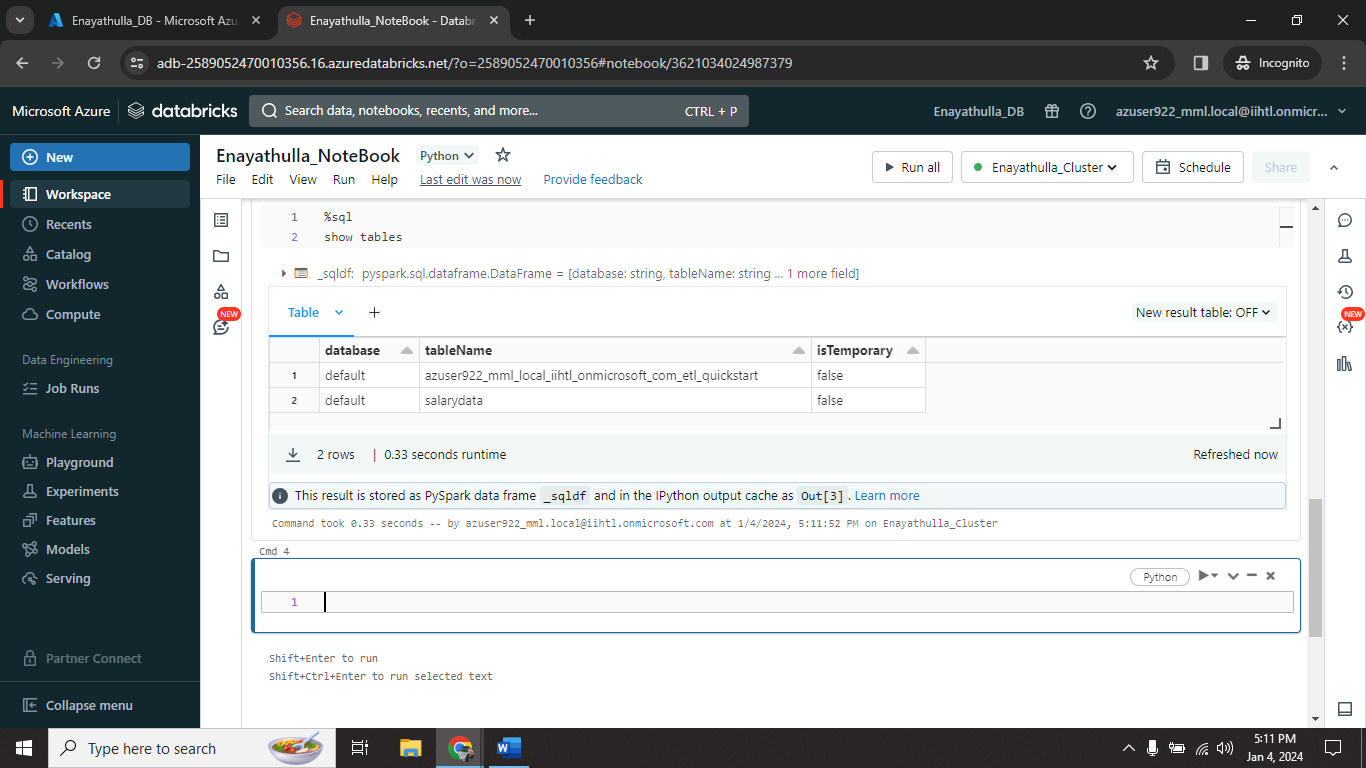
Step5🡺 display the available tables.



