Data Security and Anonymization

Introduction

Data security and privacy are paramount concerns in today's digital world, particularly when dealing with sensitive information. Organizations must ensure that individuals' data is protected and anonymized to prevent unauthorized access and comply with privacy regulations.

This project focuses on data security and anonymization using PySparkSQL in Azure Databricks. The goal is to implement techniques to anonymize sensitive data and ensure compliance with privacy regulations within Azure Databricks. By leveraging PySparkSQL's capabilities, we can process and anonymize large datasets efficiently, making it suitable for real-world applications.

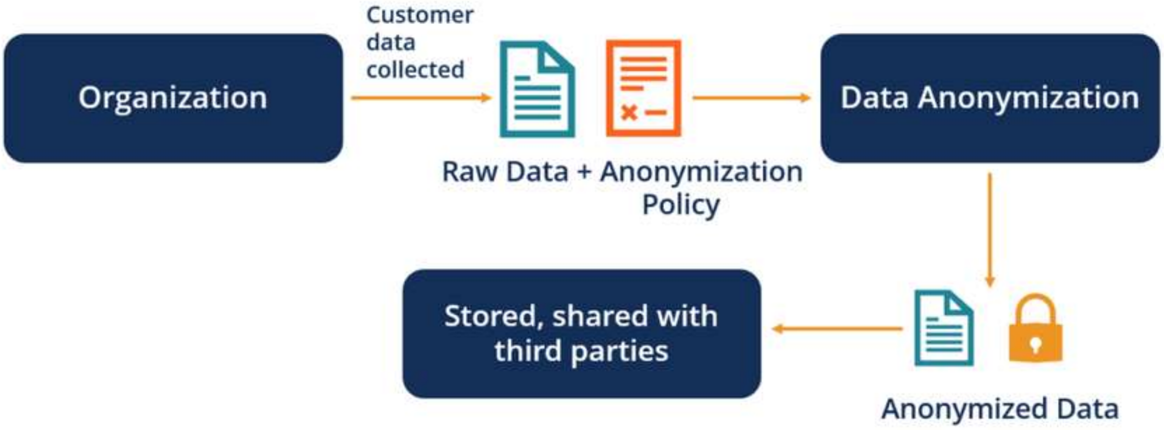
The project will involve generating a sample dataset containing sensitive information such as names, emails, SSNs, and phone numbers. We will then use PySparkSQL to anonymize this data, applying techniques such as hashing to protect the privacy of individuals' information.

Through this project, we aim to demonstrate how organizations can use PySparkSQL in Azure Databricks to enhance data security and privacy, ensuring that sensitive information is protected and compliance with privacy regulations is maintained.

Project Overview

The project aims to anonymize sensitive data in a sample dataset using PySparkSQL in Azure Databricks. This involves generating fake data, applying anonymization techniques such as hashing, and saving the anonymized data to a new CSV file. The goal is to protect the privacy of individuals' information while ensuring compliance with privacy regulations.

Architecture diagram



Execution Overview

1. **Setting up Azure Databricks:**
    - Create an Azure Databricks workspace.
    - Set up Azure Storage for data storage.
2. **Uploading Source Data:**
    - Upload the source data file containing sensitive information to Azure Databricks.
3. **Creating the PySparkSQL Script:**
    - Develop a PySparkSQL script that reads the source data, anonymizes it, and saves the anonymized data to a new CSV file.
4. **Executing the PySparkSQL Script:**
    - Run the PySparkSQL script in an Azure Databricks notebook to process the data.
5. **Analyzing the Results:**

- Review the output CSV file containing the anonymized data to ensure that the data has been successfully anonymized.

Key-components/Requirements of the Project

1. **Azure Databricks:** Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform. It provides an interactive workspace for data engineers, data scientists, and analysts to collaborate on big data and machine learning projects. In this project, Azure Databricks is used to run PySparkSQL scripts for data processing and anonymization.
2. **Azure Storage:** Azure Storage is a cloud storage solution that provides scalable, secure, and highly available storage for data. In this project, Azure Storage is used to store the source data file containing sensitive information and the output CSV file containing the anonymized data. Azure Storage provides durability and accessibility for the data files used in the project.
3. **Azure Databricks File System (DBFS):** Azure Databricks File System (DBFS) is a distributed file system that is compatible with the Hadoop Distributed File System (HDFS). It allows users to access and manage files stored in Azure Storage from within Azure Databricks. In this project, the output CSV file containing the anonymized data is saved to DBFS.
4. **Azure Databricks Cluster:** Azure Databricks Cluster is a group of virtual machines (VMs) that are used to process data in parallel. It provides the computing resources needed to execute PySparkSQL scripts and other data processing tasks. In this project, a Databricks Cluster is used to execute the PySparkSQL script for data anonymization.
5. **Azure Databricks Notebook:** Azure Databricks Notebook is an interactive environment for running code and visualizing data. It allows users to write and execute PySparkSQL scripts, Python code, and SQL queries. In this project, a Databricks Notebook is used to write and execute the PySparkSQL script for data anonymization.

