INTERIM PROJECT

Topic: Streaming ETL: Perform Extract-Transform-Load (ETL) operations on streaming data, transforming it into a more structured format or loading it into a database.

Submitted by

Sanjeev M

EMP ID: 2320436

Abstract:

Streaming Extract-Transform-Load (ETL) operations have become essential in modern data processing pipelines, especially with the rise of real-time data analytics and decision-making. This paper explores the concept of Streaming ETL, which involves extracting data from continuous streams, transforming it on-the-fly, and loading it into a structured format or database for further analysis. The primary objective of Streaming ETL is to handle high-volume, high-velocity data streams efficiently, ensuring timely and accurate data processing.

The abstract further delves into the key components and challenges of Streaming ETL. It discusses the importance of stream processing frameworks like Apache Kafka, Apache Flink, and Apache Storm for handling streaming data ingestion and processing. Additionally, it highlights the significance of scalable data transformation techniques, such as stream-based transformations using Apache Beam or Spark Streaming, to handle complex data manipulations in real-time.

Furthermore, the abstract touches upon the role of destination systems such as relational databases, data warehouses, or data lakes in storing and organizing the transformed data for downstream applications. It emphasizes the need for robust error handling, data quality checks, and monitoring mechanisms in Streaming ETL workflows to ensure data integrity and reliability.

Introduction:

In today's data-driven world, the ability to process data in both real-time and batch modes is crucial for organizations aiming to extract valuable insights and make informed decisions. Streaming Extract-Transform-Load (ETL) and batch ETL are two essential methodologies that enable organizations to process data continuously as it flows in real-time or in larger, periodic batches. These methodologies involve extracting raw data, transforming it to a more structured format, and loading it into a database or data warehouse for further analysis.

Table of Contents

Overview:

Streaming ETL (Real-Time):

Real-time data ingestion using Apache Kafka from sensors and IoT devices in stores capturing sales data.
Apache Spark Streaming is used for real-time data processing and transformations on the streaming data.
Transformations include enrichment, aggregation, fraud detection, and inventory management as described earlier.
Enriched and transformed data is loaded into a data warehouse or data lake in near real-time for immediate analysis and reporting.

Batch ETL (Near Real-Time):

In addition to real-time processing, the retail company also performs batch ETL processes at regular intervals (e.g., daily, hourly) to handle historical data and larger volumes.
Batch ETL jobs are scheduled using Apache Airflow or similar workflow management tools to process data in batches.
The batch ETL pipeline includes similar transformations as the real-time pipeline but operates on larger datasets and historical data.
Processed batch data is also loaded into the data warehouse or data lake, merging seamlessly with the real-time data for comprehensive analytics.

Real-Time Streaming ETL

Real-time customer segmentation based on current purchases and behaviours.
Instantaneous fraud detection and prevention for immediate action.
Dynamic pricing adjustments based on real-time market demand signals.
Live inventory management and restocking notifications for store managers.
Batch ETL:

Historical trend analysis to identify seasonal patterns and long-term customer trends.
Large-scale data aggregation for quarterly and annual business reviews.
Regulatory compliance reporting requiring historical data analysis.
Data archiving and backup for long-term data retention and disaster recovery.

Streaming ETL involves processing data in near real-time as it is generated, enabling organizations to derive insights and respond to events promptly. On the other hand, batch ETL processes data in larger volumes and at scheduled intervals, allowing for deeper analysis of historical data and trend identification. Both methodologies play complementary roles in modern data architectures, addressing different use cases and requirements.

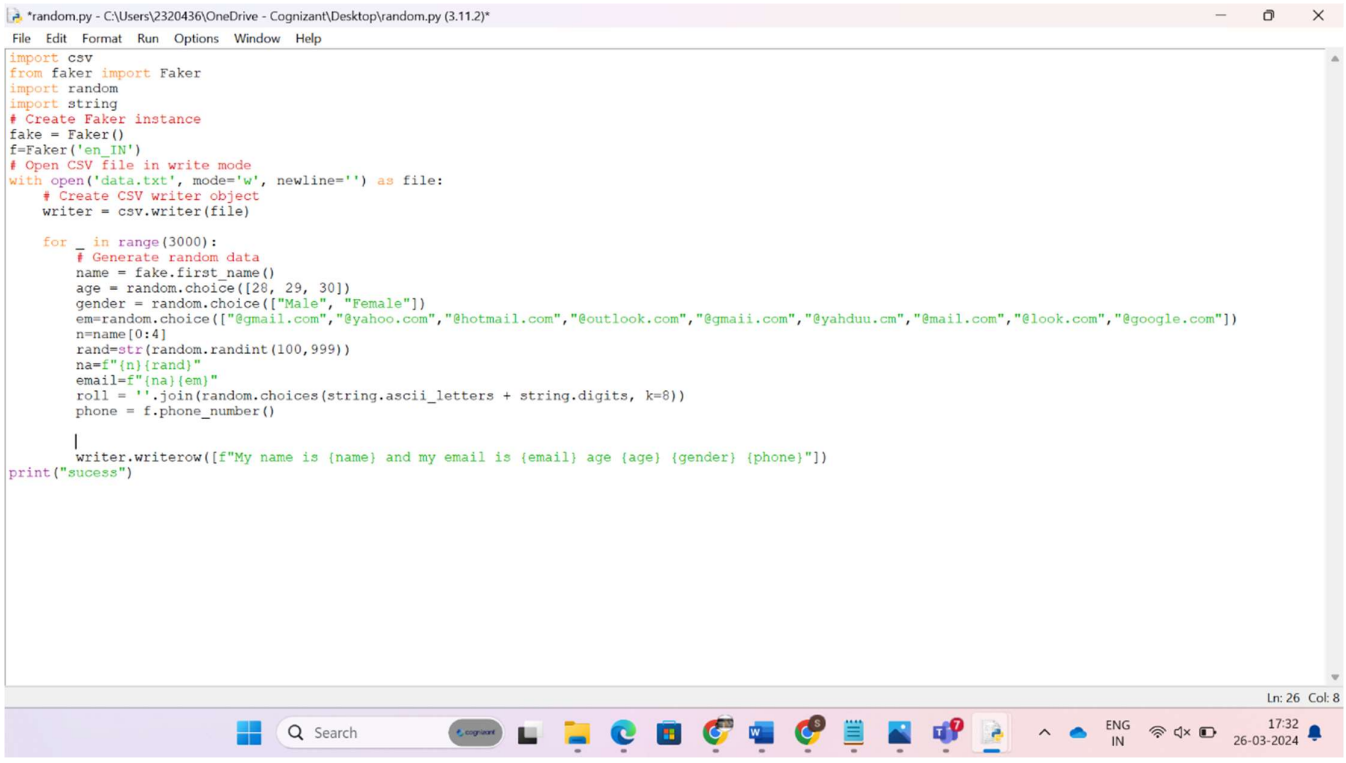
Key Components:

1. Data Ingestion: Both streaming and batch ETL begin with data ingestion, where raw data is collected from various sources such as sensors, applications, databases, and external APIs. Streaming ETL uses technologies like Apache Kafka, Amazon Kinesis, or Azure Event Hubs to ingest and buffer real-time data streams. Batch ETL typically involves bulk data extraction from databases, files, or data lakes at predefined intervals.
2. Transformation: After data ingestion, the raw data undergoes transformations to clean, enrich, aggregate, and structure it for analysis. Streaming ETL employs stream processing frameworks like Apache Flink, Apache Spark Streaming, or Apache Storm to perform transformations on-the-fly as data arrives. Batch ETL uses batch processing frameworks like Apache Spark, Apache Hadoop, or Apache Beam to process data in larger chunks and perform complex transformations.
3. Loading: The transformed data is then loaded into a destination system such as a relational database, data warehouse, or data lake for storage and analysis. Both streaming and batch ETL pipelines integrate seamlessly with various storage and processing systems, ensuring that the transformed data is available for downstream analytics, reporting, and application integration.

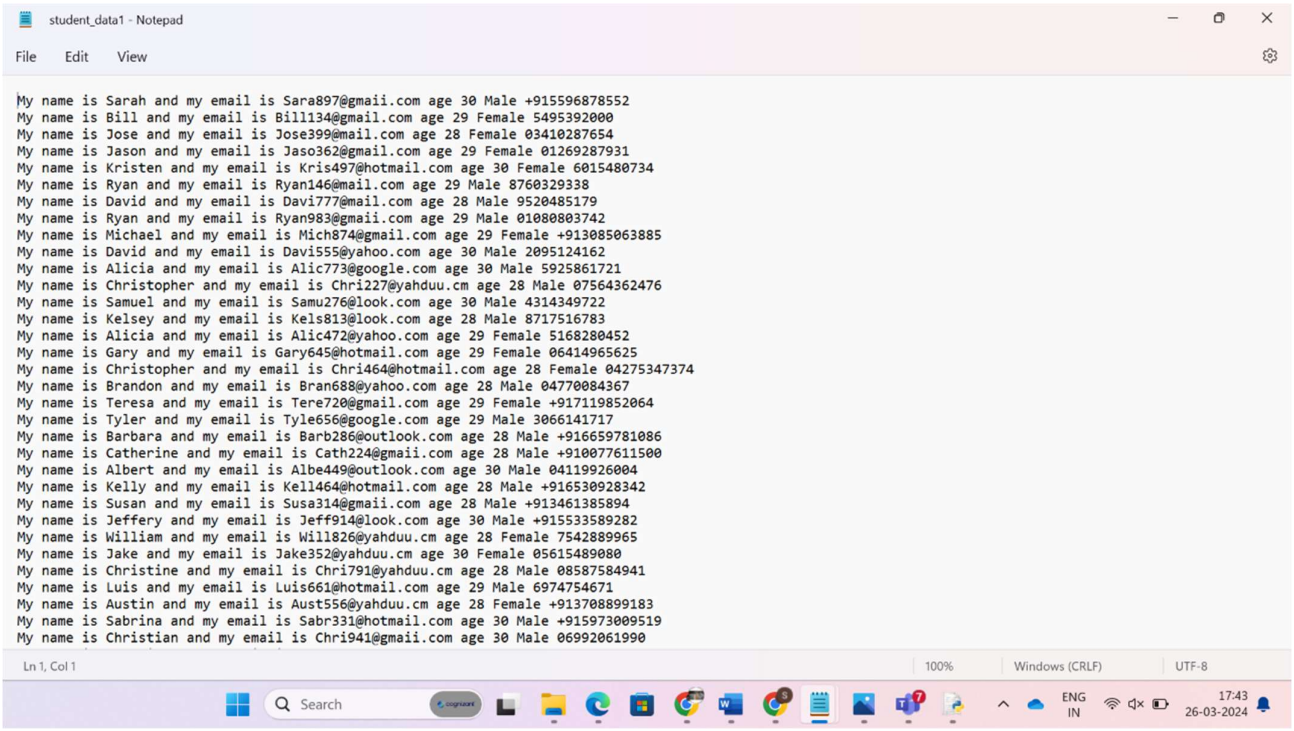
CHAPTER 1- Data Ingestion

Data ingestion is the process of importing data from various sources into the system for analysis. In this project, the Unstructured dataset is ingested into the system using Apache Spark, specifically PySpark, which is a powerful data processing framework. PySpark provides functionality to read data from various sources, including CSV files, databases, and distributed storage systems.

For this project, using Python, I generated the student dataset is generated and stored in a CSV file, and PySpark's DataFrame API is used to read the data from this file into a DataFrame for further analysis.



Here input file is in unstructured format. Our goal is unstructured to structured format using spark. In both batch processing and stream processing we are going to do.

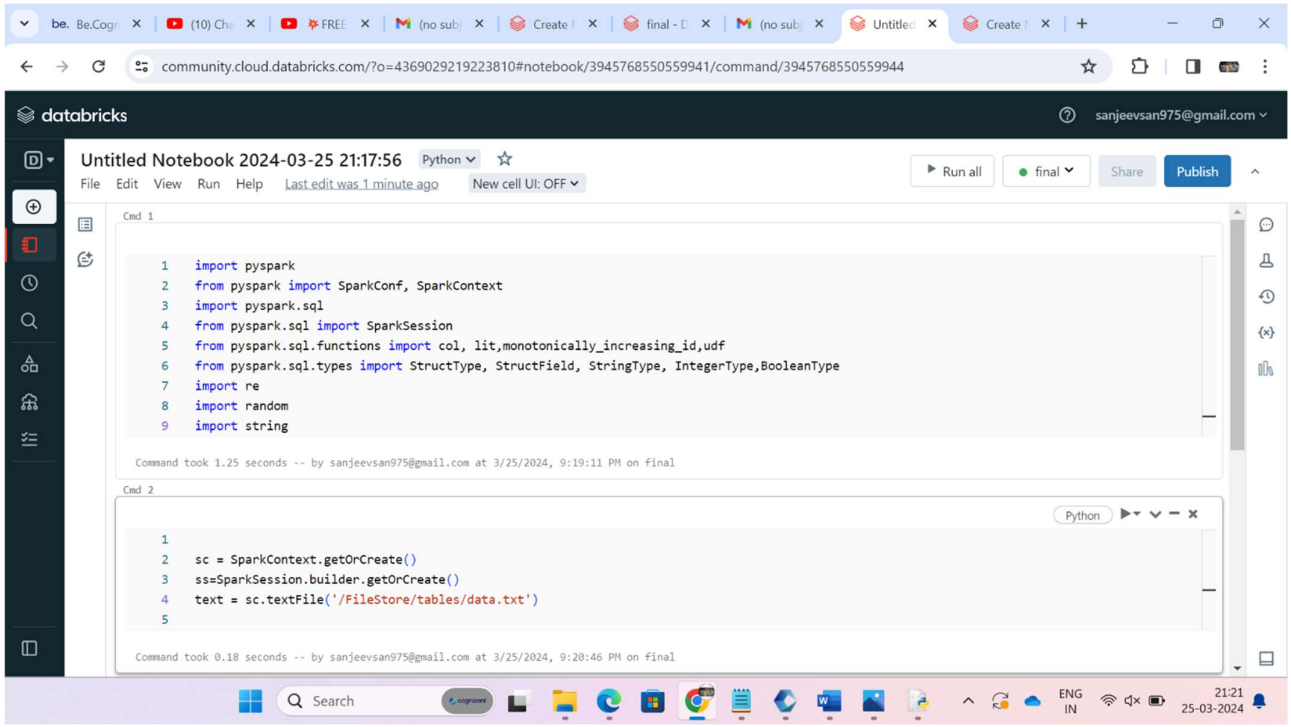


Chapter 2

BATCH STREAMING USING ETL:

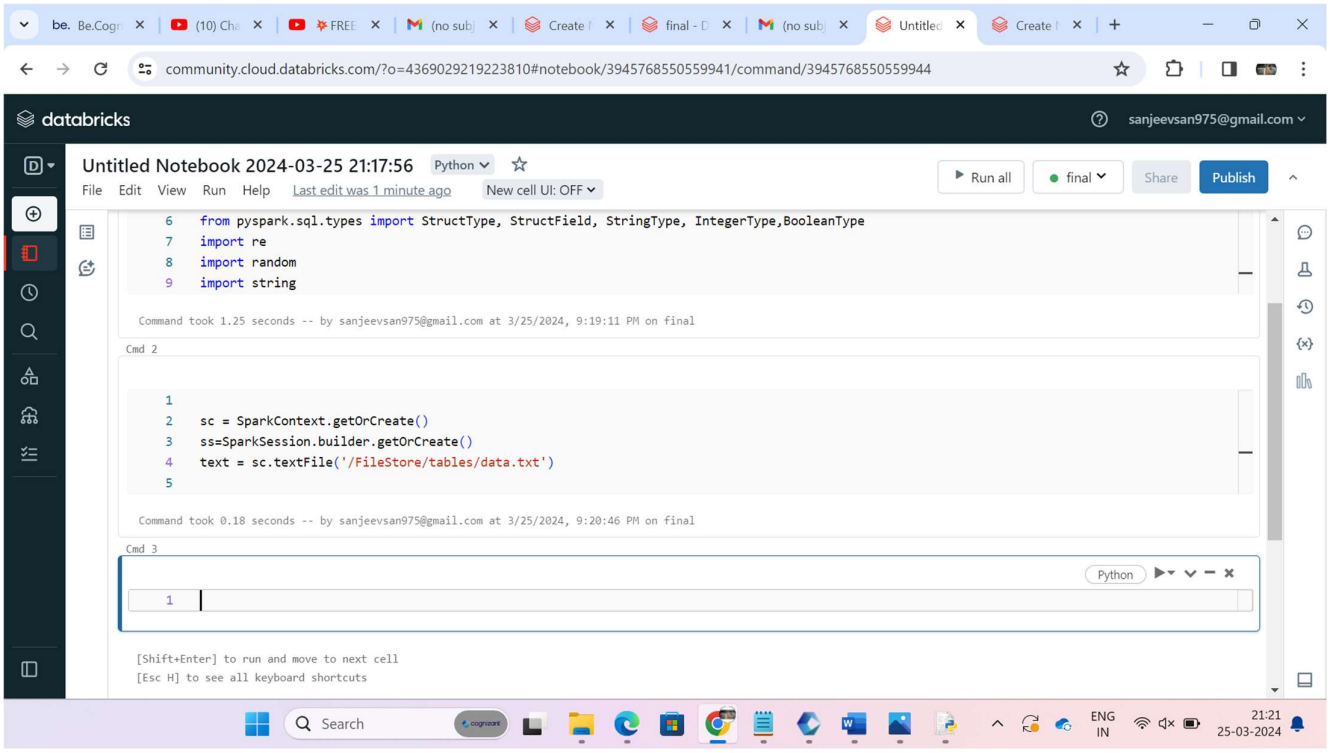
Importing Libraries:

The code imports necessary libraries for working with Spark, including `pyspark`, `SparkConf`, `SparkContext`, `SparkSession`, and various functions and types from `pyspark.sql`.



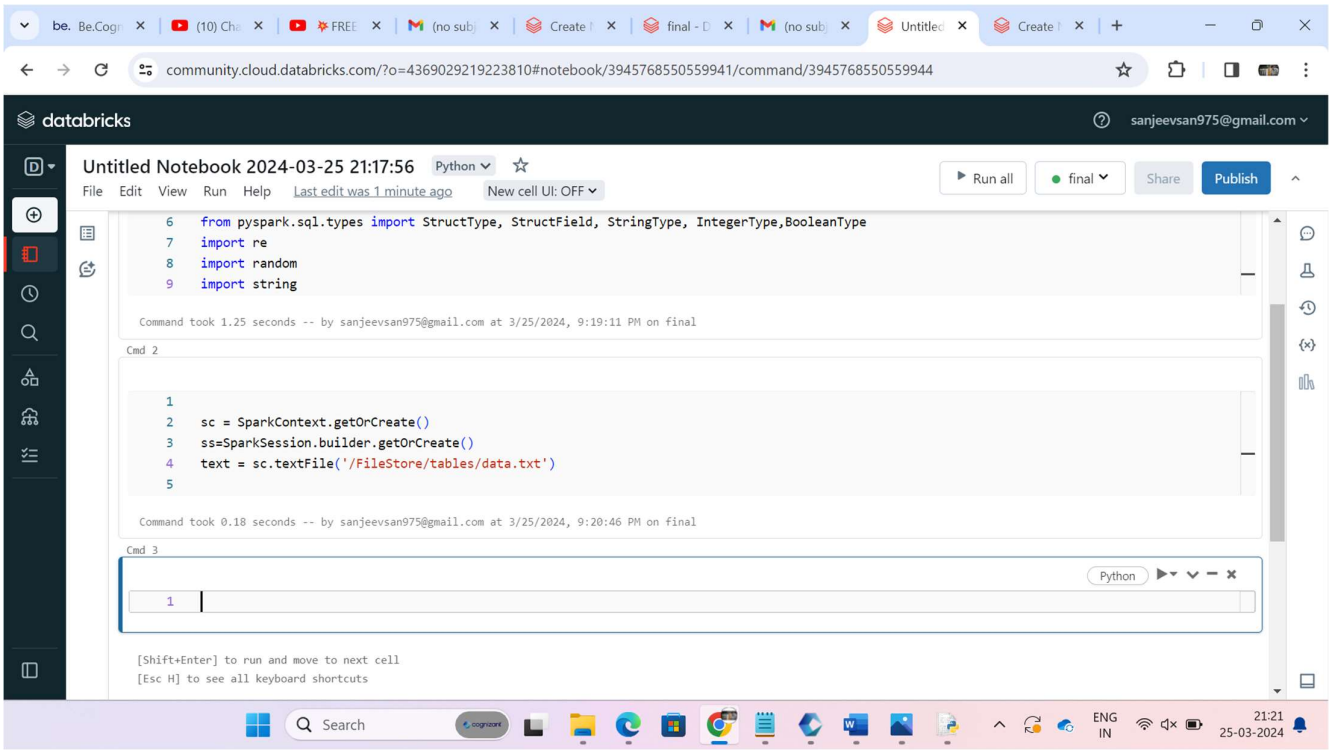
Spark Context and Session Creation:

It creates a Spark Context (`sc`) and Spark Session (`ss`) to interact with Spark functionalities.



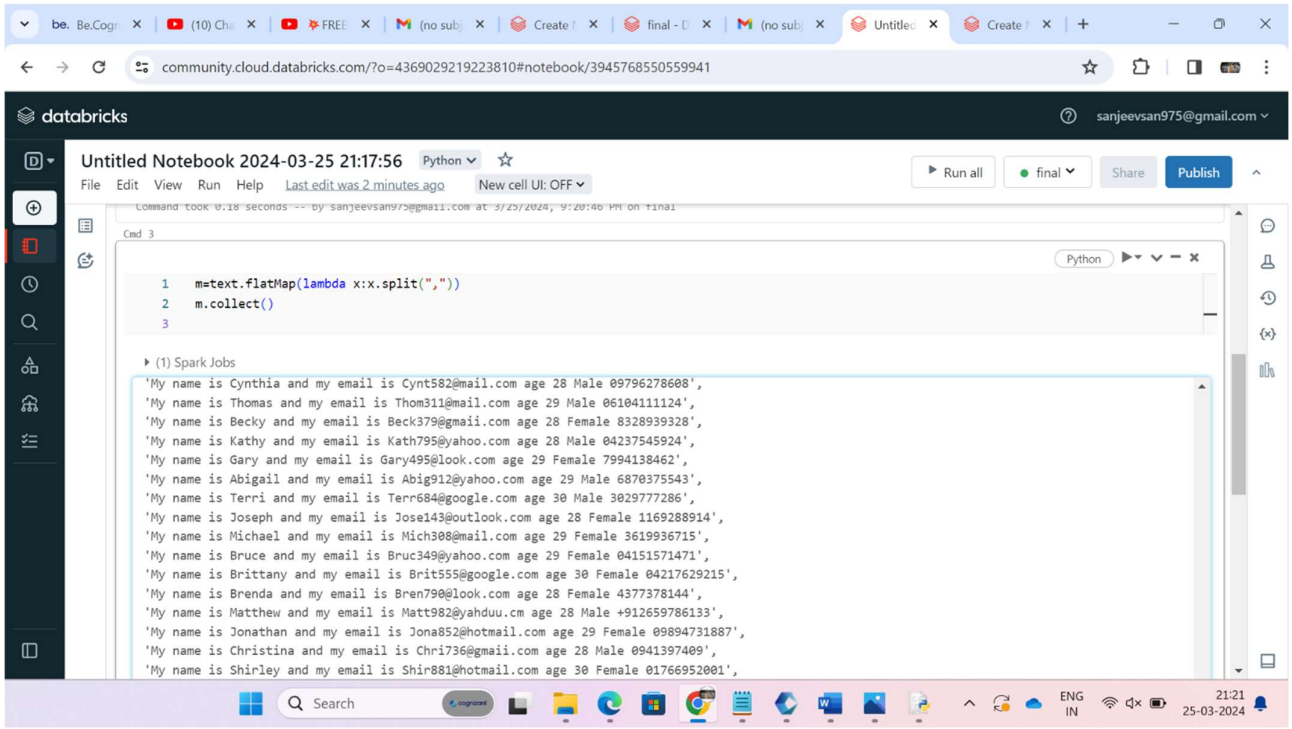
Reading Text File:

The code reads the content of a text file named `dupli.txt` located in the `/FileStore/tables/` directory using Spark's `textFile` function. This file likely contains comma-separated values (CSV).



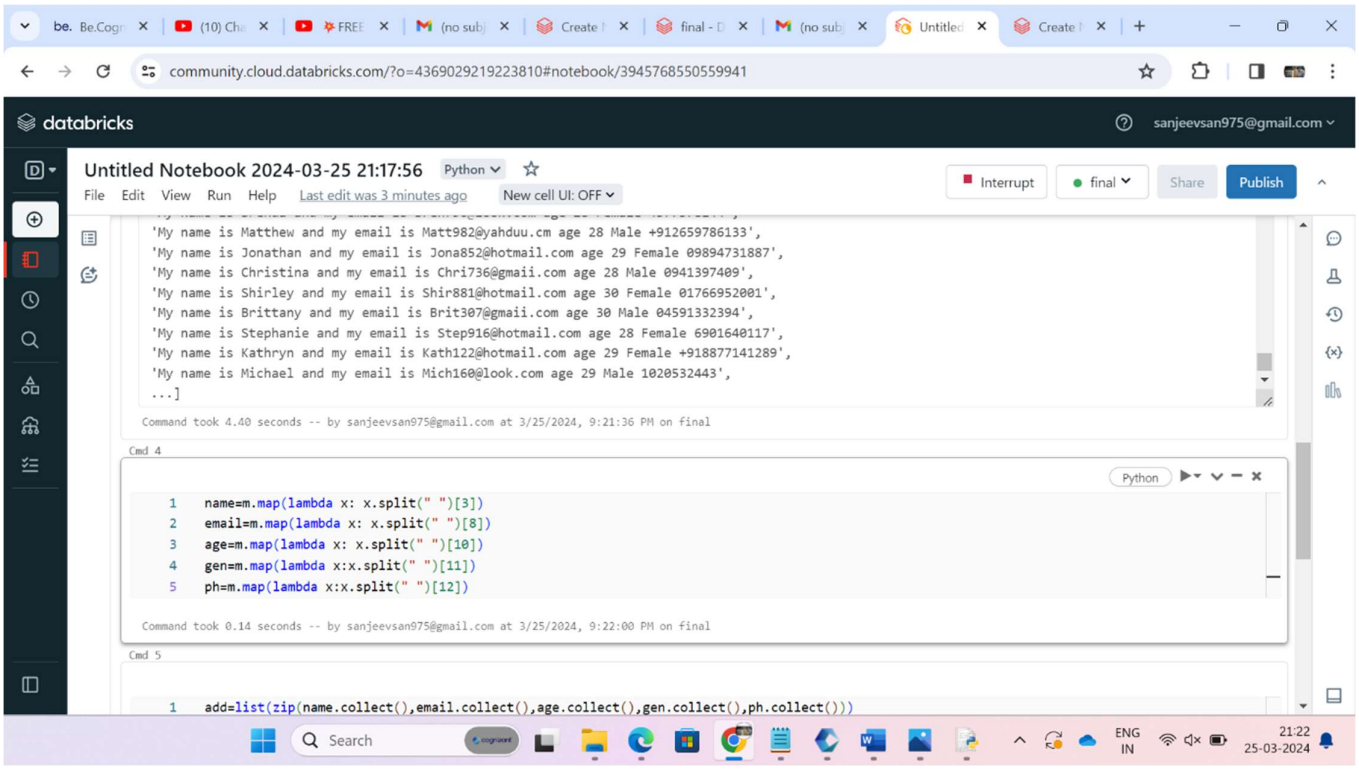
Flat Mapping and Splitting:

The `flatMap` operation is applied to split each line of the text file by comma delimiter, resulting in a flattened RDD of individual data elements.



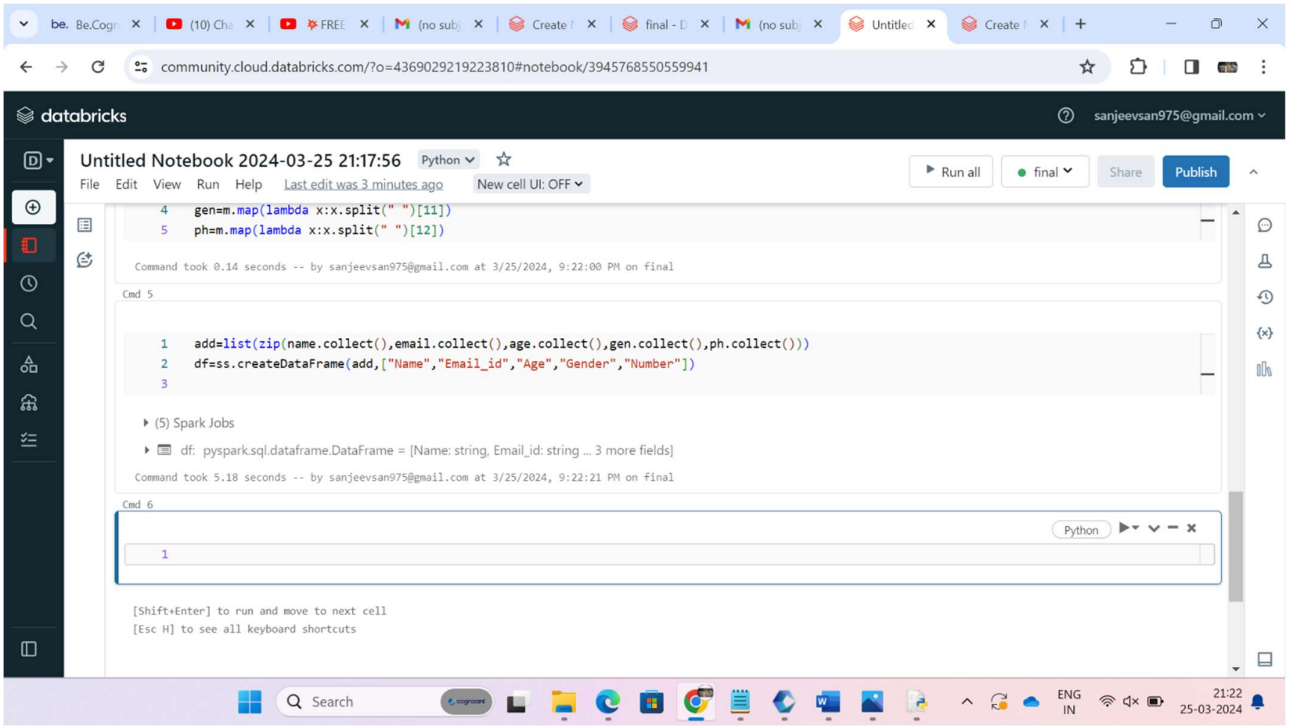
Data Extraction:

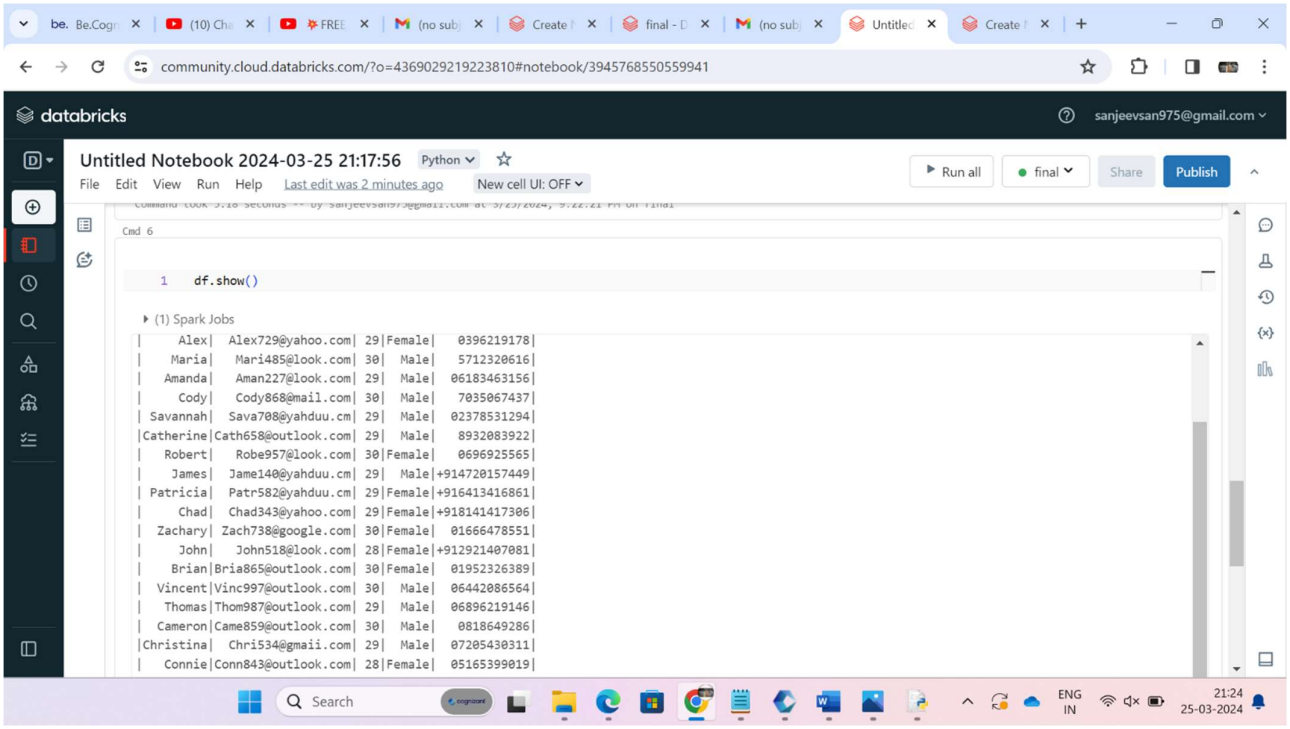
Lambda functions are used to extract specific fields like name, email, age, gender, and phone number from the flattened RDD.

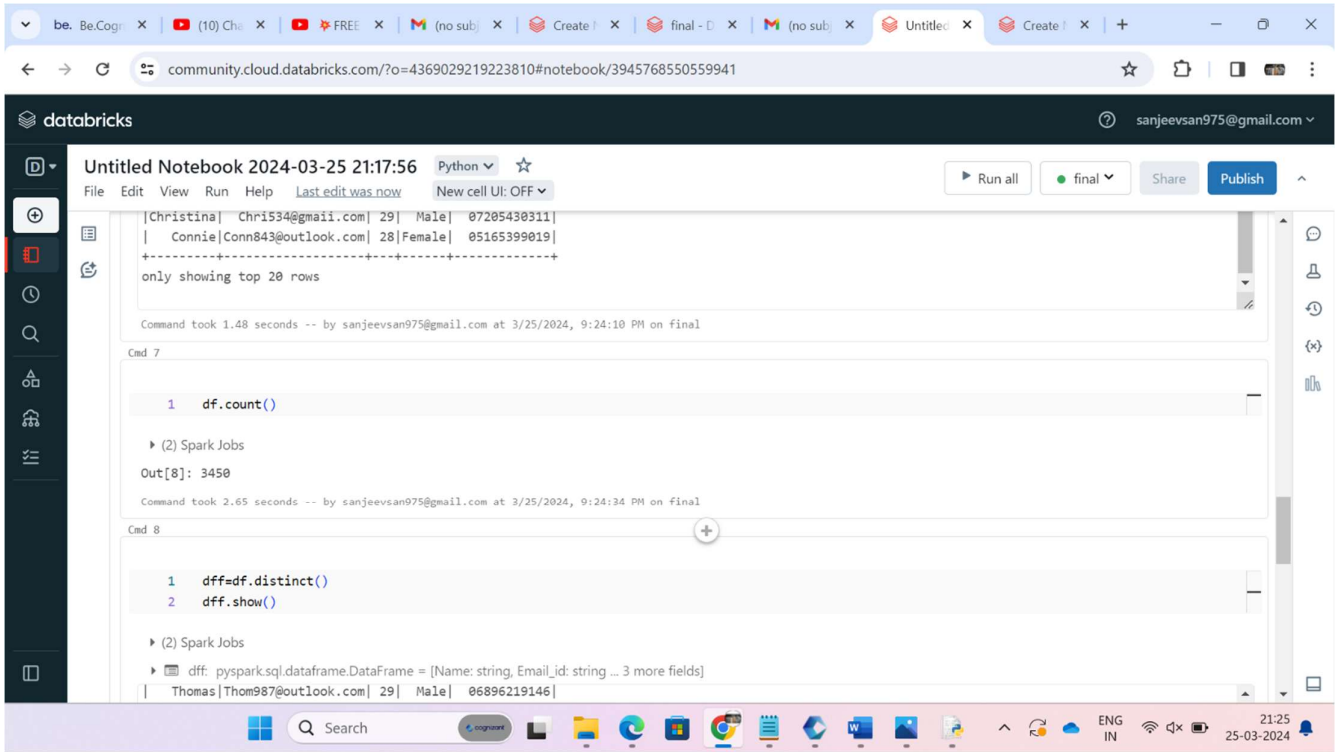


Creating DataFrame:

The extracted data is used to create a DataFrame named `df` with appropriate column names.

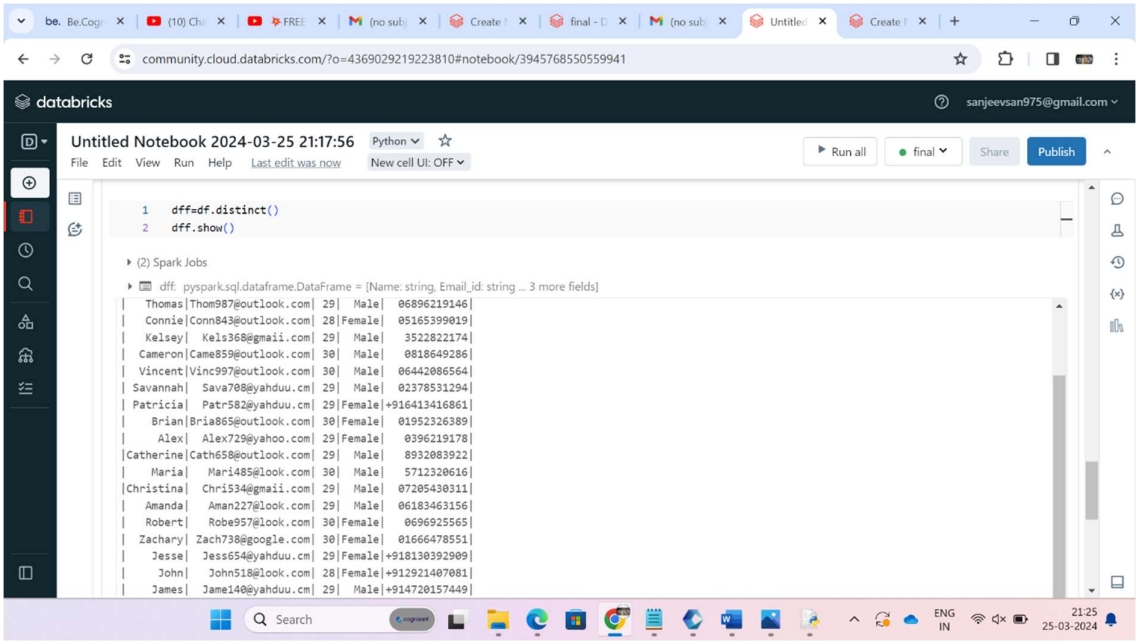


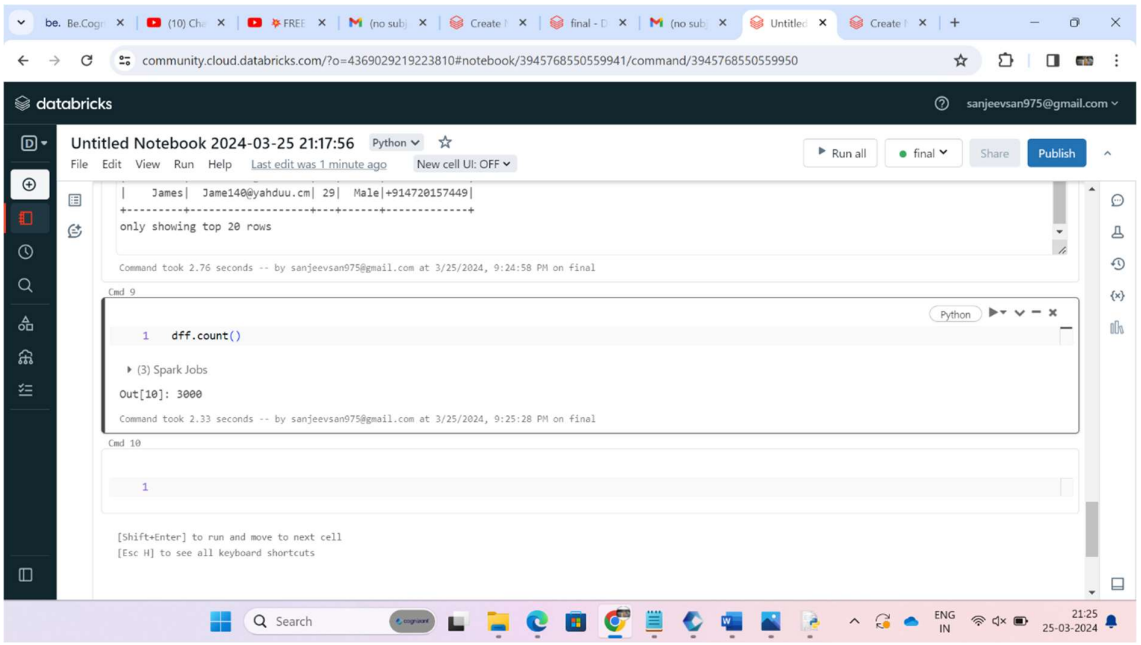

community.cloud.databricks.com/?o=4369029219223810#notebook/3945768550559941
databricks
sanjeevsan975@gmail.com
Untitled Notebook 2024-03-25 21:17:56
Python
File Edit View Run Help Last edit was 2 minutes ago
New cell UI: OFF
Run all
final
Share
Publish
Cmd 6
1 df.show()
(1) Spark Jobs
Alex	Alex729@yahoo.com	29	Female	0396219178
Maria	Mari485@look.com	30	Male	5712320616
Amanda	Aman227@look.com	29	Male	06183463156
Cody	Cody868@mail.com	30	Male	7035067437
Savannah	Sava708@yahduu.cm	29	Male	02378531294
Catherine	Cath658@outlook.com	29	Male	8932083922
Robert	Robe957@look.com	30	Female	0696925565
James	Jame140@yahduu.cm	29	Male	+914720157449
Patricia	Patr582@yahduu.cm	29	Female	+916413416861
Chad	Chad343@yahoo.com	29	Female	+918141417306
Zachary	Zach738@google.com	30	Female	01666478551
John	John518@look.com	28	Female	+912921407081
Brian	Bria865@outlook.com	30	Female	01952326389
Vincent	Vinc997@outlook.com	30	Male	06442086564
Thomas	Thom987@outlook.com	29	Male	06896219146
Cameron	Came859@outlook.com	30	Male	0818649286
Christina	Chri534@gmaii.com	29	Male	07205430311
Connie	Conn843@outlook.com	28	Female	05165399019
Search
ENG IN
21:24
25-03-2024


community.cloud.databricks.com/?o=4369029219223810#notebook/3945768550559941
databricks
sanjeevsan975@gmail.com
Untitled Notebook 2024-03-25 21:17:56
Python
File Edit View Run Help Last edit was now
New cell UI: OFF
Run all
final
Share
Publish
|Christina| Chri534@gmaii.com| 29| Male| 07205430311|
| Connie|Conn843@outlook.com| 28|Female| 05165399019|
only showing top 20 rows
Command took 1.48 seconds -- by sanjeevsan975@gmail.com at 3/25/2024, 9:24:10 PM on final
Cmd 7
1 df.count()
(2) Spark Jobs
Out[8]: 3450
Command took 2.65 seconds -- by sanjeevsan975@gmail.com at 3/25/2024, 9:24:34 PM on final
Cmd 8
1 dff=df.distinct()
2 dff.show()
(2) Spark Jobs
dff: pyspark.sql.dataframe.DataFrame = [Name: string, Email_id: string ... 3 more fields]
| Thomas|Thom987@outlook.com| 29| Male| 06896219146|
Search
ENG IN
21:25
25-03-2024

Distinct Records:

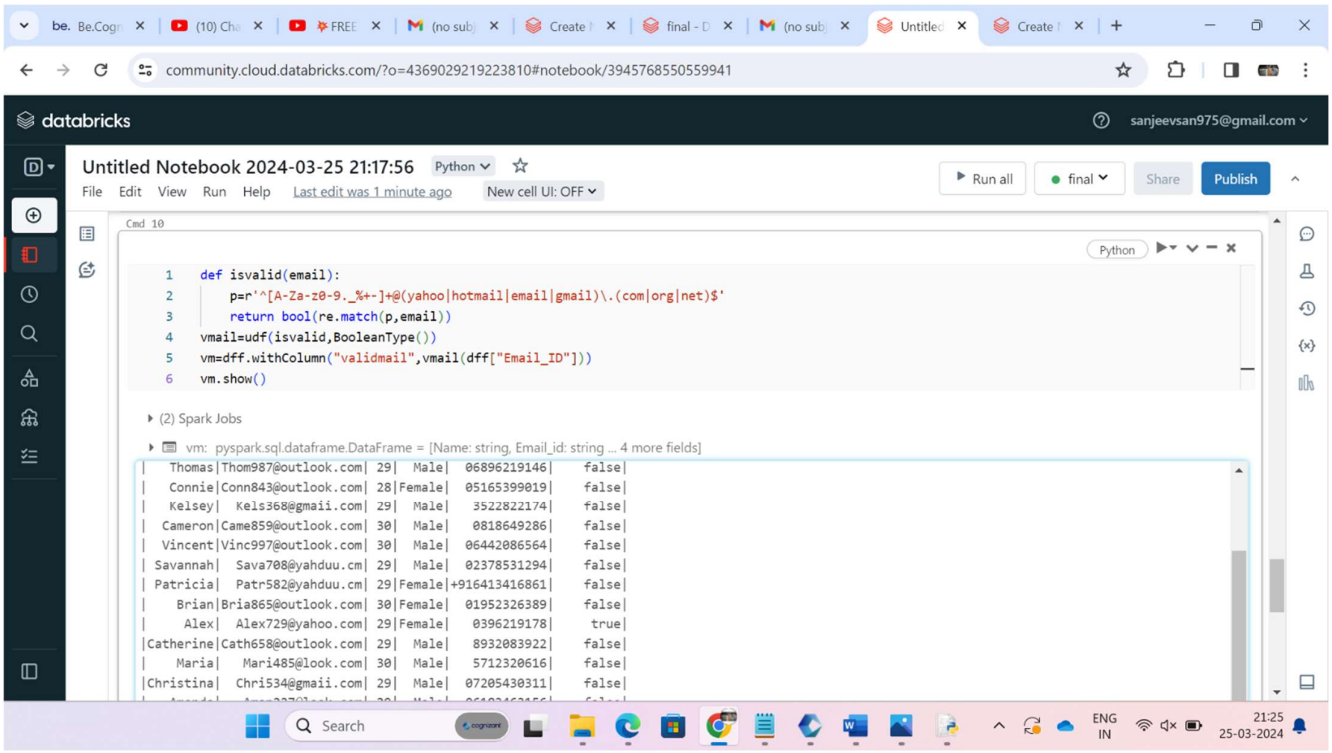
Duplicate records are removed from the DataFrame using the `distinct()` function to ensure each record is unique.