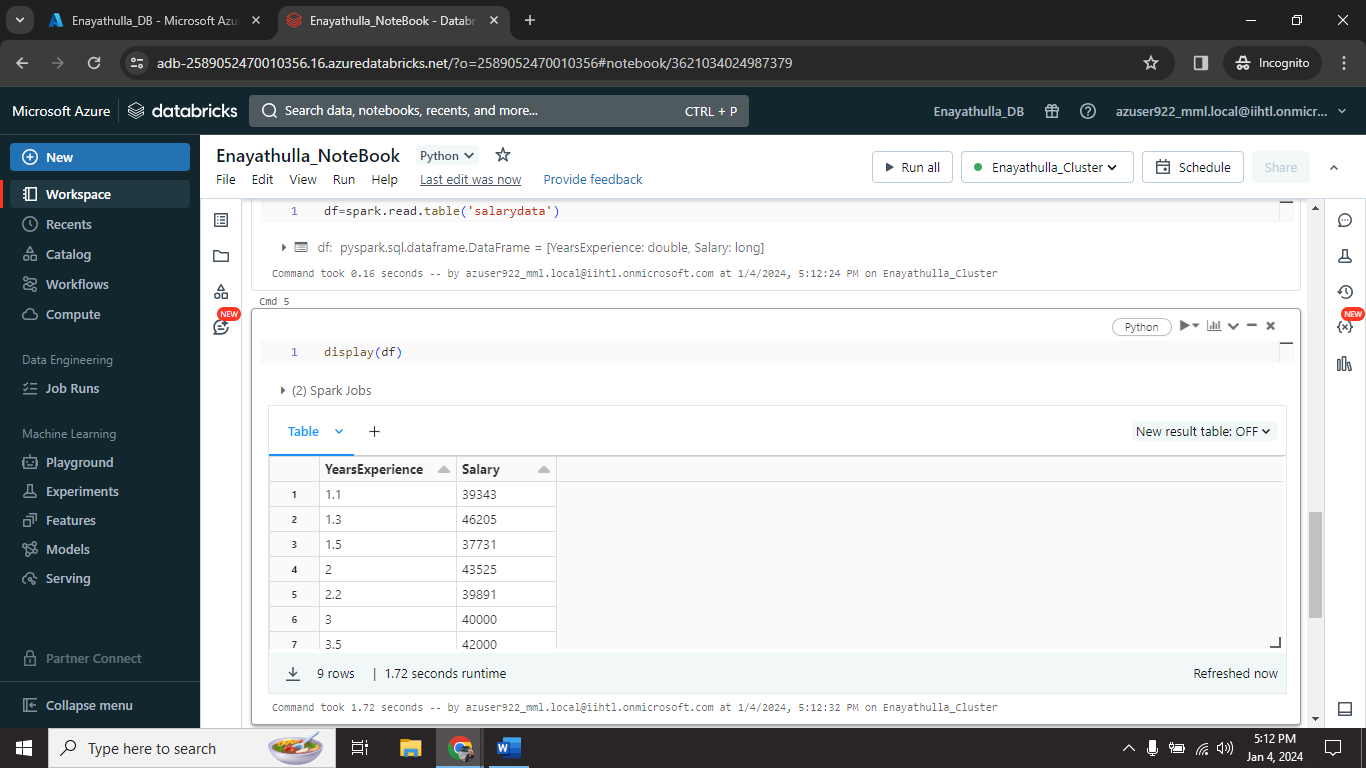
Step6🡺 create new table with local csv file.



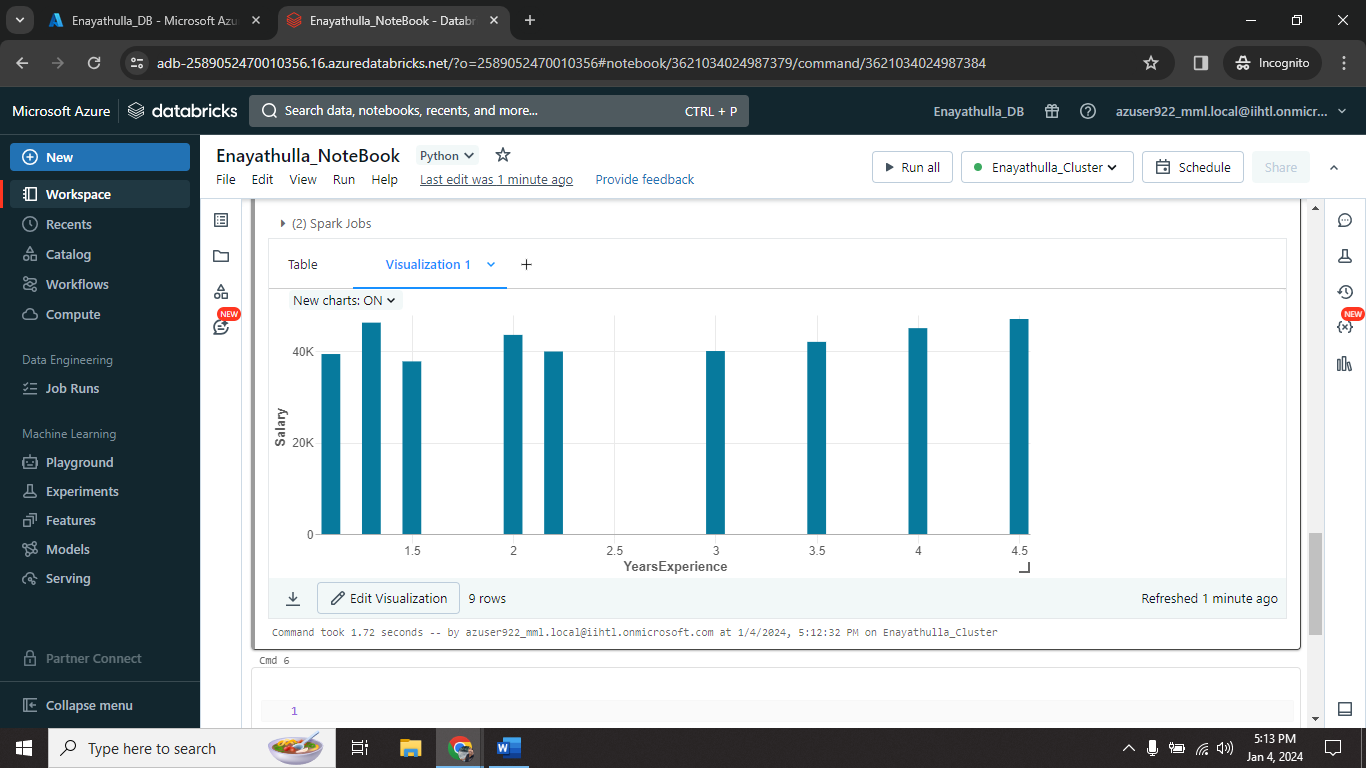