Tasks Performed:

1. **Data Generation:** Generate a sample dataset with fake names, emails, SSNs, phone numbers, types of care, and insurance types using the Faker library.
2. **Data Anonymization:** Anonymize the **name**, **email**, **ssn**, and **phone_number** columns using MD5 hashing to protect sensitive information.
3. **Spark Implementation:** Use PySparkSQL to read the sample dataset, apply anonymization techniques, and write the anonymized data to a new CSV file.
4. **File Input/Output:** Read the sample dataset from a CSV file, anonymize the data, and save the anonymized data to a new CSV file.

Technologies/Tools Used

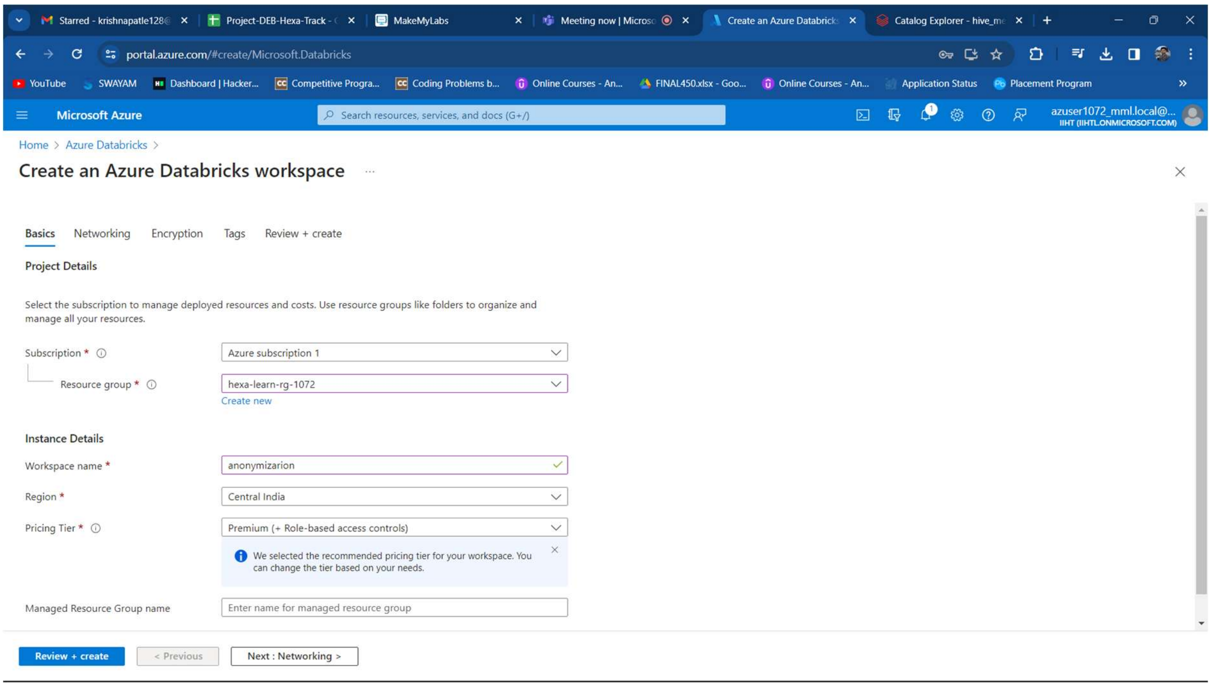
1. **Azure Databricks:** Azure Databricks is a cloud-based analytics platform built on Apache Spark. It provides a collaborative environment for data scientists, data engineers, and business analysts to work together on big data and machine learning projects. In this project, Azure Databricks is used for running PySparkSQL scripts to process and anonymize data.
2. **PySparkSQL:** PySparkSQL is the Python API for Spark SQL, which is a module in Apache Spark for processing structured data. PySparkSQL allows users to run SQL queries and manipulate data using DataFrame APIs. In this project, PySparkSQL is used to read, process, and anonymize the source data.
3. **Faker:** Faker is a Python library that generates fake data such as names, addresses, and phone numbers. It is used in this project to create a sample dataset for testing and demonstration purposes.
4. **Hashlib:** Hashlib is a Python library that provides cryptographic hashing functions. In this project, hashlib is used to hash sensitive data such as names, emails, SSNs, and phone numbers for anonymization.
5. **Azure Storage:** Azure Storage is a cloud storage solution provided by Microsoft Azure. It is used in this project to store the source data file and