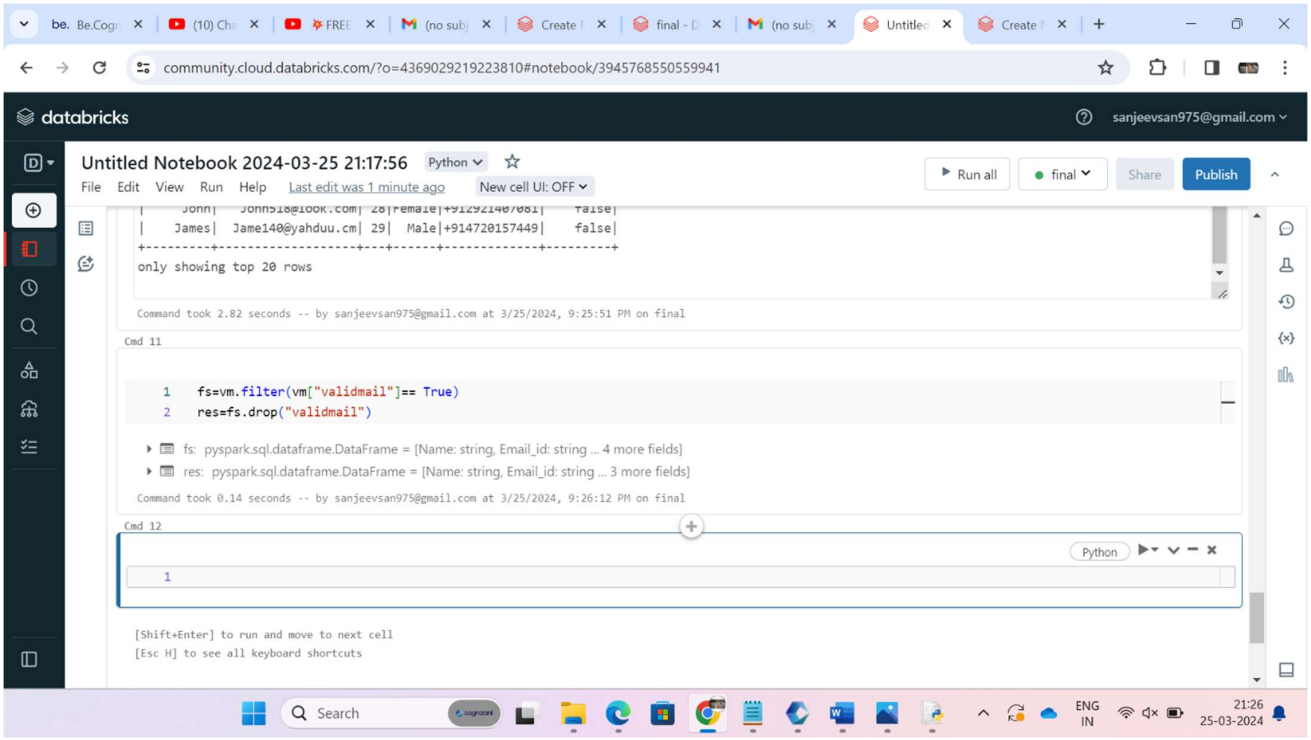
Email Validation:

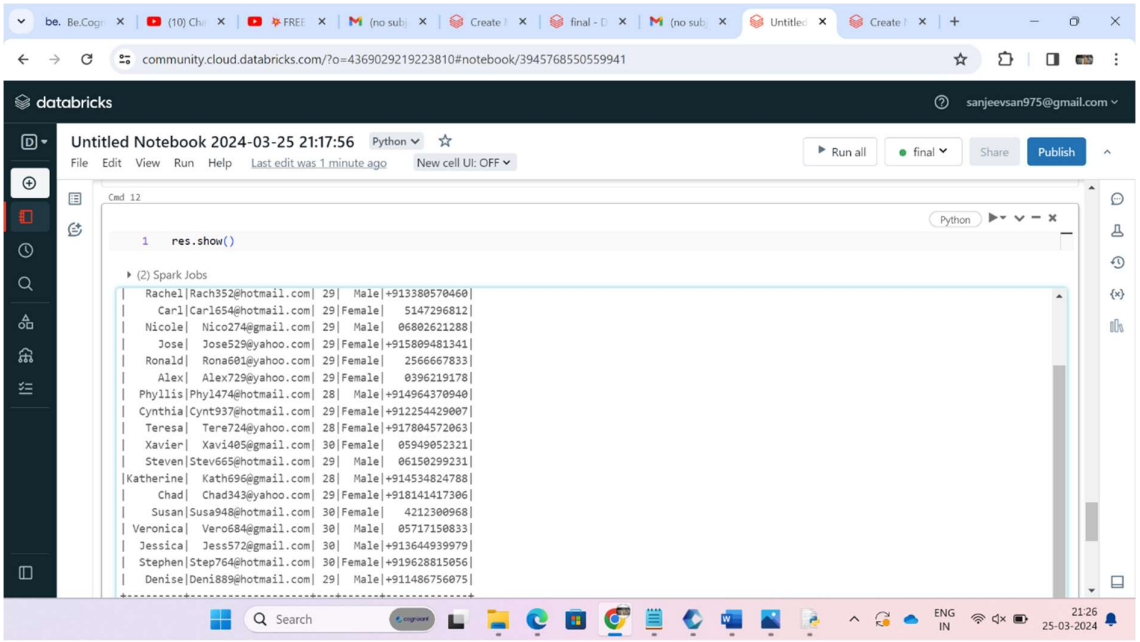
User-defined function (UDF) named `isvalid` is defined to check the validity of email addresses based on a regular expression pattern. This UDF is applied to the DataFrame to create a new column `validmail` indicating whether each email is valid or not.

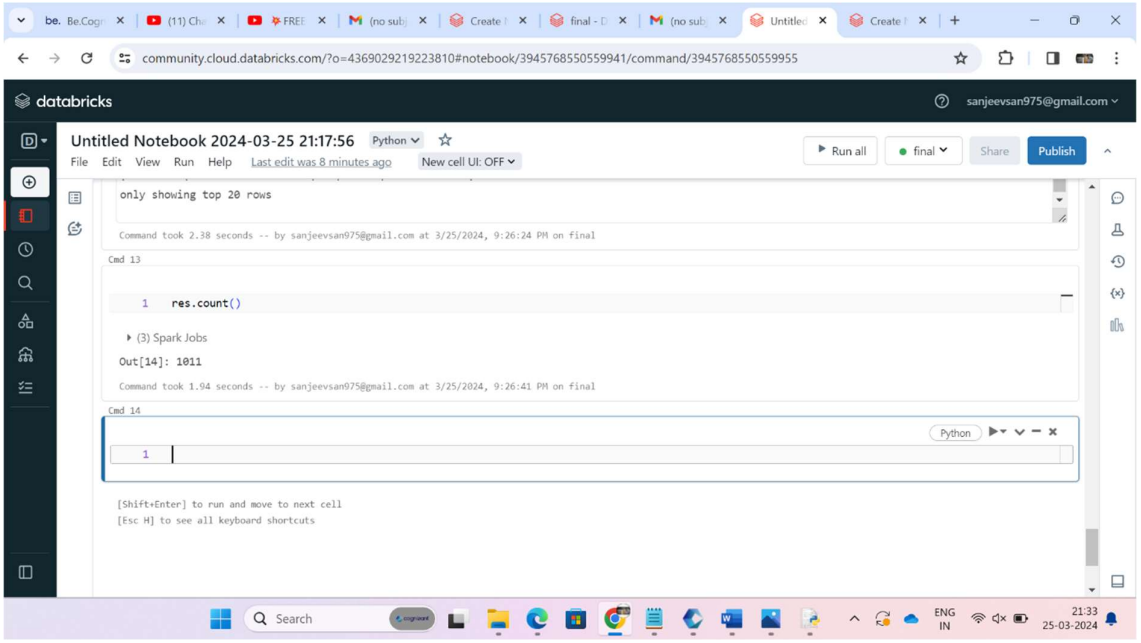


Filtering Valid Email Addresses:

Rows with valid email addresses are filtered out from the DataFrame..

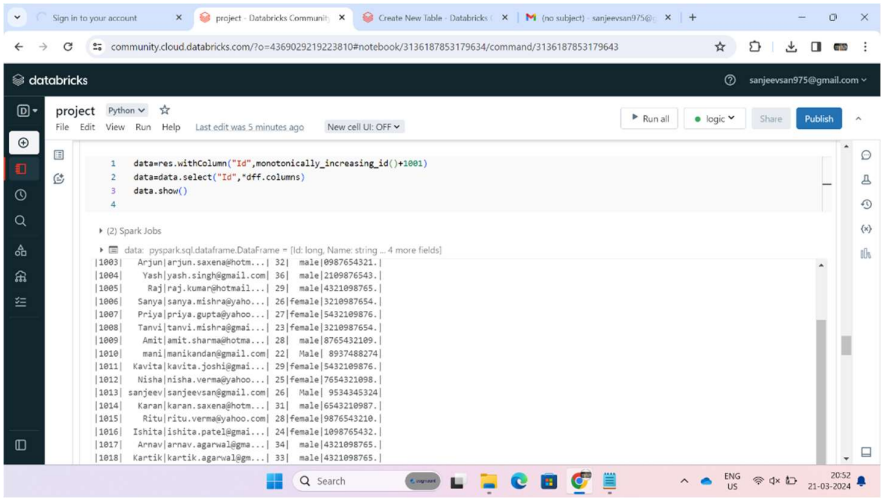



community.cloud.databricks.com/?o=4369029219223810#notebook/3945768550559941
databricks
sanjeevsan975@gmail.com
Untitled Notebook 2024-03-25 21:17:56
Python
File Edit View Run Help Last edit was 1 minute ago New cell UI: OFF
Run all
final
Share
Publish
Cmd 12
1 res.show()
(2) Spark Jobs
Rachel	Rach352@hotmail.com	29	Male	+913380570460
Carl	Carl654@hotmail.com	29	Female	5147296812
Nicole	Nico274@gmail.com	29	Male	06802621288
Jose	Jose529@yahoo.com	29	Female	+915809481341
Ronald	Rona601@yahoo.com	29	Female	2566667833
Alex	Alex729@yahoo.com	29	Female	0396219178
Phyllis	Phyl474@hotmail.com	28	Male	+914964370940
Cynthia	Cynt937@hotmail.com	29	Female	+912254429007
Teresa	Tere724@yahoo.com	28	Female	+917804572063
Xavier	Xavi405@gmail.com	30	Female	05949052321
Steven	Stev665@hotmail.com	29	Male	06150299231
Katherine	Kath696@gmail.com	28	Male	+914534824788
Chad	Chad343@yahoo.com	29	Female	+918141417306
Susan	Susa948@hotmail.com	30	Female	4212300968
Veronica	Vero684@gmail.com	30	Male	05717150833
Jessica	Jess572@gmail.com	30	Male	+913644939979
Stephen	Step764@hotmail.com	30	Female	+919628815056
Denise	Deni889@hotmail.com	29	Male	+911486756075
Search
ENG IN
21:26 25-03-2024


community.cloud.databricks.com/?o=4369029219223810#notebook/3945768550559941/command/3945768550559955
databricks
sanjeevsan975@gmail.com
Untitled Notebook 2024-03-25 21:17:56
Python
File Edit View Run Help Last edit was 8 minutes ago New cell UI: OFF
Run all
final
Share
Publish
only showing top 20 rows
Command took 2.38 seconds -- by sanjeevsan975@gmail.com at 3/25/2024, 9:26:24 PM on final
Cmd 13
1 res.count()
(3) Spark Jobs
Out[14]: 1011
Command took 1.94 seconds -- by sanjeevsan975@gmail.com at 3/25/2024, 9:26:41 PM on final
Cmd 14
1
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG IN
21:33 25-03-2024

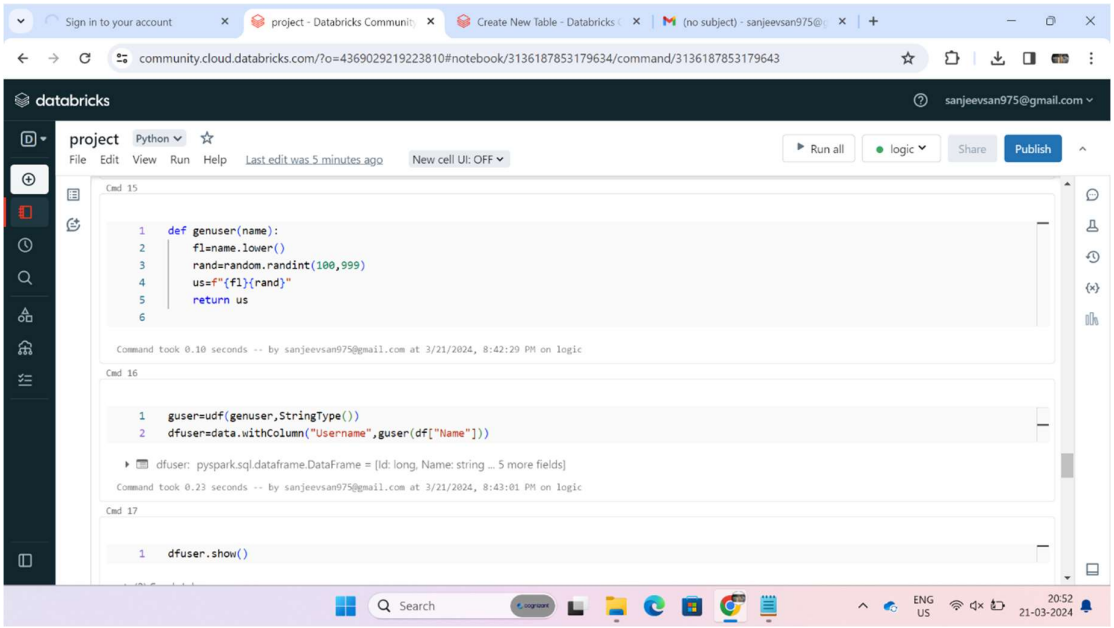
Adding Unique IDs:

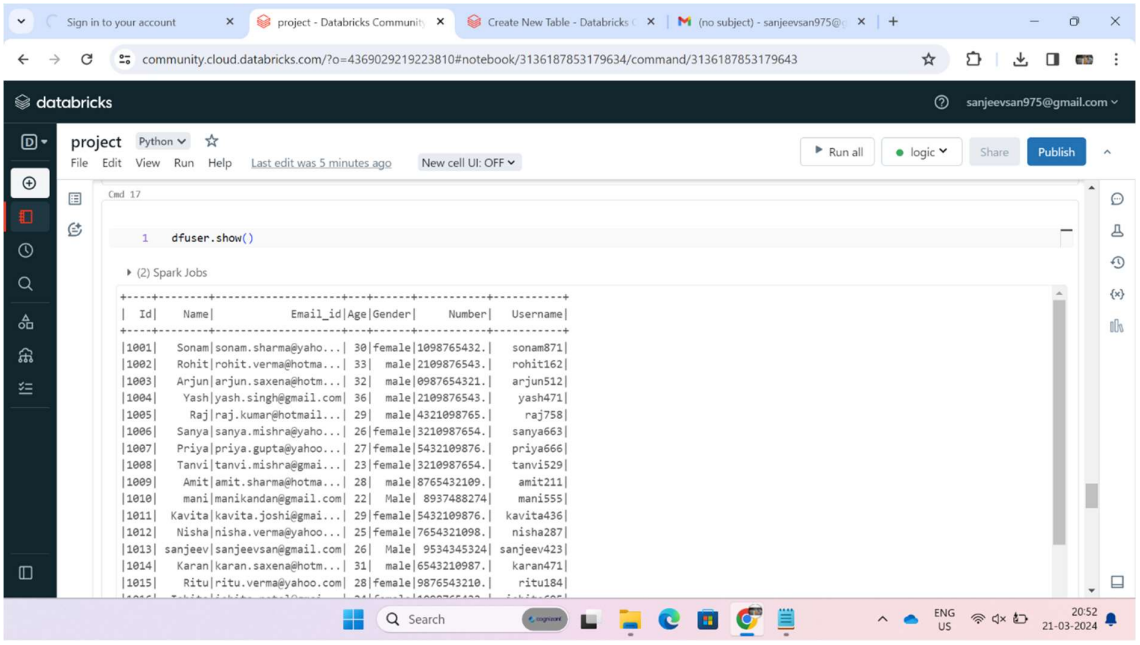
Unique IDs are added to each row using the `monotonically_increasing_id()` function, ensuring each row has a distinct identifier.



Generating Usernames:

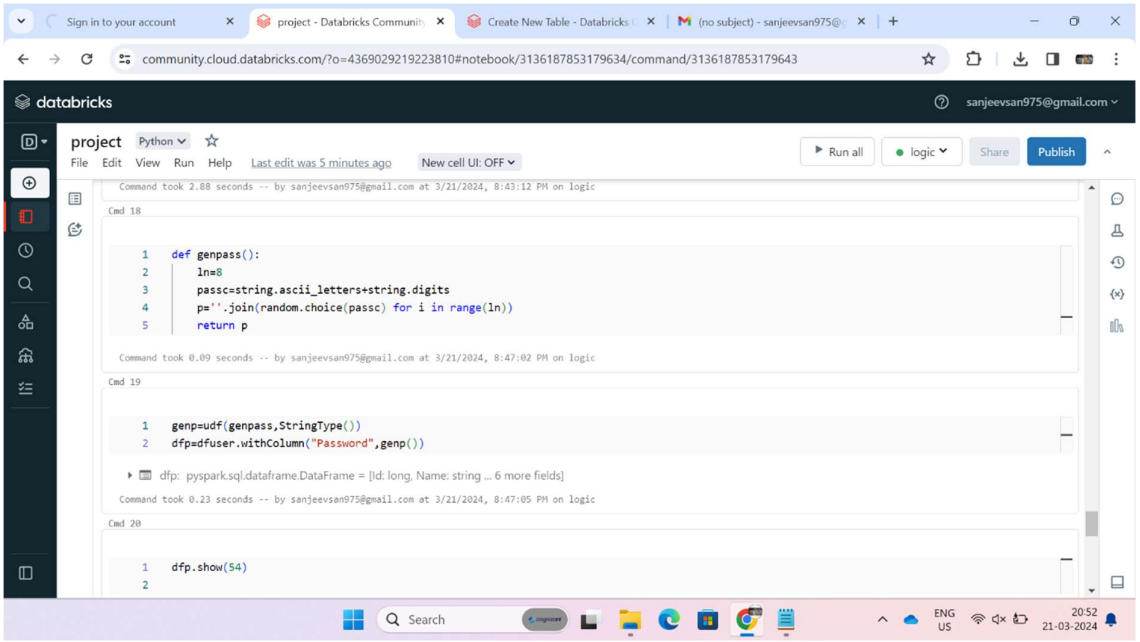
Another UDF named `genuser` is defined to generate usernames based on the lowercased first name and a random integer. This UDF is applied to create a new column `Username`.

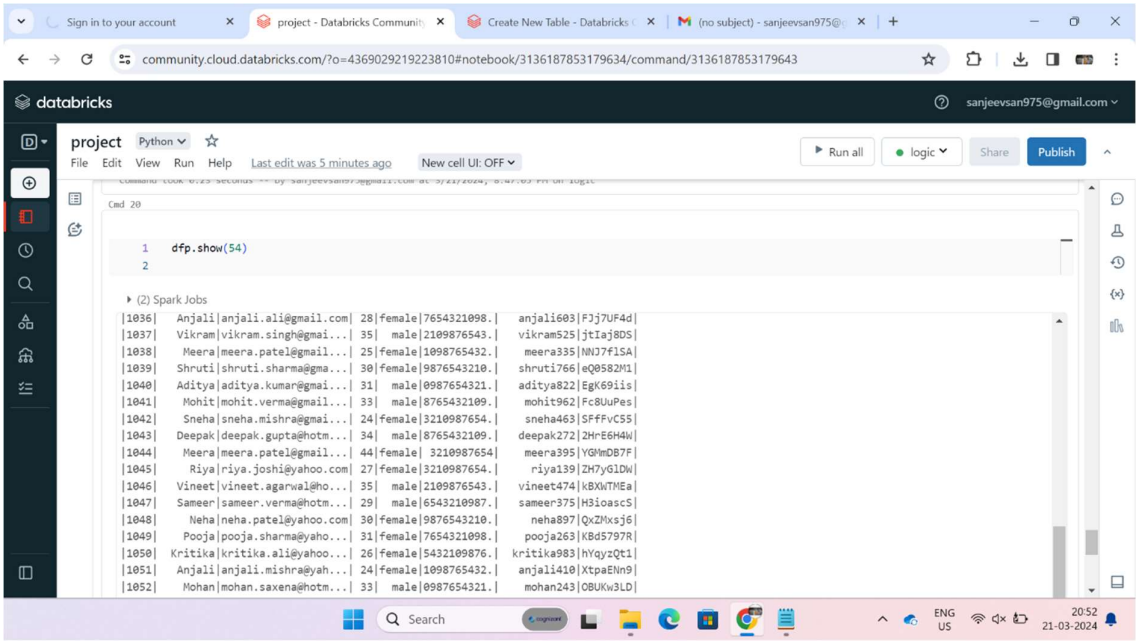




Generating Passwords:

Similarly, a UDF named `genpass` is defined to generate random passwords consisting of letters and digits. This UDF is applied to create a new column `Password`





Loading to MySQL Cloud:

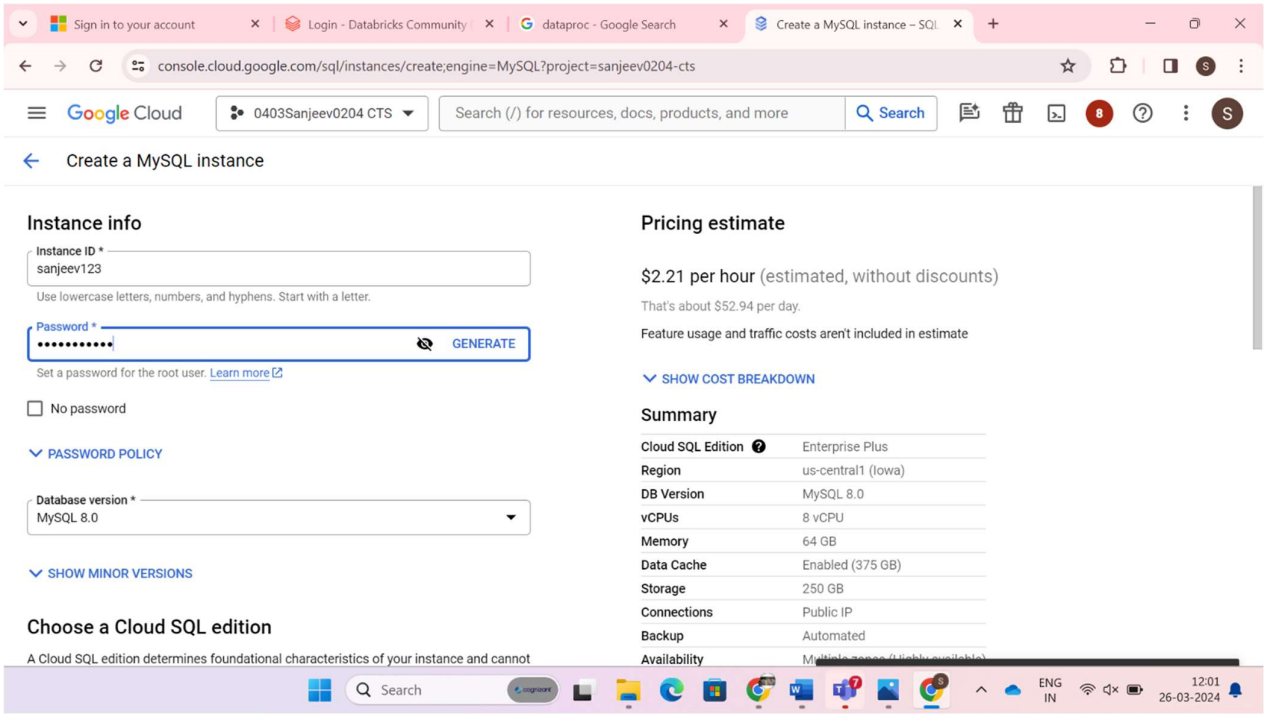
Finally, the processed DataFrame (`dfp`) is written to a MySQL database table specified by the `table` parameter using JDBC connection parameters such as driver, URL, username, and password. The data is appended to the table in the database.

For that we are creating a google cloud SQL for loading a data.

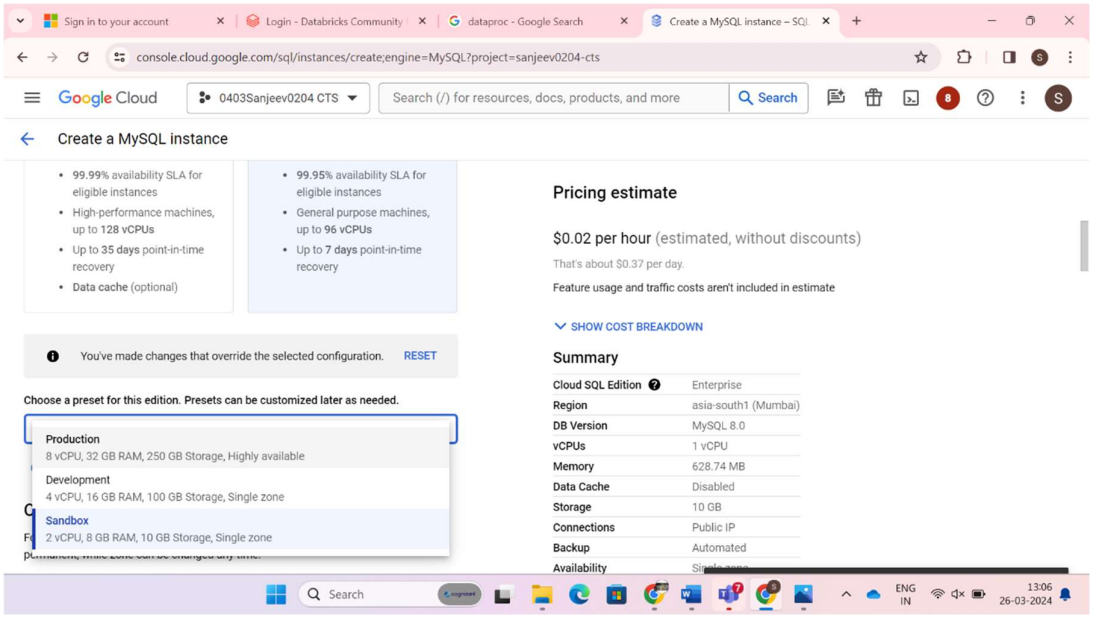
MySQL in the cloud offers scalability, managed services, high availability, global accessibility, and cost efficiency. It easily scales to accommodate growing data needs, freeing up time with managed services for backups, maintenance, and updates. High availability features ensure uptime and data durability, while global accessibility fosters collaboration. Cost efficiency stems from a pay-as-you-go model.

Conversely, local MySQL provides performance advantages, data control, predictable costs, customization, and offline access. It offers faster query execution, control over infrastructure, predictable costs, customization options, and ensures uninterrupted access in environments with limited internet connectivity. Organizations often choose based on needs, budget, and regulatory requirements.

1) Create instance in Mysql cloud:

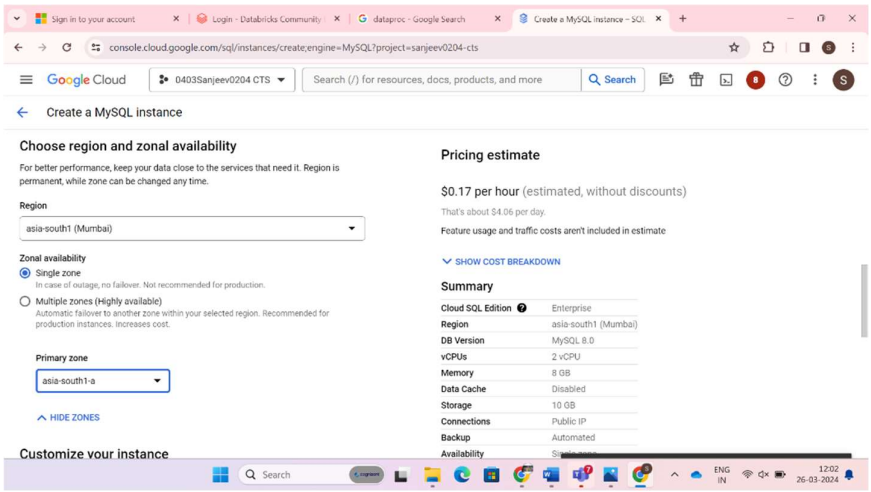


2) Compare to other production sandbox is less in features and prize.