Successfully table created with the help of local CSV file.



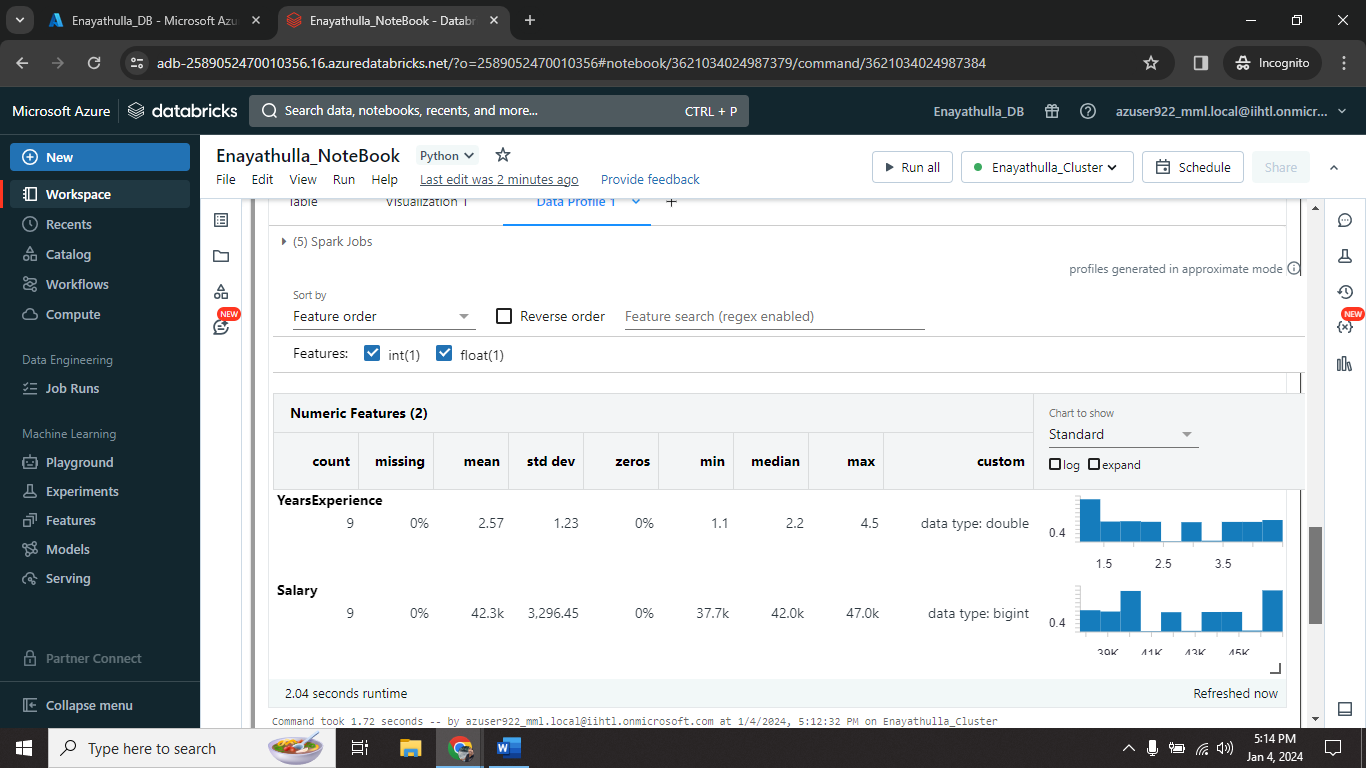