the anonymized data file. Azure Storage provides scalable, secure, and reliable storage for big data applications.

How It works

1. 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.


portal.azure.com/#create/Microsoft.Databricks
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Azure Databricks >
Basics
Networking
Encryption
Tags
Review + create
Review + create
< Previous
Next : Networking >

Create an Azure Databricks workspace

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: Azure subscription 1

Resource group *: hexa-learn-rg-1072

Create new

Instance Details

Workspace name *: anonymizarion

Region *: Central India

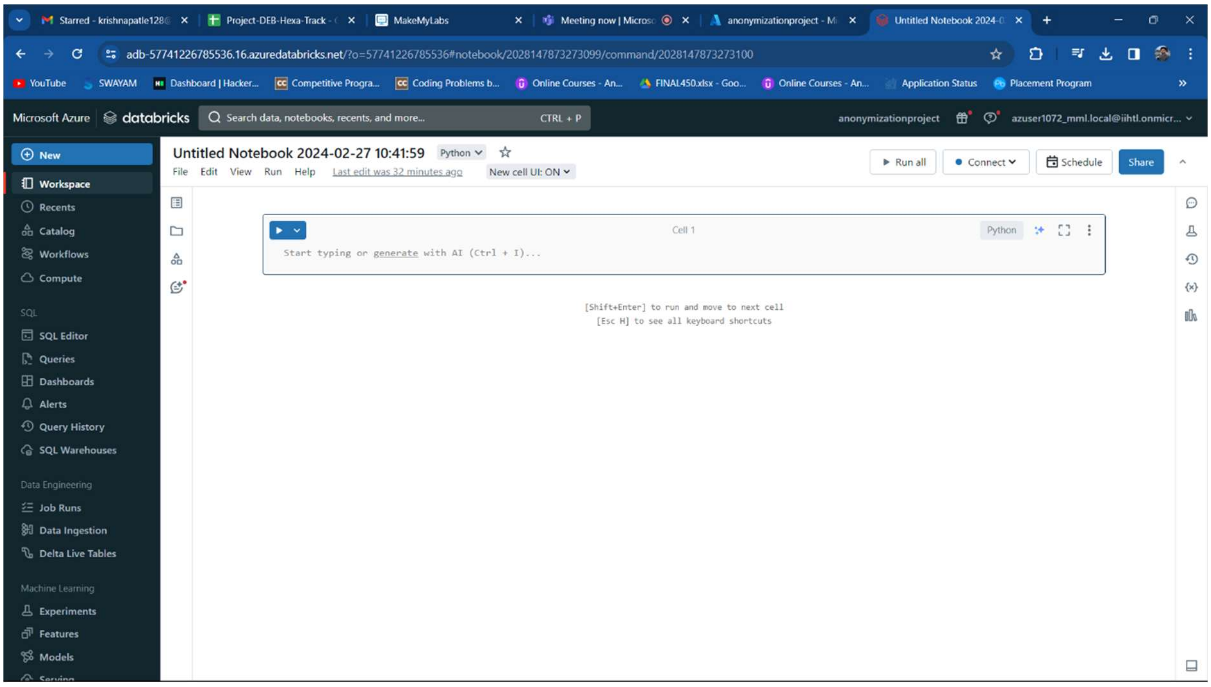
Pricing Tier *: Premium (+ Role-based access controls)

We selected the recommended pricing tier for your workspace. You can change the tier based on your needs.