Region and zone:

A region is a specific geographical location where you can run your resources. Each region is subdivided into several zones. For example, the us-central1 region in the central United States has zones us-central1-a , us-central1-b , us-central1-c , and us-central1-f .



3) Shared Core (vCPU) Instances and Dedicated Core (vCPU) Instances:

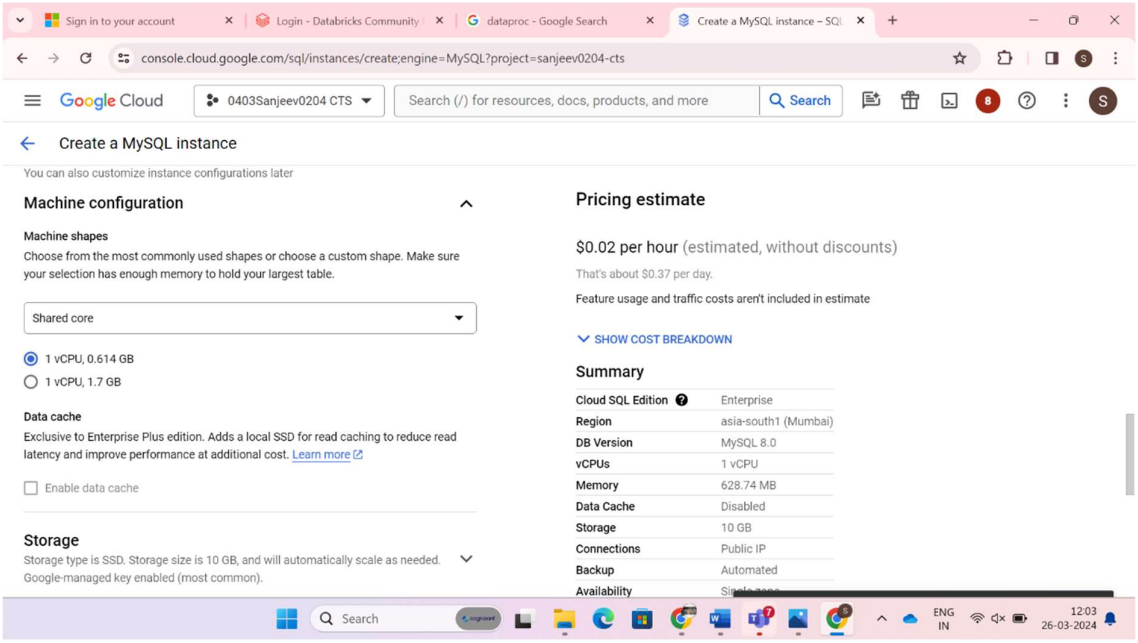
Shared Core (vCPU) Instances:

- Shared-core machine types provide **one virtual CPU** that runs for a portion of the time on a single hardware hyper-thread of the host CPU.
- These instances are **cost-effective** for running small, non-resource-intensive applications.

Dedicated Core (vCPU) Instances:

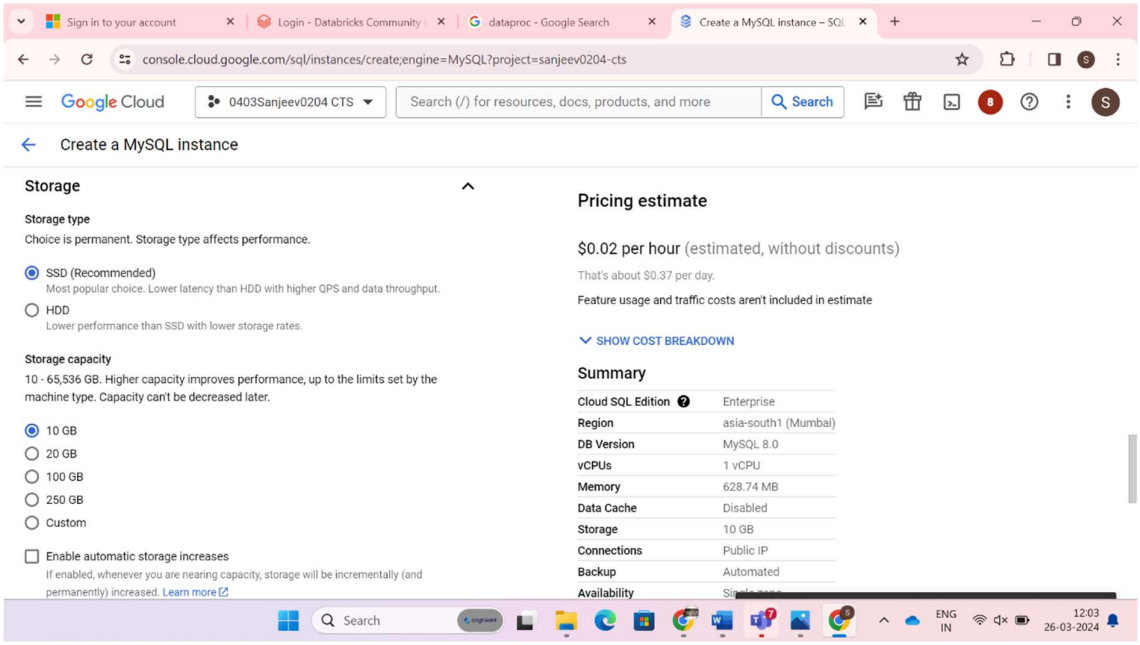
- In contrast, dedicated core instances allocate **exclusive CPU resources** to your virtual machine.
- Each core is reserved for your instance, ensuring consistent performance and security.

So, we using shared core for cost-effective



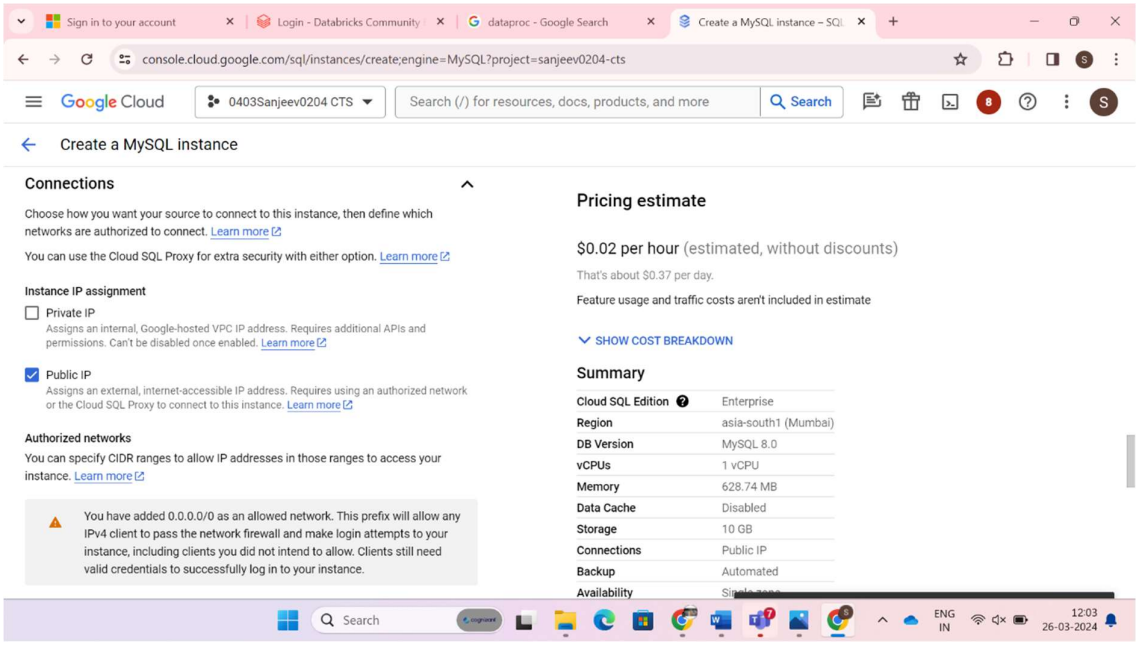
4) Storage SSD or HDD:

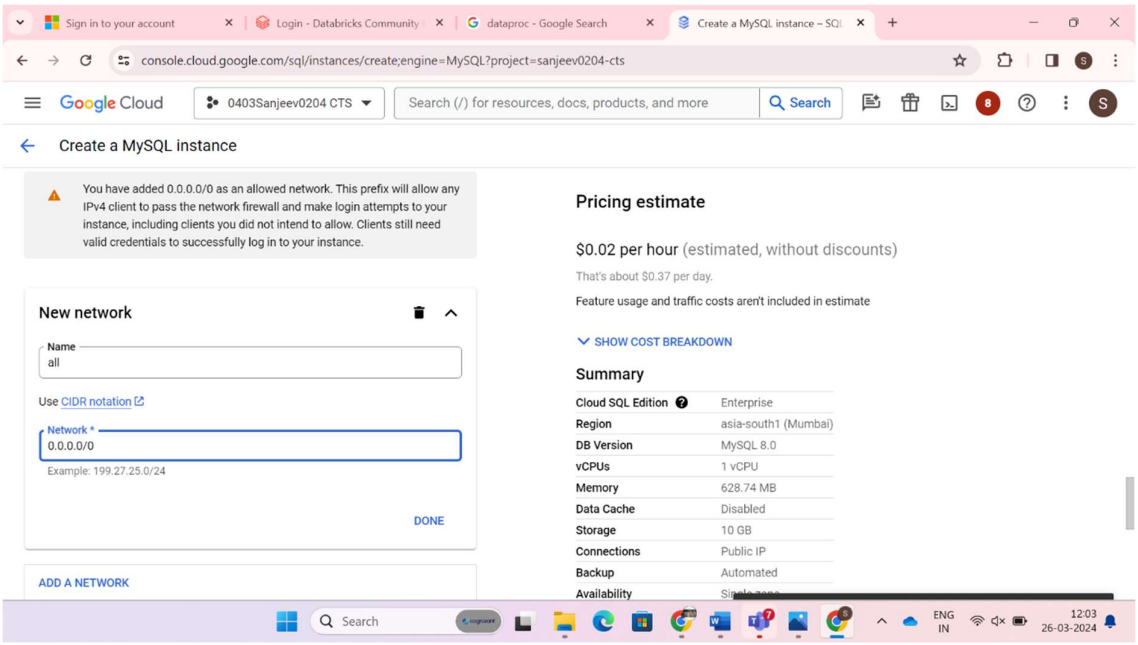
Solid state drives (SSDs) typically use flash-based memory to store data and thus have no moving parts. They have faster read/write speeds than HDDs, lower access times (less latency), and a higher cost per gigabyte of storage. So, we choose SSD.



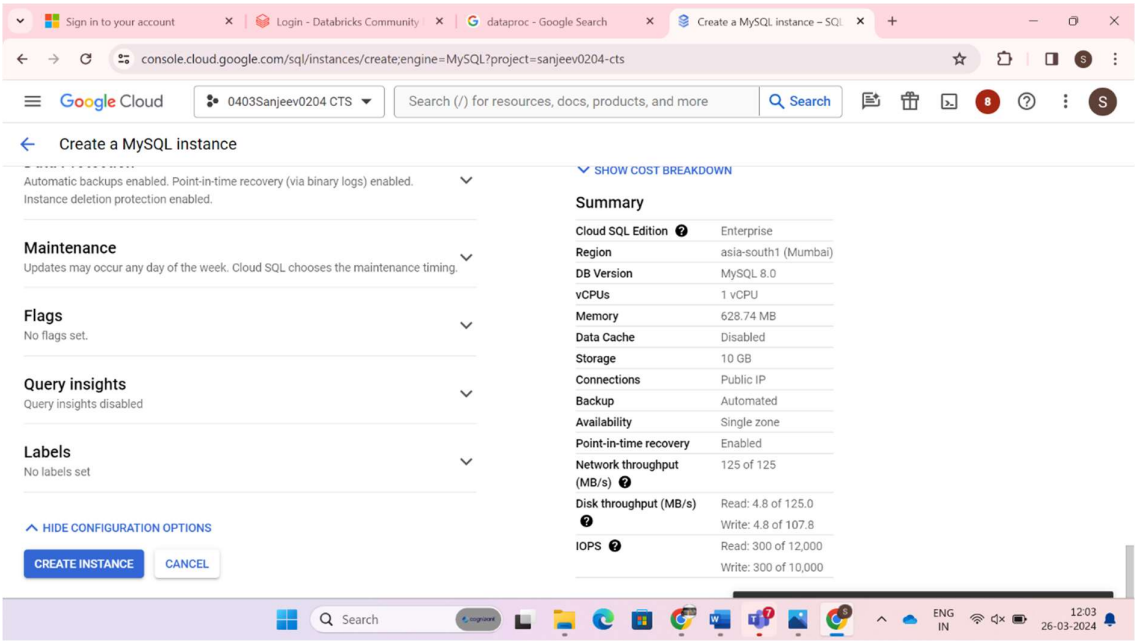
5) Network connection :

A public IP address is a unique IP address assigned to your network router by your internet service provider and can be accessed directly over the internet. A private IP address is a unique address that your network router assigns to your device. It is used within a private network to connect securely to other devices.