Step7🡺 now again run same query to display available tables.



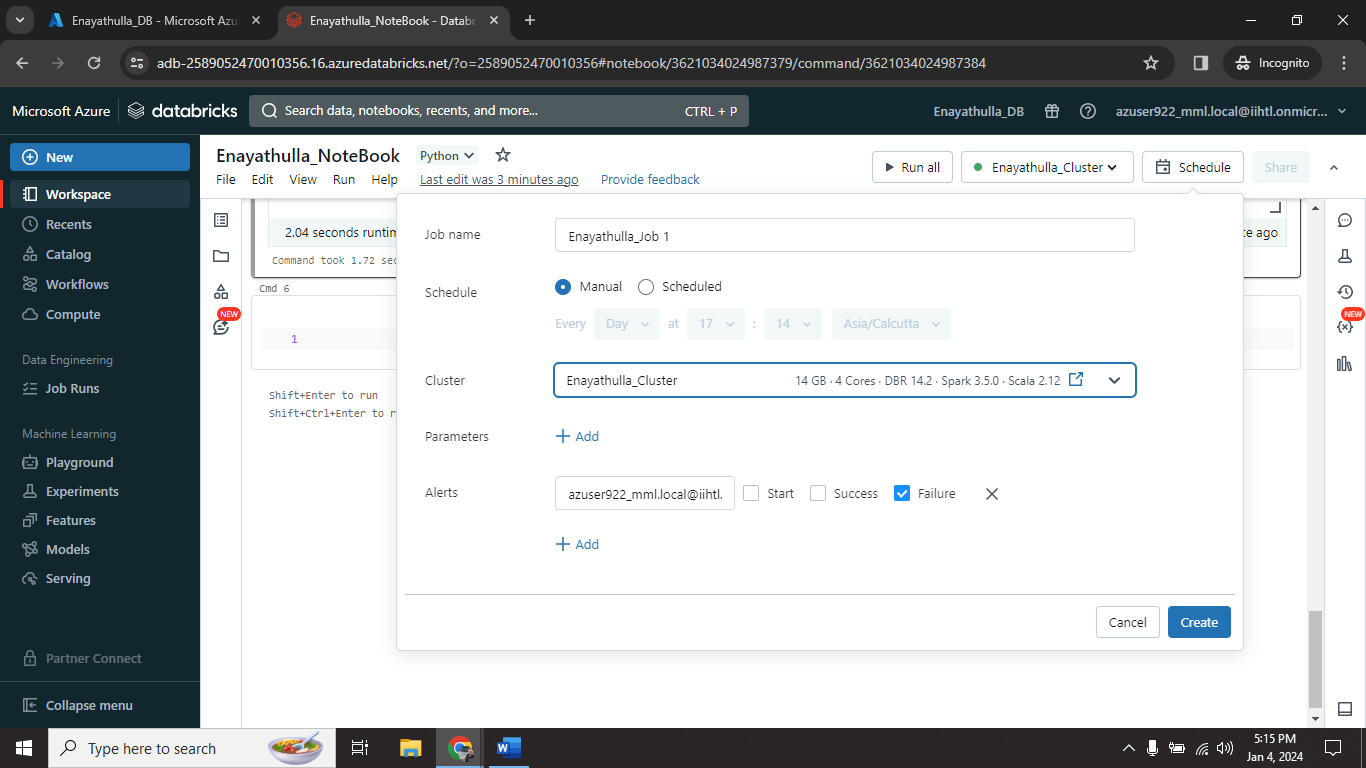
Step8🡺 read the table in pyspark.