Managed Resource Group name: Enter name for managed resource group

2. 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.



adb-577412267855536.16.azuredatabricks.net/?o=577412267855536#notebook/2028147873273099/command/2028147873273100

Untitled Notebook 2024-02-27 10:41:59

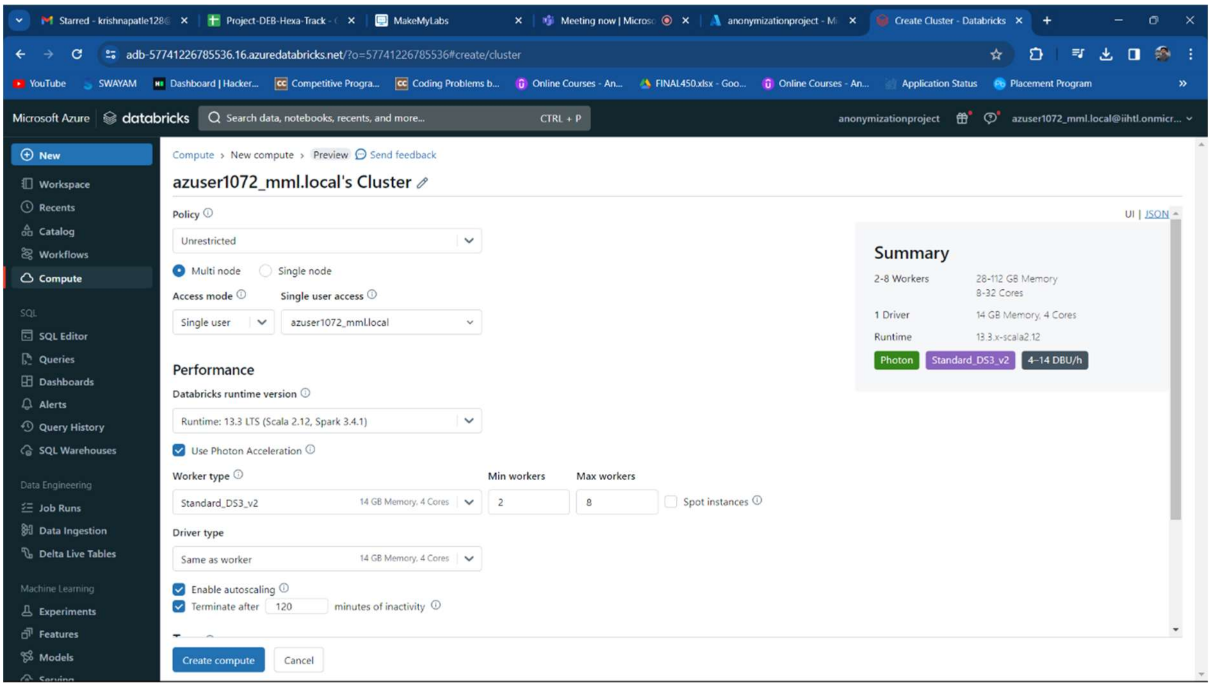
Cell 1

```
Start typing or generate with AI (Ctrl + I)...
```

```
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
```

3. <u>Create Cluster and Connecting to notebook</u>

- 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.



adb-577412267855536.16.azuredatabricks.net/?o=577412267855536#create/cluster

Compute › New compute › Preview

azuser1072_mml.local's Cluster

Policy: Unrestricted

● Multi node ○ Single node

Access mode: Single user

Single user access: azuser1072_mml.local

Performance

Databricks runtime version: Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)

