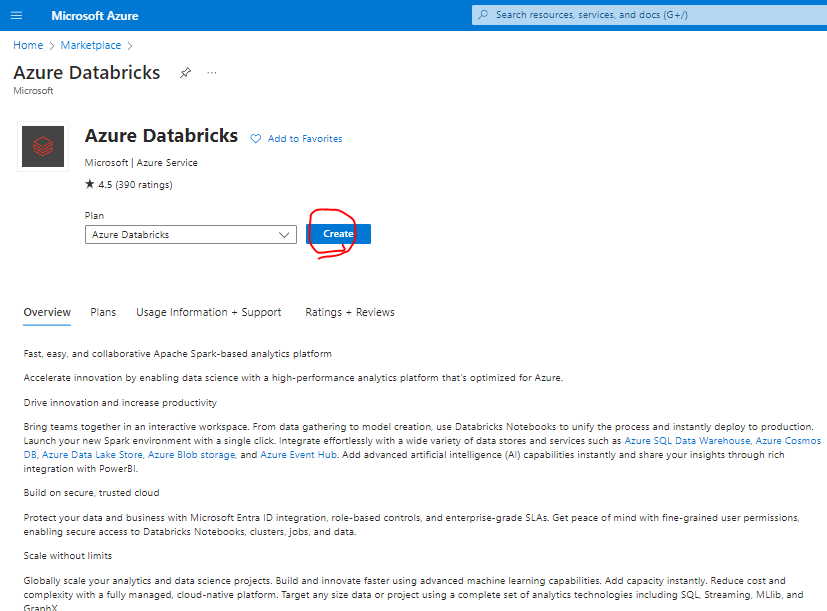
[Azure Databricks Tutorial | Data transformations at scale - YouTube](https://www.youtube.com/watch?v=M7t1T1Q5MNc)