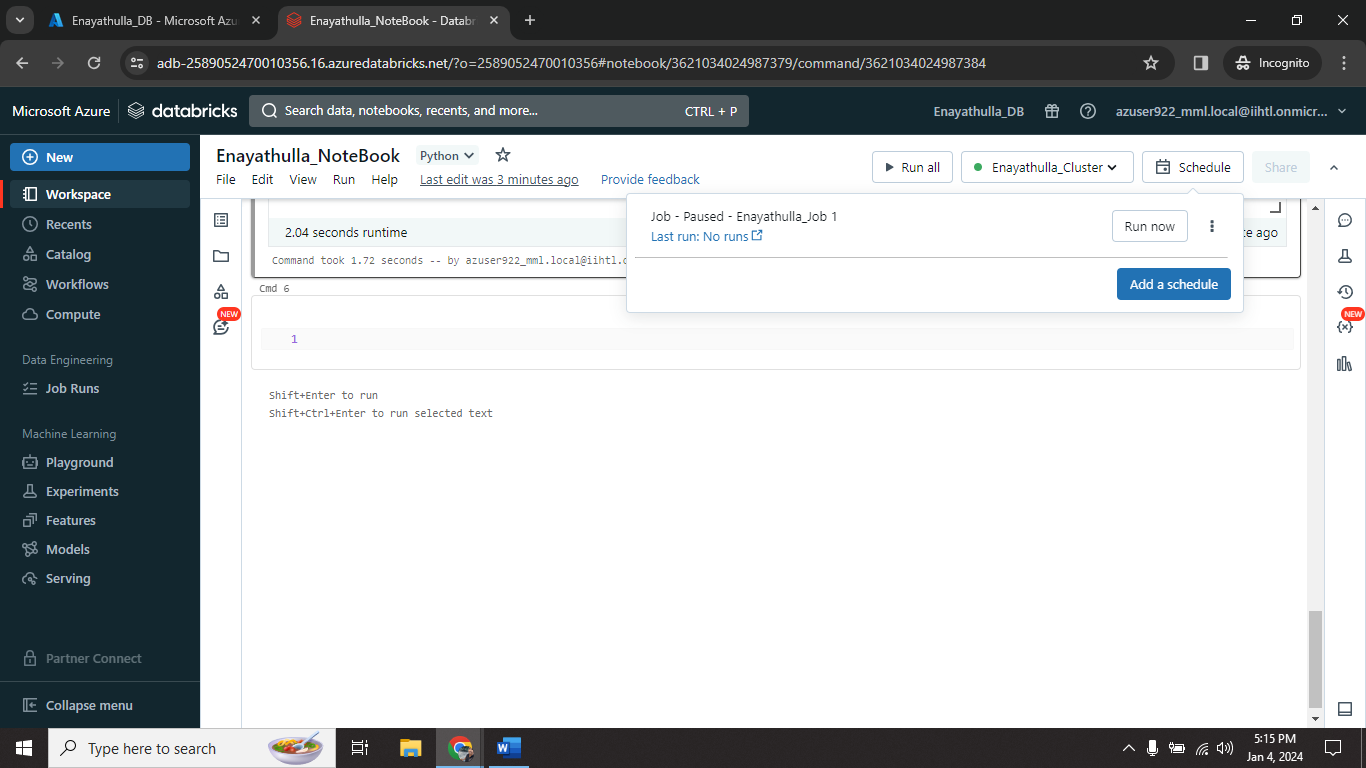
Step9🡺 Now display the table and perform various task on that.