Sign in to your account
Login - Databricks Community
dataproc - Google Search
Create a MySQL instance – SQL
console.cloud.google.com/sql/instances/create;engine=MySQL?project=sanjeev0204-cts
Google Cloud
0403Sanjeev0204 CTS
Search (/) for resources, docs, products, and more
Search
Create a MySQL instance
Connections
Choose how you want your source to connect to this instance, then define which networks are authorized to connect. Learn more
You can use the Cloud SQL Proxy for extra security with either option. Learn more
Instance IP assignment
Private IP
Assigns an internal, Google-hosted VPC IP address. Requires additional APIs and permissions. Can't be disabled once enabled. Learn more
Public IP
Assigns an external, internet-accessible IP address. Requires using an authorized network or the Cloud SQL Proxy to connect to this instance. Learn more
Authorized networks
You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. Learn more
You have added 0.0.0.0/0 as an allowed network. This prefix will allow any IPv4 client to pass the network firewall and make login attempts to your instance, including clients you did not intend to allow. Clients still need valid credentials to successfully log in to your instance.
Pricing estimate
$0.02 per hour (estimated, without discounts)
That's about $0.37 per day.
Feature usage and traffic costs aren't included in estimate
SHOW COST BREAKDOWN
Summary
Cloud SQL Edition Enterprise
Region asia-south1 (Mumbai)
DB Version MySQL 8.0
vCPUs 1 vCPU
Memory 628.74 MB
Data Cache Disabled
Storage 10 GB
Connections Public IP
Backup Automated
Availability
Search
ENG IN
12:03 26-03-2024

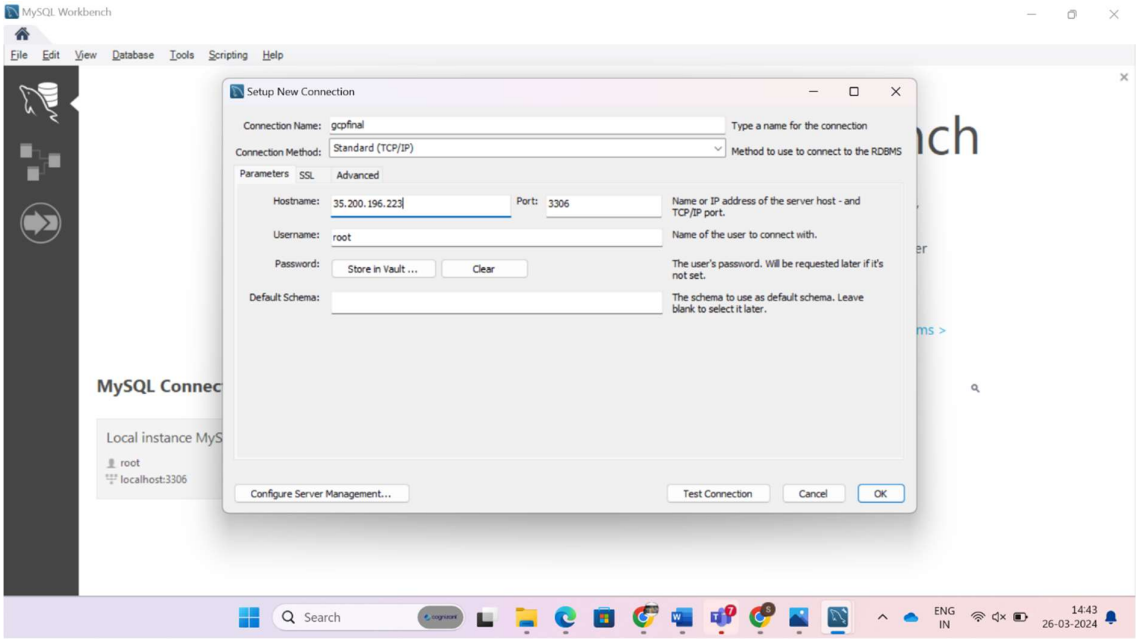

Sign in to your account
Login - Databricks Community
dataproc - Google Search
Create a MySQL instance – SQL
console.cloud.google.com/sql/instances/create;engine=MySQL?project=sanjeev0204-cts
Google Cloud
0403Sanjeev0204 CTS
Search (/) for resources, docs, products, and more
Search
Create a MySQL instance
You have added 0.0.0.0/0 as an allowed network. This prefix will allow any IPv4 client to pass the network firewall and make login attempts to your instance, including clients you did not intend to allow. Clients still need valid credentials to successfully log in to your instance.
New network
Name
all
Use CIDR notation
Network *
0.0.0.0/0
Example: 199.27.25.0/24
DONE
ADD A NETWORK
Pricing estimate
$0.02 per hour (estimated, without discounts)
That's about $0.37 per day.
Feature usage and traffic costs aren't included in estimate
SHOW COST BREAKDOWN
Summary
Cloud SQL Edition Enterprise
Region asia-south1 (Mumbai)
DB Version MySQL 8.0
vCPUs 1 vCPU
Memory 628.74 MB
Data Cache Disabled
Storage 10 GB
Connections Public IP
Backup Automated
Availability
Search
ENG IN
12:03 26-03-2024

6) Then create an instance:

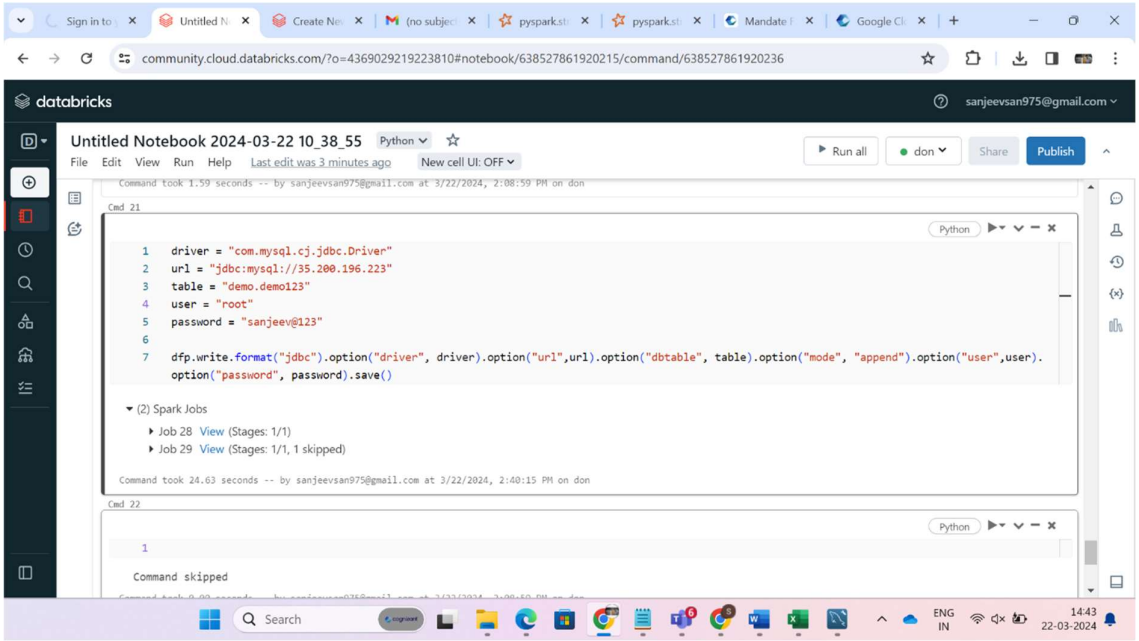


Now MySql instance is create,

1) Open mysql work bench.
2) And open setup new connections and enter a connection name. And Hostname in that we need give mysql instance pubile IP address.
3) Enter a password what we are given in mysql instance.
4) And check for the connection After the successful connection.
5) Then open the mysql work bench and create a database as “demo”.

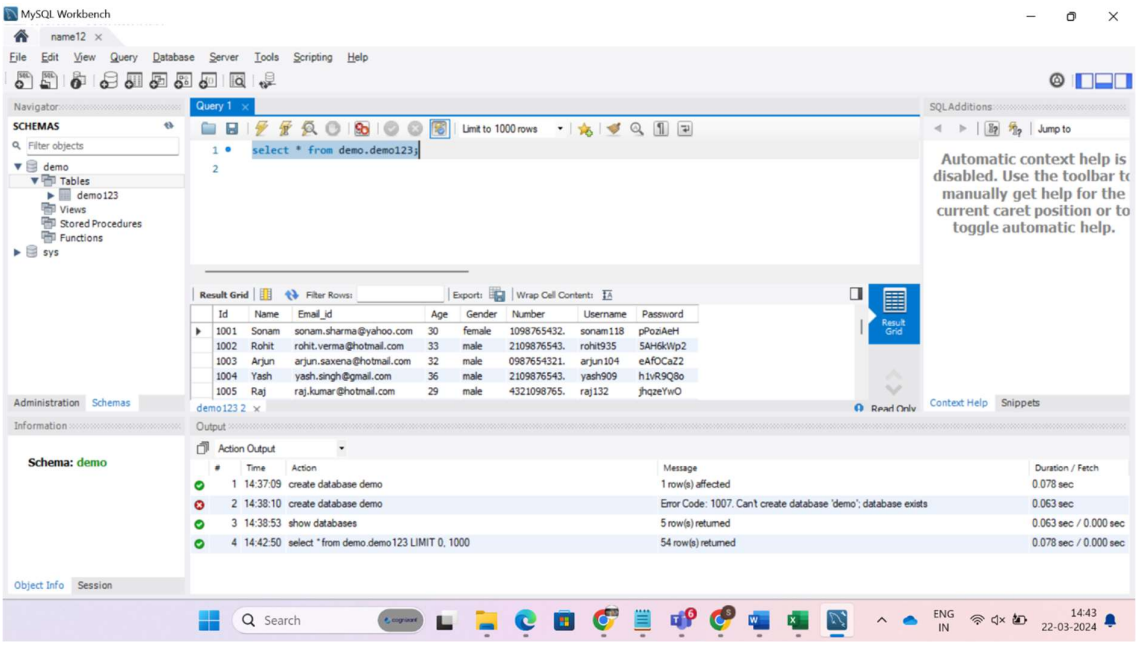


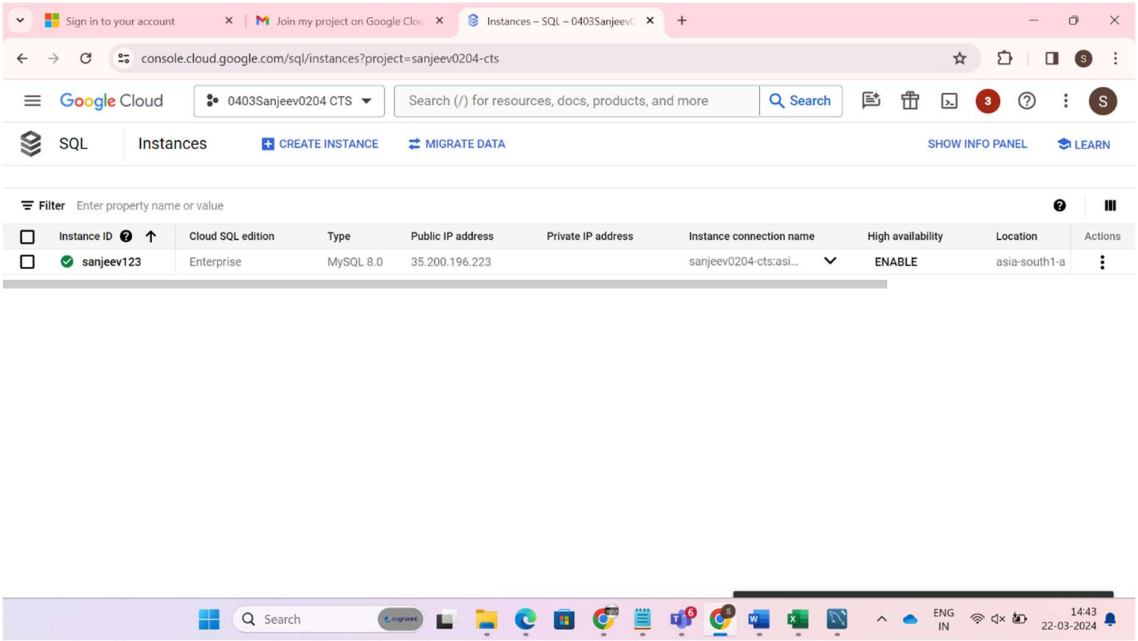
- We have a code to connect Mysql cloud to Databricks.
- Now we need to give IP address in URL.
- Enter database name and table name in table row.
- Then enter the password and run the program.



Check load data in Mysql cloud:

- Open Mysql work bench and enter the command. Using select statement.
- You will the loaded data.





Chapter-3

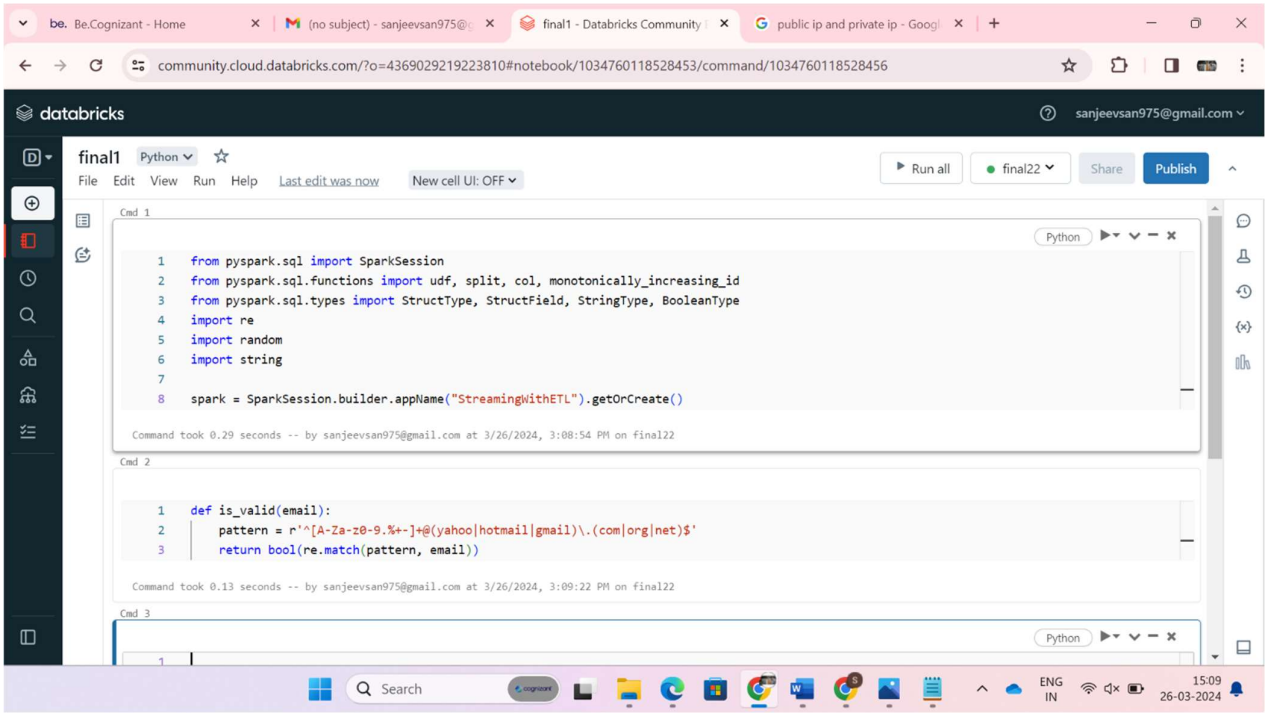
STREAMING USING ETL:

1)Imports:

It imports necessary modules and functions from PySpark, such as SparkSession, udf, split, col, monotonically_increasing_id, StructType, StructField, StringType, BooleanType, as well as standard Python modules like re (regular expressions), random, and string.