1. Buscar el servicio en azure:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

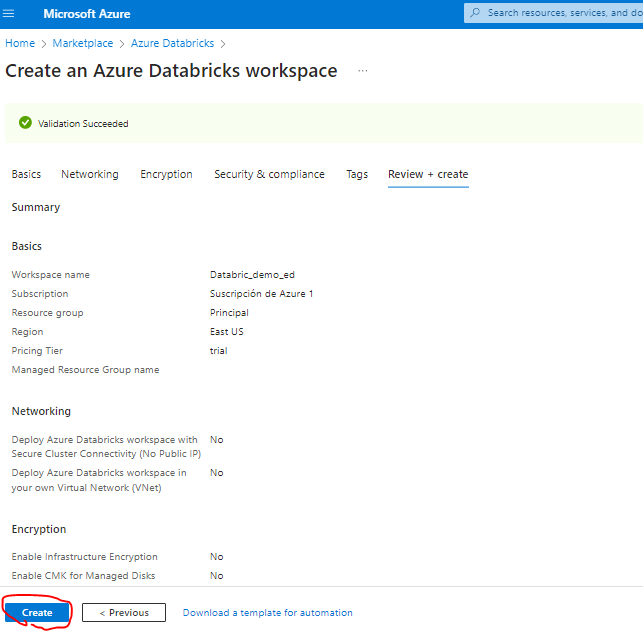
Descripción generada automáticamente

1. Crear el servicio

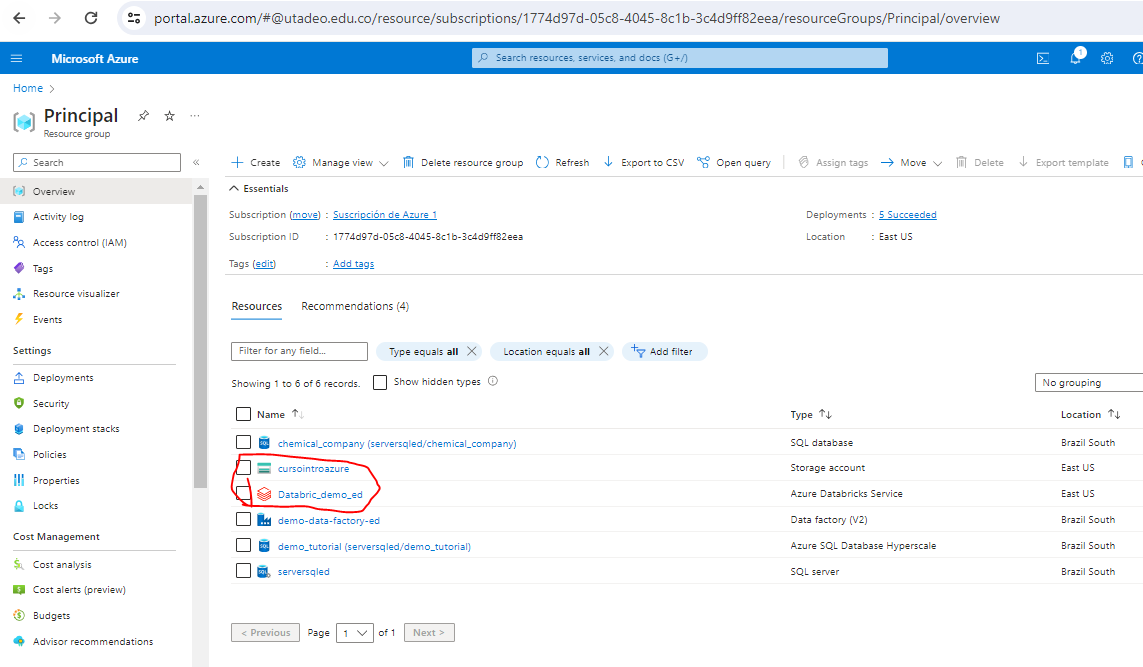


Interfaz de usuario gráfica, Aplicación

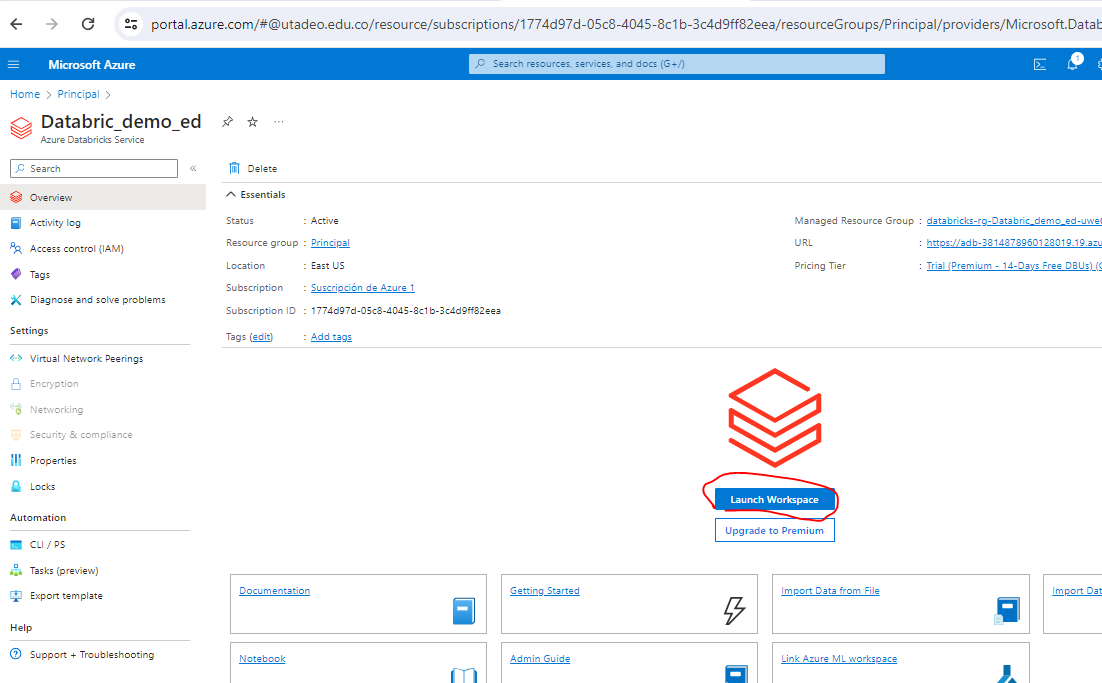
Descripción generada automáticamente



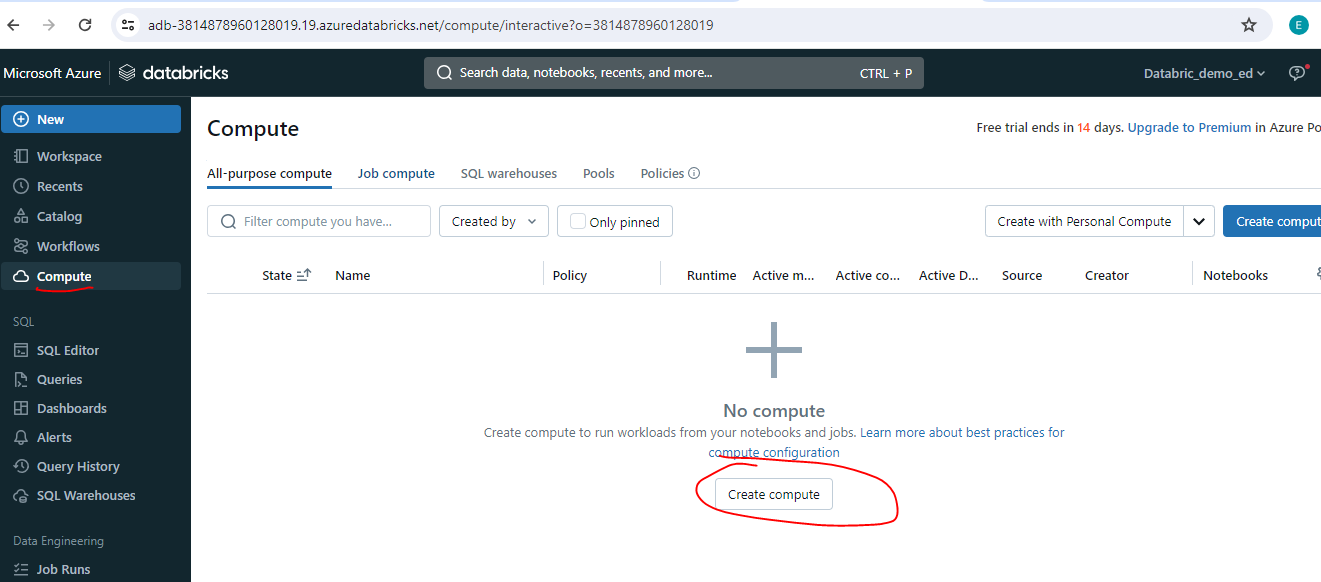
1. Crear un Storage

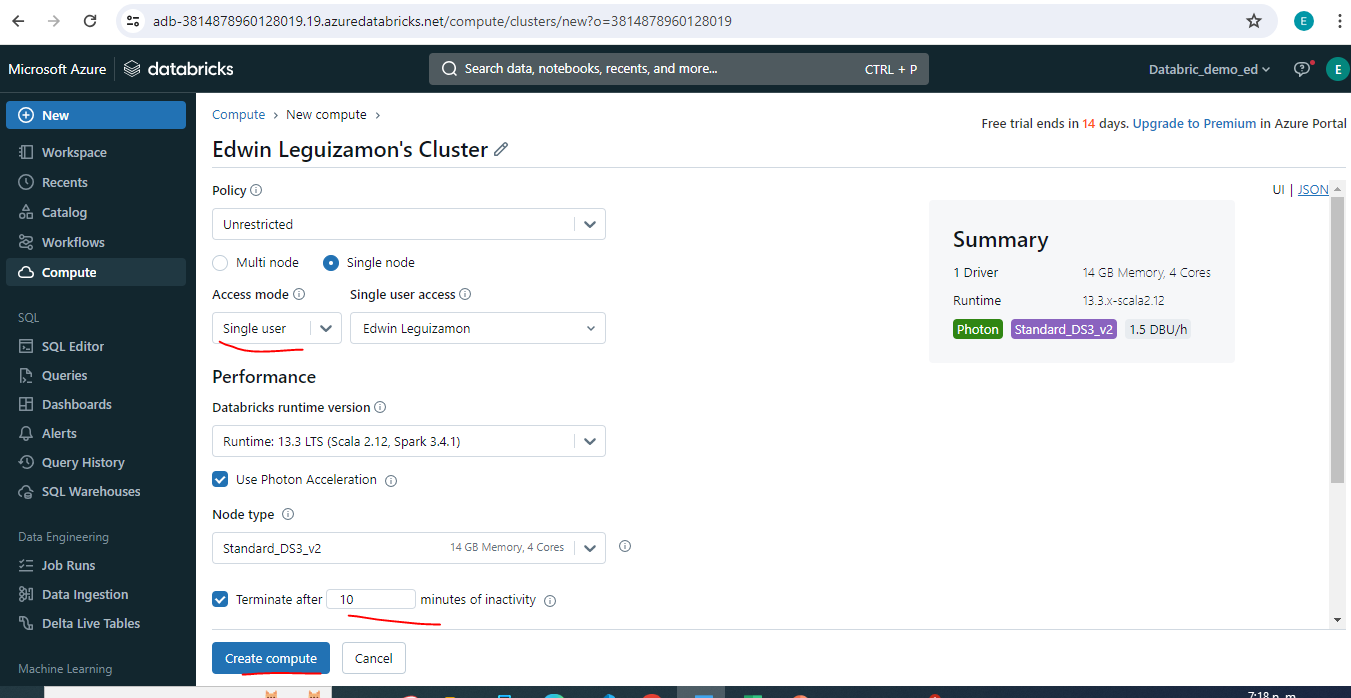


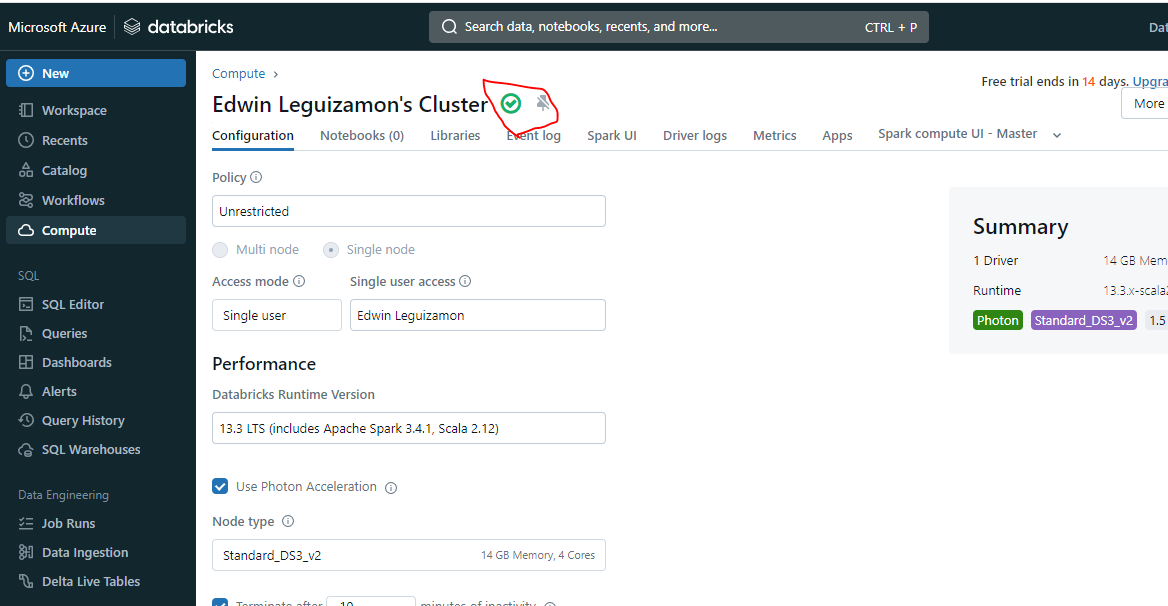
1. Ir al servicio de databrick