☑ Use Photon Acceleration

Worker type: Standard_DS3_v2 14 GB Memory, 4 Cores

Min workers: 2

Max workers: 8

□ Spot instances

Driver type: Same as worker 14 GB Memory, 4 Cores

☑ Enable autoscaling

☑ Terminate after 120 minutes of inactivity

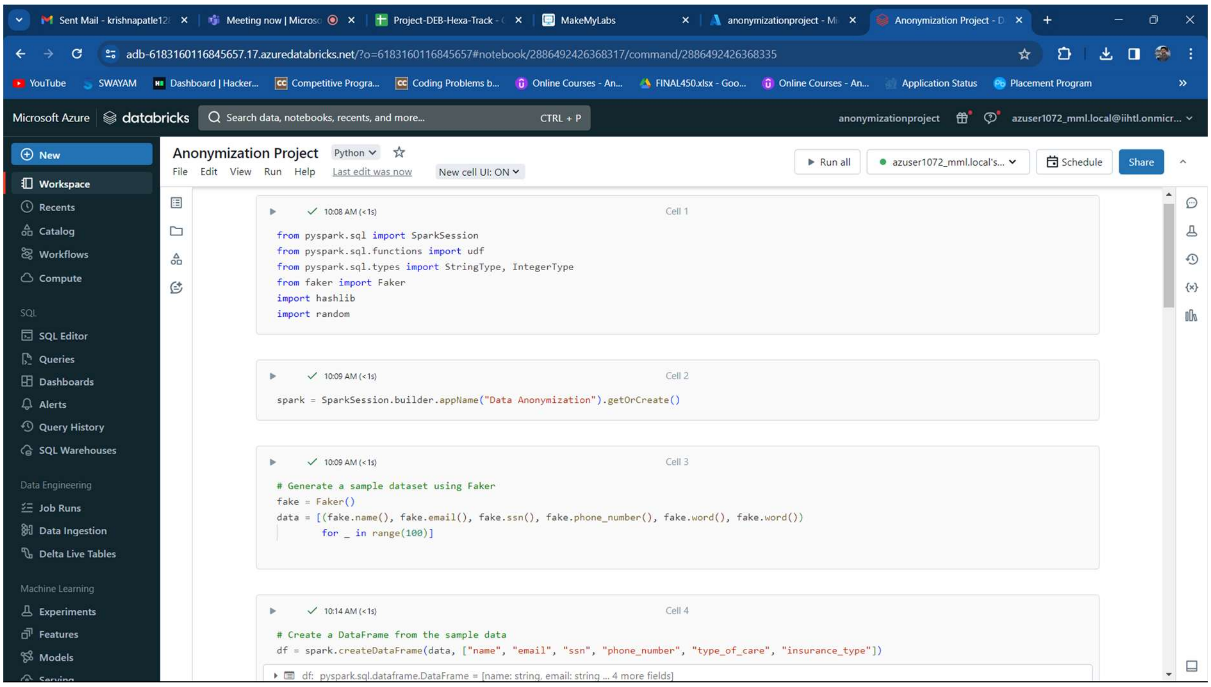
Summary

2-8 Workers	28-112 GB Memory 8-32 Cores
1 Driver	14 GB Memory, 4 Cores
Runtime	13.3.x-scala2.12

Photon Standard_DS3_v2 4-14 DBU/h

4. 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



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, IntegerType
from faker import Faker
import hashlib
import random
```

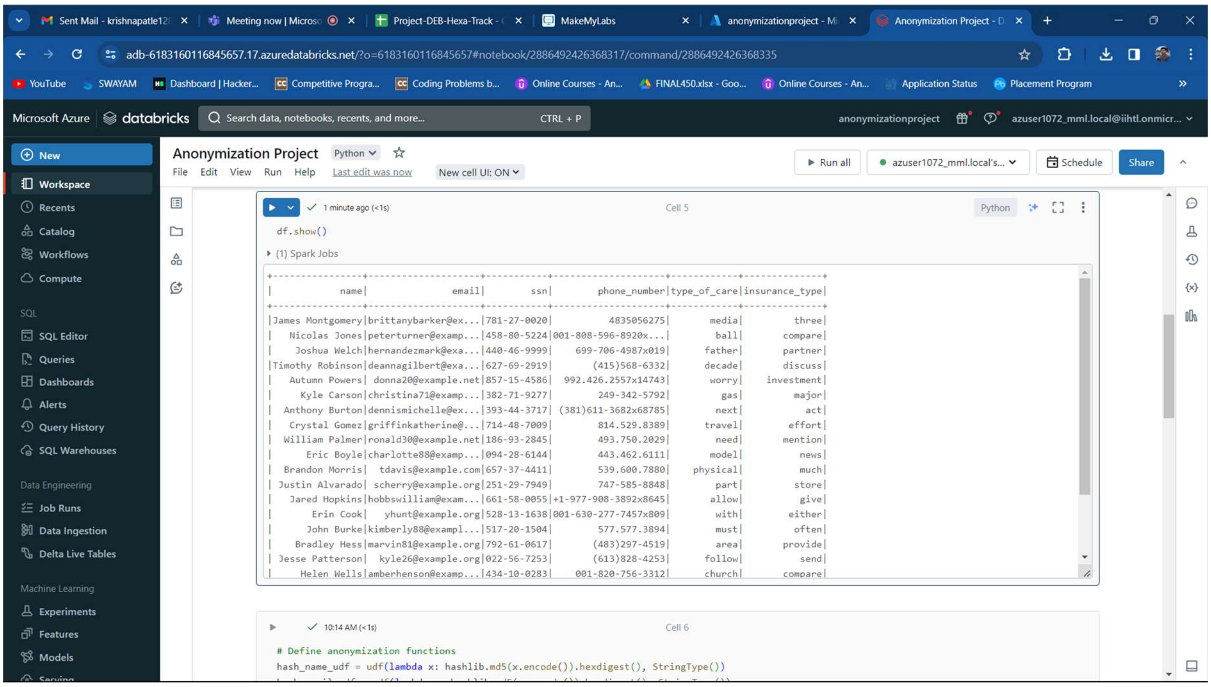
```
spark = SparkSession.builder.appName("Data Anonymization").getOrCreate()
```