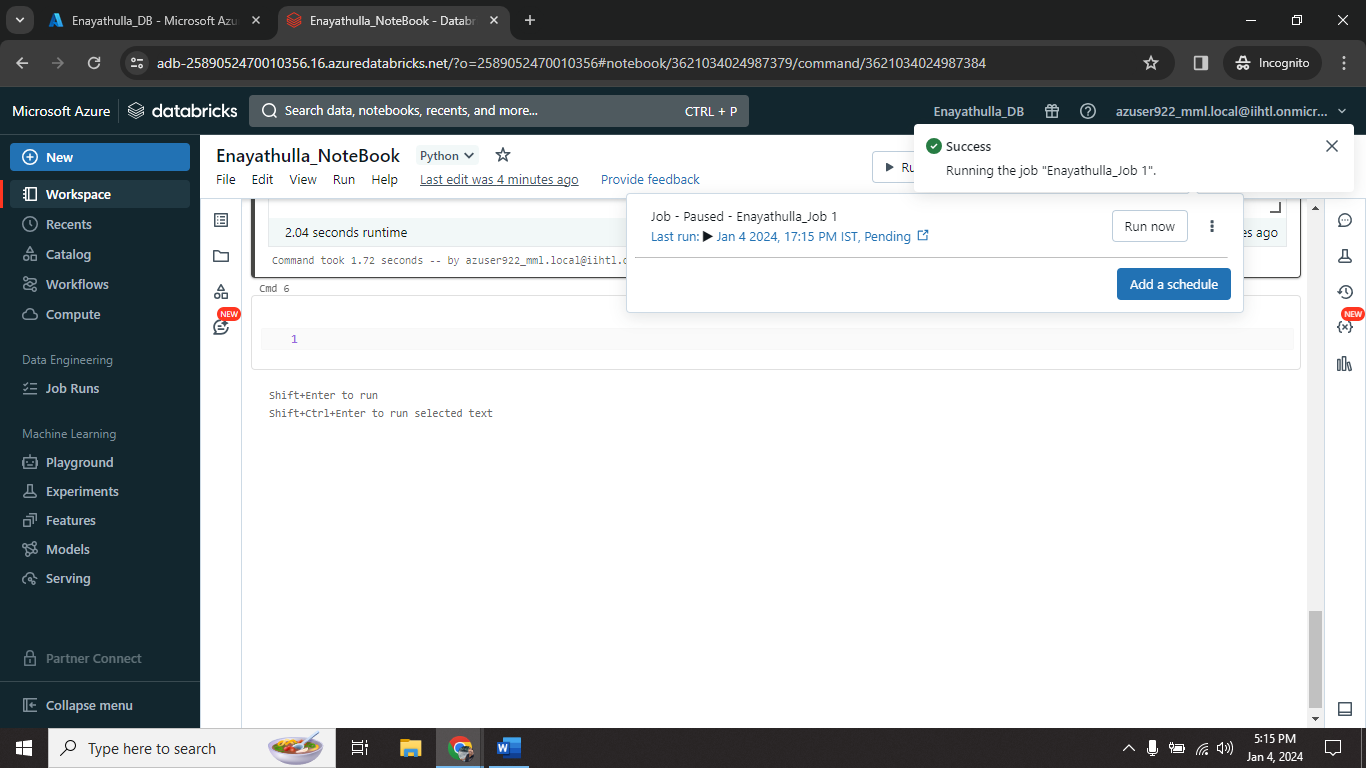
Successfully displayed Visualization and also Dataprofile.

**Schedule a Job:**

Step10🡺 now click on schedule to schedule a job.



Step11🡺 job successfully created and ready to run.



Step12🡺 after successfully executed job ; we delete that job based on our need.