1. Crear un cluster (que se apague en 10 minutos)



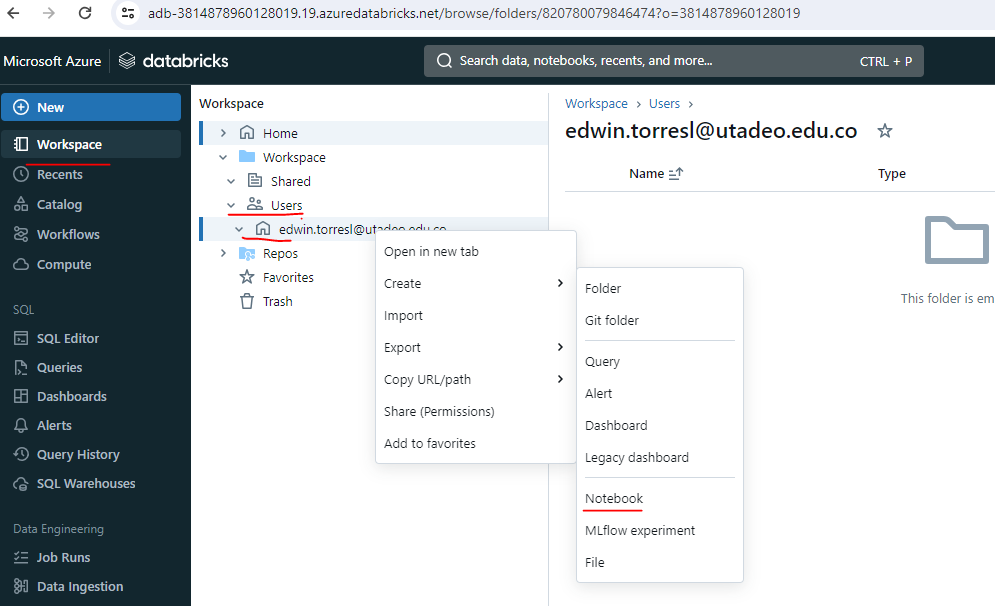




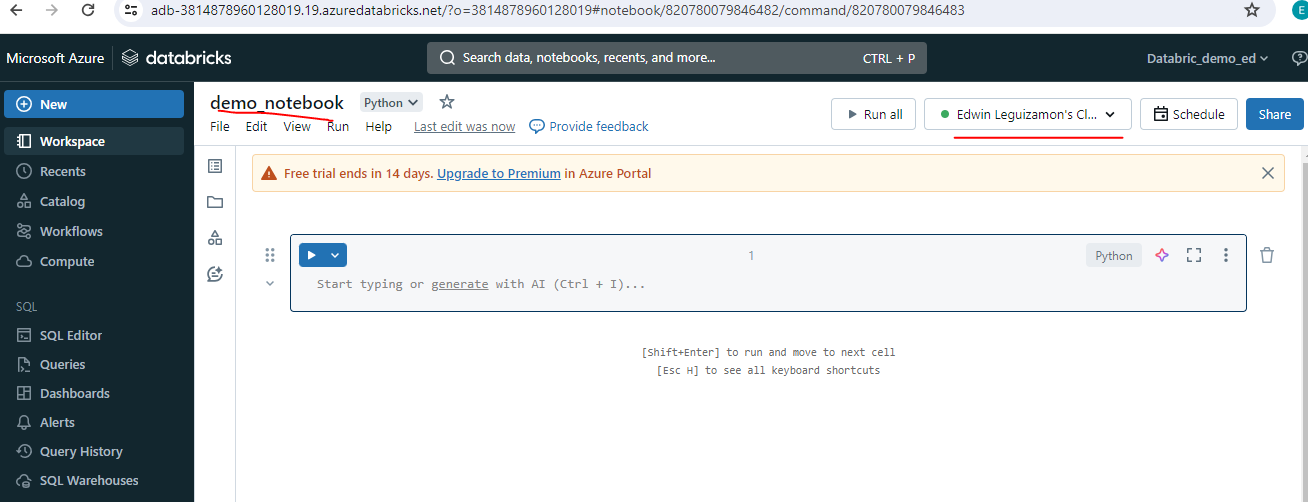
Una captura de pantalla de una red social

Descripción generada automáticamente

1. Crear un notebook:



**Primer notebook**



1. Con el cluster prendido importa la data de un API y convertirla en un dataframe

import requests

from pyspark.sql import SparkSession

import json

# Inicializar una sesión de Spark

spark = SparkSession.builder \

    .appName("API to Spark DataFrame") \

    .getOrCreate()

# URL que contiene los datos JSON

url = 'https://api.football-data.org/v2/competitions/2000/teams'

token = '9947b1e60e2a4d37b26ceb23fa3179e3'

headers = {

    'X-Auth-Token': token

}

try:

    # Realizar la solicitud GET a la URL con el token de autenticación

    response = requests.get(url, headers=headers)

    # Verificar si la solicitud fue exitosa y tiene contenido

    if response.status\_code == 200 and response.text:

        data = response.json()

        # Imprime los datos obtenidos de la API

        print(data)

        # Convertir los datos JSON a un DataFrame de Spark

        # Para esto, primero convertimos el JSON a un string

        json\_data = json.dumps(data['teams'])  # Asumiendo que 'teams' es la clave que contiene los datos relevantes

        # Crear un DataFrame de Spark a partir del string JSON

        df = spark.read.json(spark.sparkContext.parallelize([json\_data]))

        # Mostrar el esquema del DataFrame

        df.printSchema()

        # Mostrar algunas filas del DataFrame

        df.show()