```
# Generate a sample dataset using Faker
fake = Faker()
data = [(fake.name(), fake.email(), fake.ssn(), fake.phone_number(), fake.word(), fake.word())
        for _ in range(100)]
```

```
# Create a DataFrame from the sample data
df = spark.createDataFrame(data, ["name", "email", "ssn", "phone_number", "type_of_care", "insurance_type"])
```

df: pyspark.sql.dataframe.DataFrame = [name: string, email: string ... 4 more fields]

5. Generate a sample dataset using Faker and apply anonymization techniques:

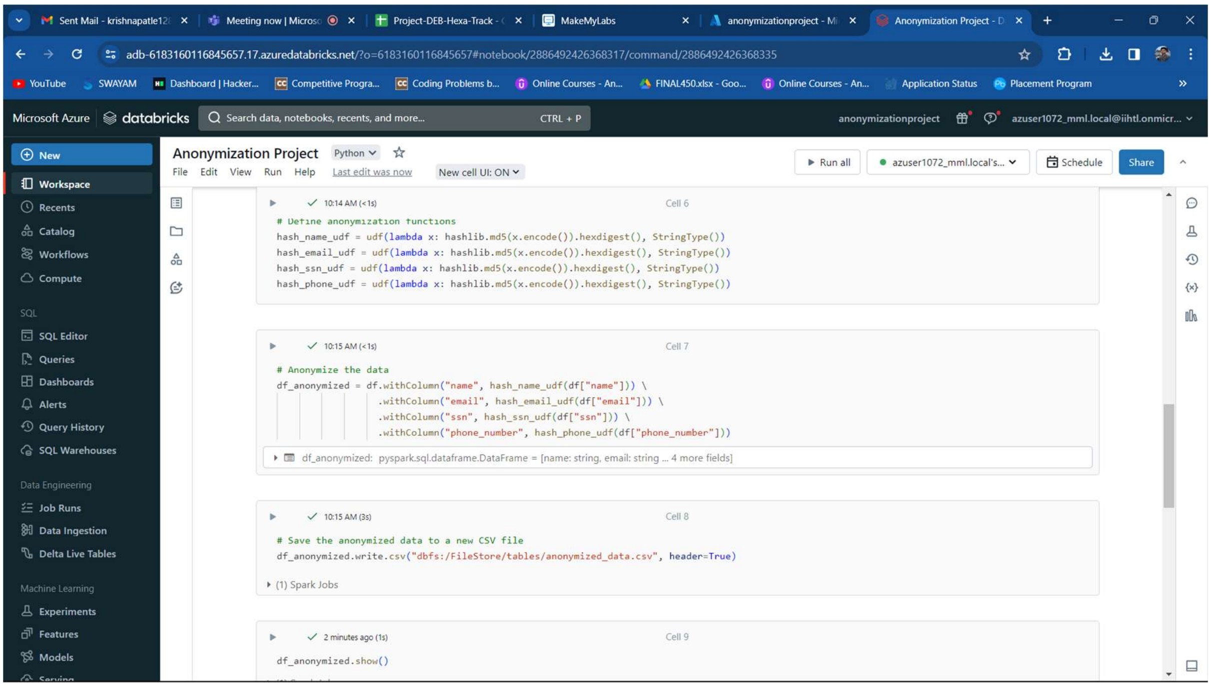