        # Continuar con el procesamiento de los datos...

    else:

        print("La solicitud a la API no fue exitosa o la respuesta está vacía.")

        print("Código de estado de la respuesta:", response.status\_code)

        print("Contenido de la respuesta:", response.text)

except Exception as e:

    print("Ocurrió un error:", e)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

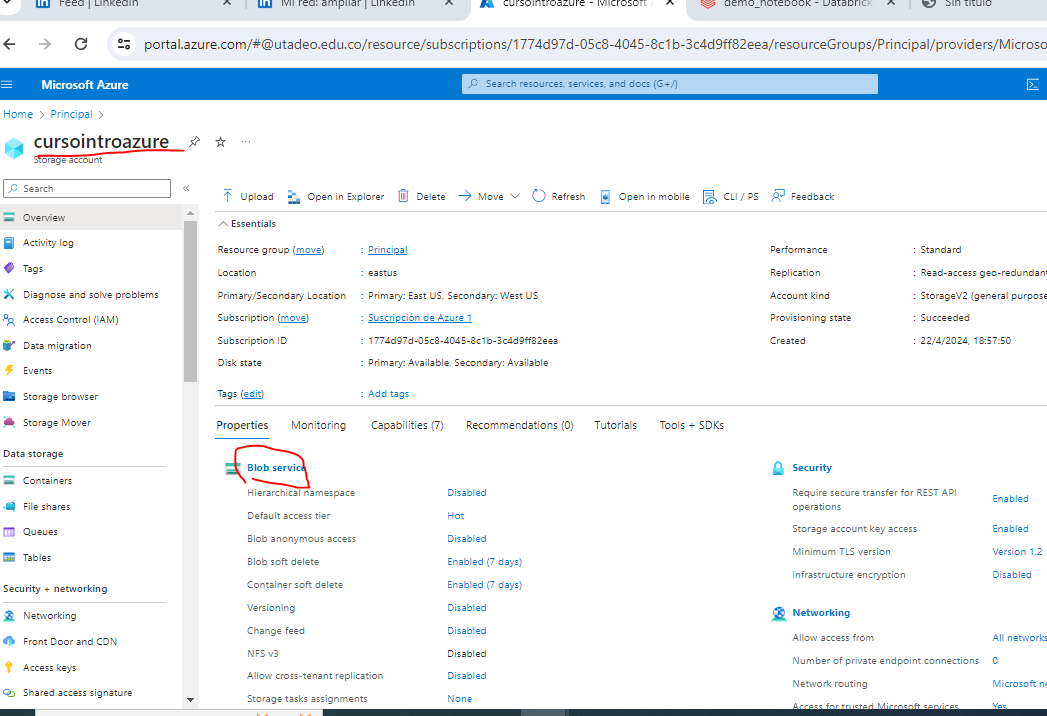
Descripción generada automáticamente

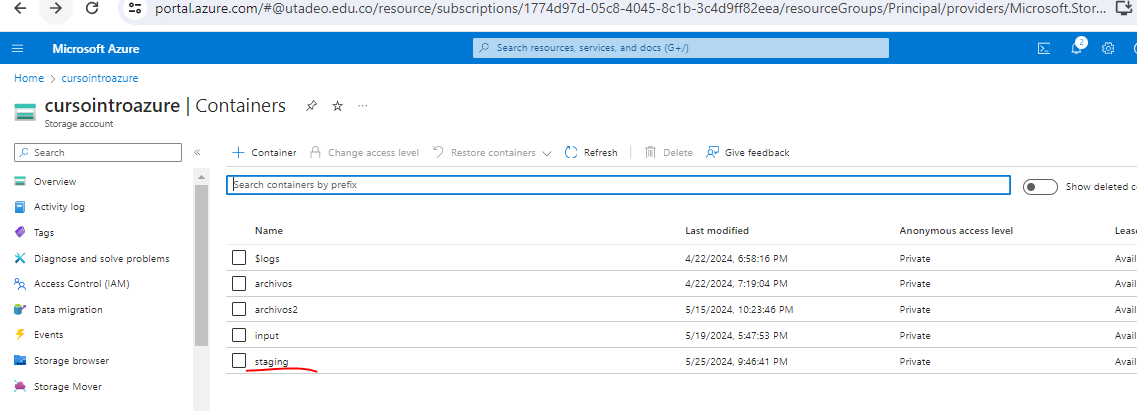
Una captura de pantalla de una computadora