```
df.show()
```

(1) Spark Jobs

```
+---------------+--------------------+-----------+--------------------+------------+--------------+
|           name|               email|        ssn|        phone_number|type_of_care|insurance_type|
+---------------+--------------------+-----------+--------------------+------------+--------------+
|James Montgomery|brittanybarker@ex...|781-27-0020|          4835056275|       media|         three|
|   Nicolas Jones|peterturner@examp...|458-80-5224|001-808-596-8920x...|        ball|       compare|
|    Joshua Welch|hernandezmark@exa...|440-46-9999|   699-706-4987x019|      father|       partner|
|Timothy Robinson|deannagilbert@exa...|627-69-2919|       (415)568-6332|      decade|       discuss|
|   Autumn Powers| donna20@example.net|857-15-4586|  992.426.2557x14743|       worry|    investment|
|     Kyle Carson|christina71@examp...|382-71-9277|        249-342-5792|         gas|         major|
|  Anthony Burton|dennismichelle@ex...|393-44-3717| (381)611-3682x68785|        next|           act|
|   Crystal Gomez|griffinkatherine@...|714-48-7009|        814.529.8389|      travel|        effort|
|  William Palmer|ronald30@example.net|186-93-2845|        493.750.2029|        need|       mention|
|      Eric Boyle|charlotte88@examp...|094-28-6144|        443.462.6111|       model|          news|
|  Brandon Morris|   tdavis@example.com|657-37-4411|        539.600.7880|    physical|          much|
| Justin Alvarado|  scherry@example.org|251-29-7949|        747-585-8848|        part|         store|
|   Jared Hopkins|hobbswilliam@exam...|661-58-0055|+1-977-908-3892x8645|       allow|          give|
|       Erin Cook|    yhunt@example.org|528-13-1638|001-630-277-7457x809|        with|        either|
|      John Burke|kimberly88@exampl...|517-20-1504|        577.577.3894|        must|         often|
|    Bradley Hess|marvin81@example.org|792-61-0617|       (483)297-4519|        area|       provide|
| Jesse Patterson|   kyle26@example.org|022-56-7253|       (613)828-4253|      follow|          send|
|     Helen Wells|amberhenson@examp...|434-10-0283|     001-820-756-3312|      church|       compare|
```

```
# Define anonymization functions
hash_name_udf = udf(lambda x: hashlib.md5(x.encode()).hexdigest(), StringType())
```



10:14 AM (<1s) Cell 6

```python
# Define anonymization functions
hash_name_udf = udf(lambda x: hashlib.md5(x.encode()).hexdigest(), StringType())
hash_email_udf = udf(lambda x: hashlib.md5(x.encode()).hexdigest(), StringType())
hash_ssn_udf = udf(lambda x: hashlib.md5(x.encode()).hexdigest(), StringType())
hash_phone_udf = udf(lambda x: hashlib.md5(x.encode()).hexdigest(), StringType())
```

10:15 AM (<1s) Cell 7

```python
# Anonymize the data
df_anonymized = df.withColumn("name", hash_name_udf(df["name"])) \
                .withColumn("email", hash_email_udf(df["email"])) \
                .withColumn("ssn", hash_ssn_udf(df["ssn"])) \
                .withColumn("phone_number", hash_phone_udf(df["phone_number"]))
```

df_anonymized: pyspark.sql.dataframe.DataFrame = [name: string, email: string ... 4 more fields]

10:15 AM (3s) Cell 8

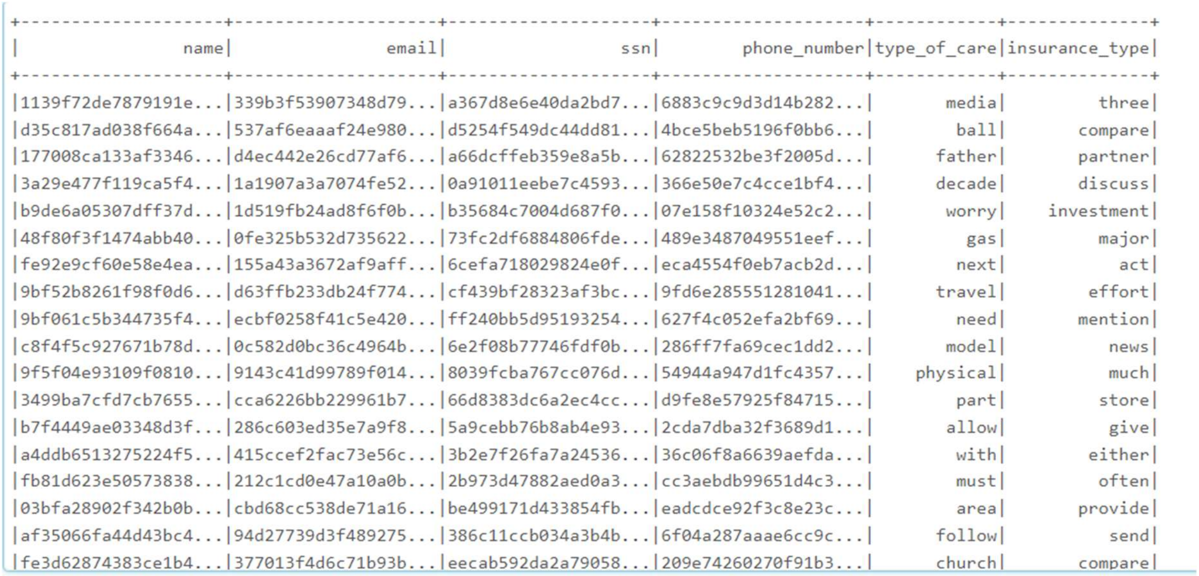
```python
# Save the anonymized data to a new CSV file
df_anonymized.write.csv("dbfs:/FileStore/tables/anonymized_data.csv", header=True)
```

(1) Spark Jobs

2 minutes ago (1s) Cell 9

```python
df_anonymized.show()
```

6. Project Output:



```
+--------------------+--------------------+--------------------+--------------------+------------+--------------+
|                name|               email|                 ssn|        phone_number|type_of_care|insurance_type|
+--------------------+--------------------+--------------------+--------------------+------------+--------------+
|1139f72de7879191e...|339b3f53907348d79...|a367d8e6e40da2bd7...|6883c9c9d3d14b282...|       media|         three|
|d35c817ad038f664a...|537af6eaaaf24e980...|d5254f549dc44dd81...|4bce5beb5196f0bb6...|        ball|       compare|
|177008ca133af3346...|d4ec442e26cd77af6...|a66dcffeb359e8a5b...|62822532be3f2005d...|      father|       partner|
|3a29e477f119ca5f4...|1a1907a3a7074fe52...|0a91011eebe7c4593...|366e50e7c4cce1bf4...|      decade|       discuss|
|b9de6a05307dff37d...|1d519fb24ad8f6f0b...|b35684c7004d687f0...|07e158f10324e52c2...|       worry|    investment|
|48f80f3f1474abb40...|0fe325b532d735622...|73fc2df6884806fde...|489e3487049551eef...|         gas|         major|
|fe92e9cf60e58e4ea...|155a43a3672af9aff...|6cefa718029824e0f...|eca4554f0eb7acb2d...|        next|           act|
|9bf52b8261f98f0d6...|d63ffb233db24f774...|cf439bf28323af3bc...|9fd6e285551281041...|      travel|        effort|
|9bf061c5b344735f4...|ecbf0258f41c5e420...|ff240bb5d95193254...|627f4c052efa2bf69...|        need|       mention|
|c8f4f5c927671b78d...|0c582d0bc36c4964b...|6e2f08b77746fdf0b...|286ff7fa69cec1dd2...|       model|          news|
|9f5f04e93109f0810...|9143c41d99789f014...|8039fcba767cc076d...|54944a947d1fc4357...|    physical|          much|
|3499ba7cfd7cb7655...|cca6226bb229961b7...|66d8383dc6a2ec4cc...|d9fe8e57925f84715...|        part|         store|
|b7f4449ae03348d3f...|286c603ed35e7a9f8...|5a9cebb76b8ab4e93...|2cda7dba32f3689d1...|       allow|          give|
|a4ddb6513275224f5...|415ccef2fac73e56c...|3b2e7f26fa7a24536...|36c06f8a6639aefda...|        with|        either|
|fb81d623e50573838...|212c1cd0e47a10a0b...|2b973d47882aed0a3...|cc3aebdb99651d4c3...|        must|         often|
|03bfa28902f342b0b...|cbd68cc538de71a16...|be499171d433854fb...|eadcdce92f3c8e23c...|        area|       provide|
|af35066fa44d43bc4...|94d27739d3f489275...|386c11ccb034a3b4b...|6f04a287aaae6cc9c...|      follow|          send|
|fe3d62874383ce1b4...|377013f4d6c71b93b...|eecab592da2a79058...|209e74260270f91b3...|      church|       compare|
```

Conclusion:

By applying anonymization techniques to the sample dataset, we have successfully protected sensitive information such as names, emails, SSNs, and phone numbers. This project demonstrates how PySparkSQL can be used for data anonymization in compliance with privacy regulations.