Descripción generada automáticamente

1. Sql
2. Gráficos

**Segundo notebook**





Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

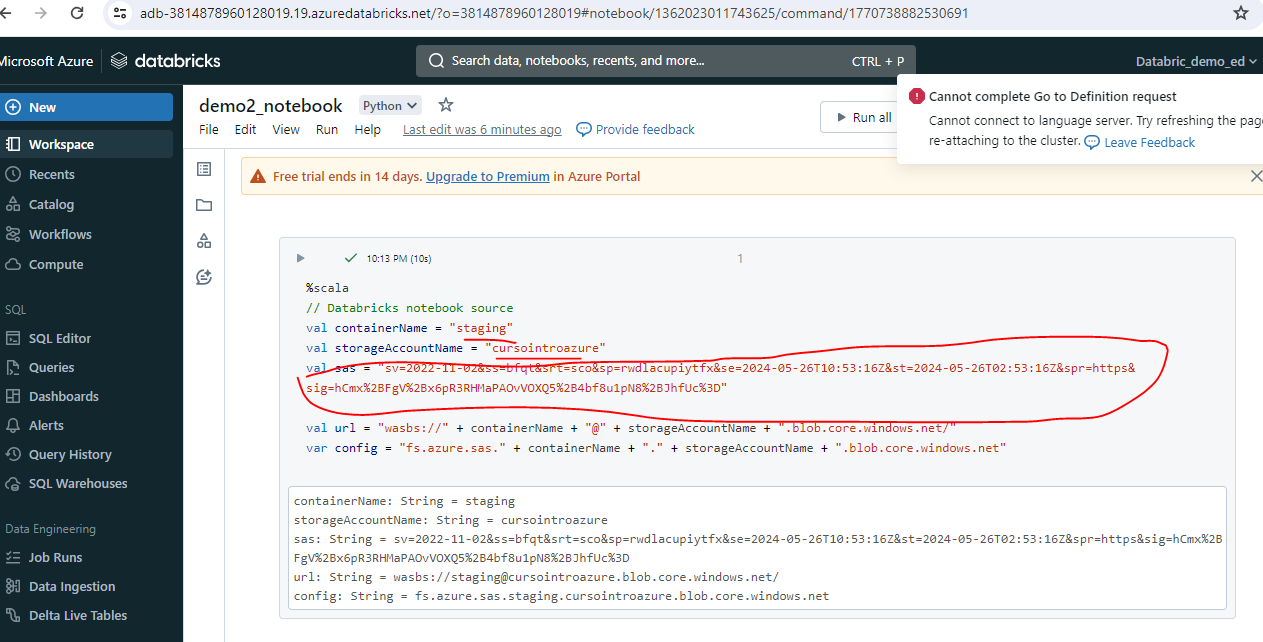
Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

// Databricks notebook source

val containerName = "<container\_name>"

val storageAccountName = "<storage\_account\_name>"

val sas = "<sas\_token>"

val url = "wasbs://" + containerName + "@" + storageAccountName + ".blob.core.windows.net/"

var config = "fs.azure.sas." + containerName + "." + storageAccountName + ".blob.core.windows.net"

// COMMAND ----------

dbutils.fs.mount(

source = url,

mountPoint = "/mnt/staging",

extraConfigs = Map(config -> sas))

// COMMAND ----------

val df = spark.read.json("/mnt/staging/small\_radio\_json.json")

display(df)

// COMMAND ----------

val specificColumnsDf = df.select("firstname", "lastname", "gender", "location", "level")

display(specificColumnsDf)

// COMMAND ----------

val renamedColumnsDF = specificColumnsDf.withColumnRenamed("level", "subscription\_type")

display(renamedColumnsDF)

// COMMAND ----------

renamedColumnsDF.createOrReplaceTempView("renamed")

// COMMAND ----------

// MAGIC %sql

// MAGIC SELECT

// MAGIC count(\*) as count,

// MAGIC subscription\_type

// MAGIC FROM renamed

// MAGIC GROUP BY

// MAGIC subscription\_type

// COMMAND ----------

val aggregate = spark.sql("""

SELECT

count(\*) as count,

subscription\_type

FROM renamed

GROUP BY

subscription\_type

""")

// COMMAND ----------

aggregate.write.mode("overwrite").csv("/mnt/staging/output/aggregate.csv")

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Aplicación

Descripción generada automáticamente