2)SparkSession:

It creates a SparkSession named "StreamingWithETL" or retrieves an existing one if available

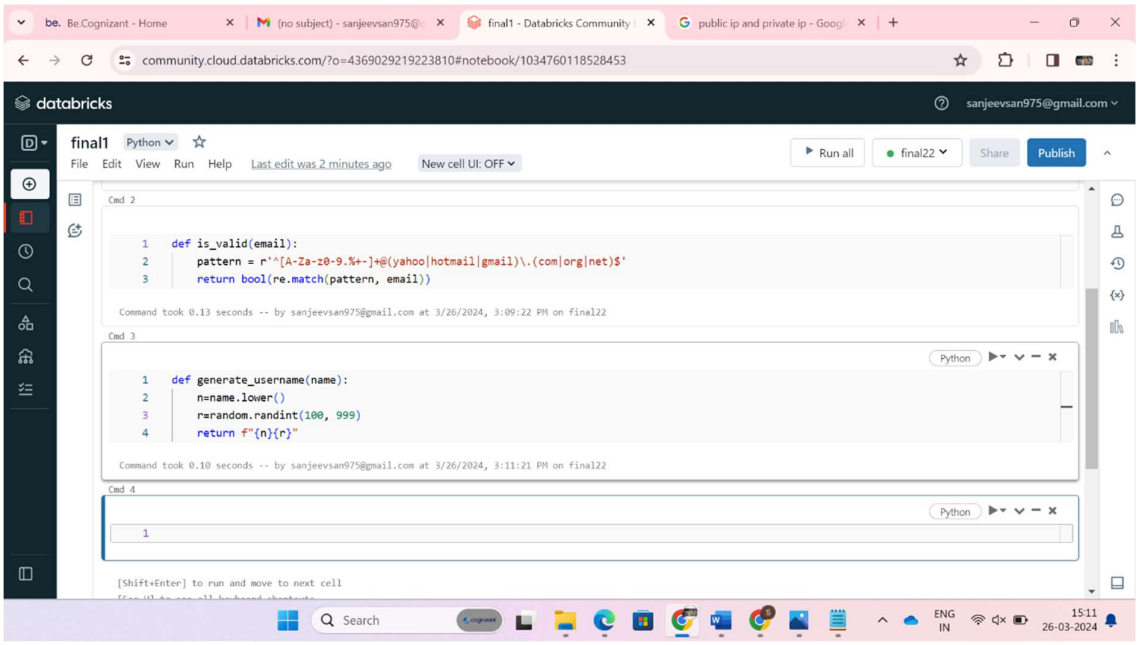


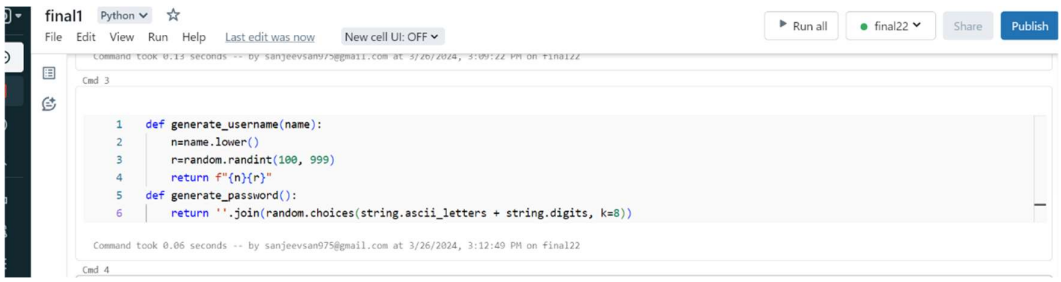
3) User-Defined Functions (UDFs):

-**`is_valid`:** A function to validate email addresses based on a regular expression pattern.

- **`generate_username`:** A function to generate a username from a given name.

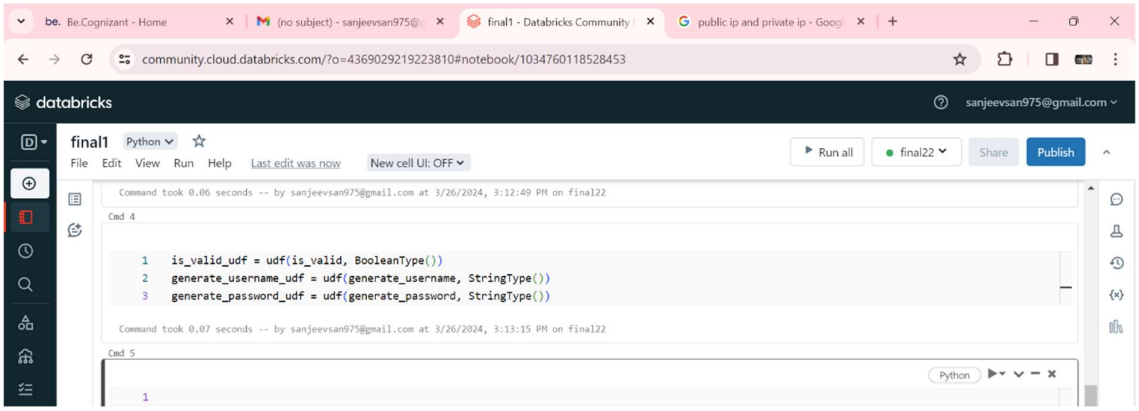
- **`generate_password`:** A function to generate a random password.





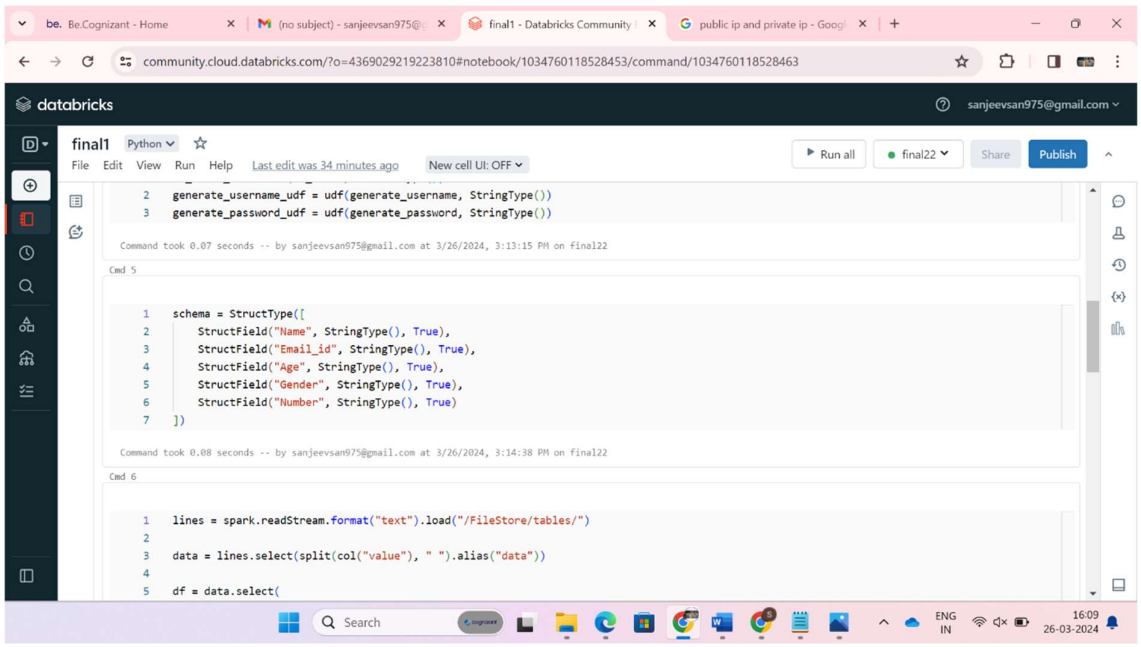
4) UDF Registration:

Registers the UDFs using the `udf` function provided by PySpark.



5) Schema Definition:

Defines the schema for the structured streaming data. The schema includes fields for "Name", "Email id", "Gender", and "Number".



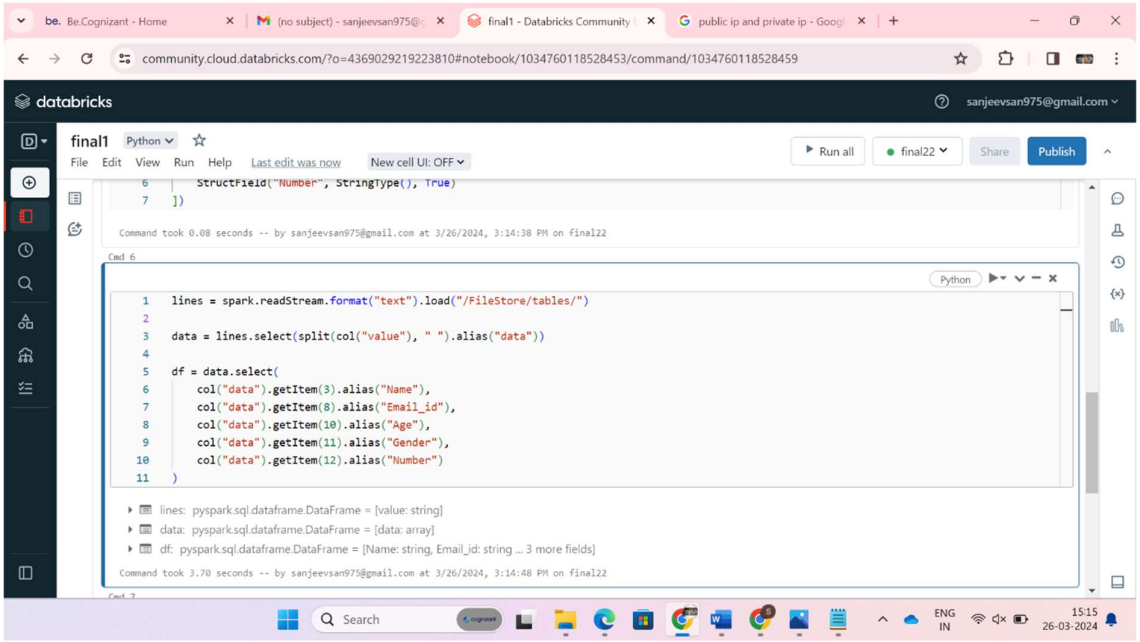
6) Reading Stream:

- Reads streaming data from a specified location ("/FileStore/tables/") as text files.

Data Transformation:

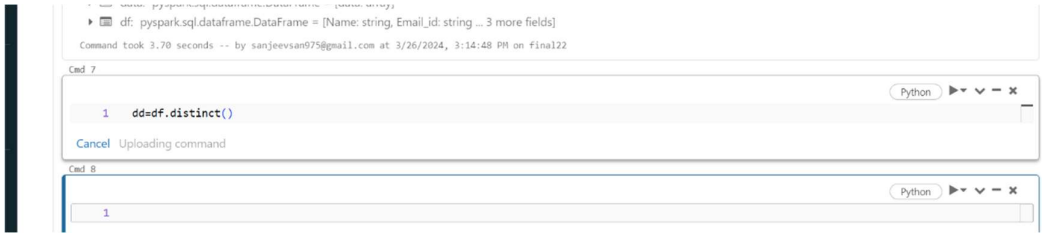
- Splits the lines into columns using the `split` function and assigns the result to a column named "data".

- Selects specific columns from the "data" column to create a DataFrame (`df`) with columns "Name", "Email_id", "Age", "Gender", and "Number".



7) Data Deduplication:

- Removes duplicate records from the DataFrame (`df`) using the `distinct` method.

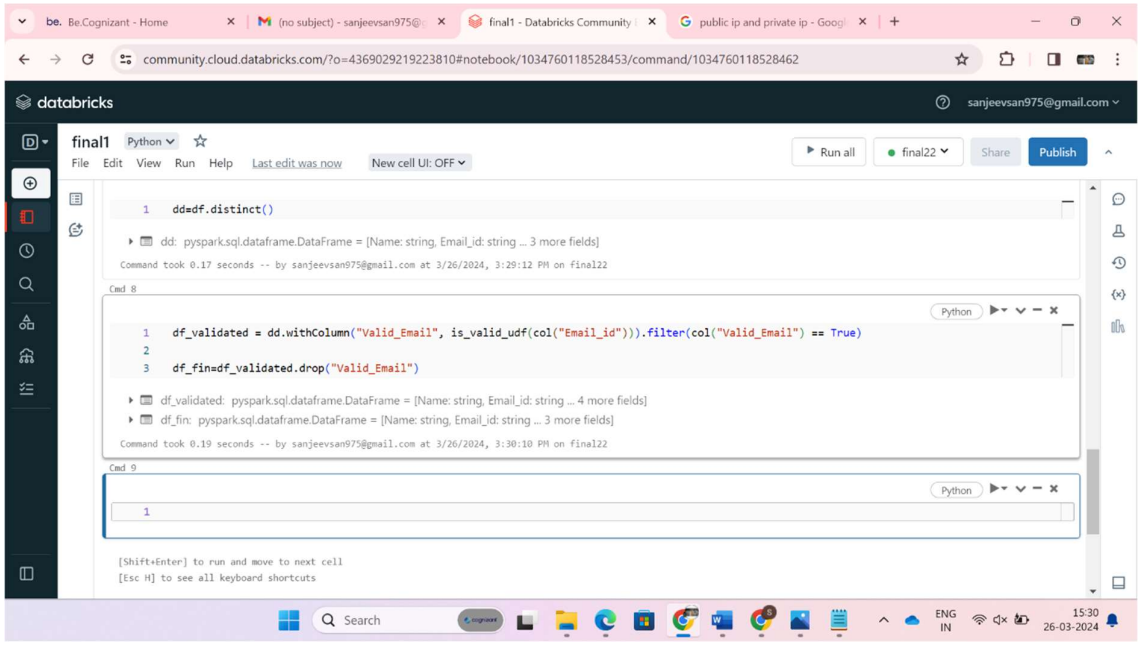


8) Data Validation:

- Adds a column "Valid Email" to indicate whether the email addresses are valid by applying the `is_valid_udf` function.

- Filters out rows where the email address is valid.

- Drops the "Valid_Email" column from the DataFrame



Additional Data Transformation:

- Adds a "Username" column by applying the `generate_username_udf` function.

- Adds a "Password" column by applying the `generate_password_udf` function.

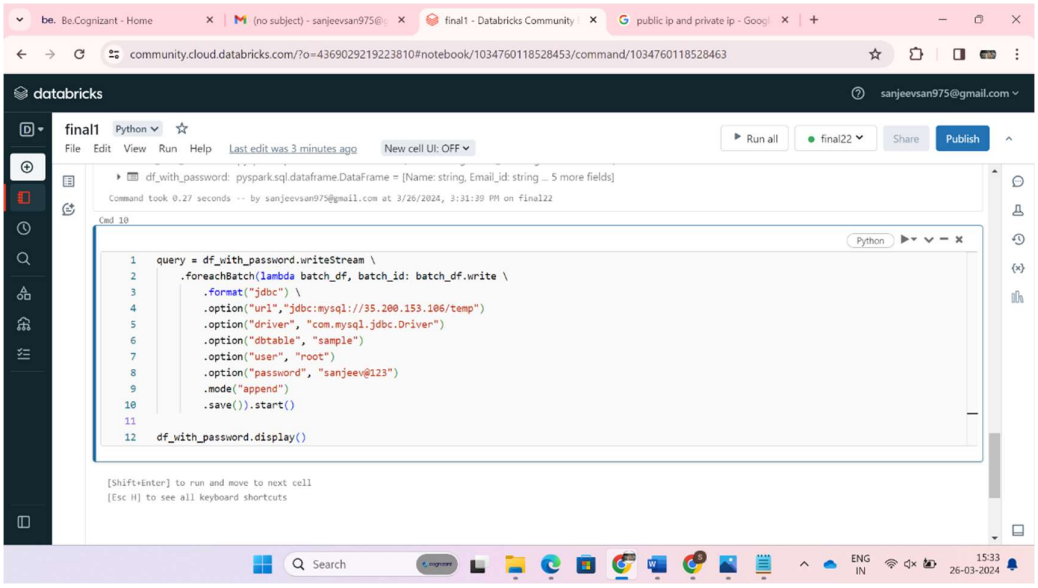
9)Writing to Sink in MySQL Cloud:

- Writes the processed data to a JDBC sink (MySQL Cloud) using a foreach Batch writer.

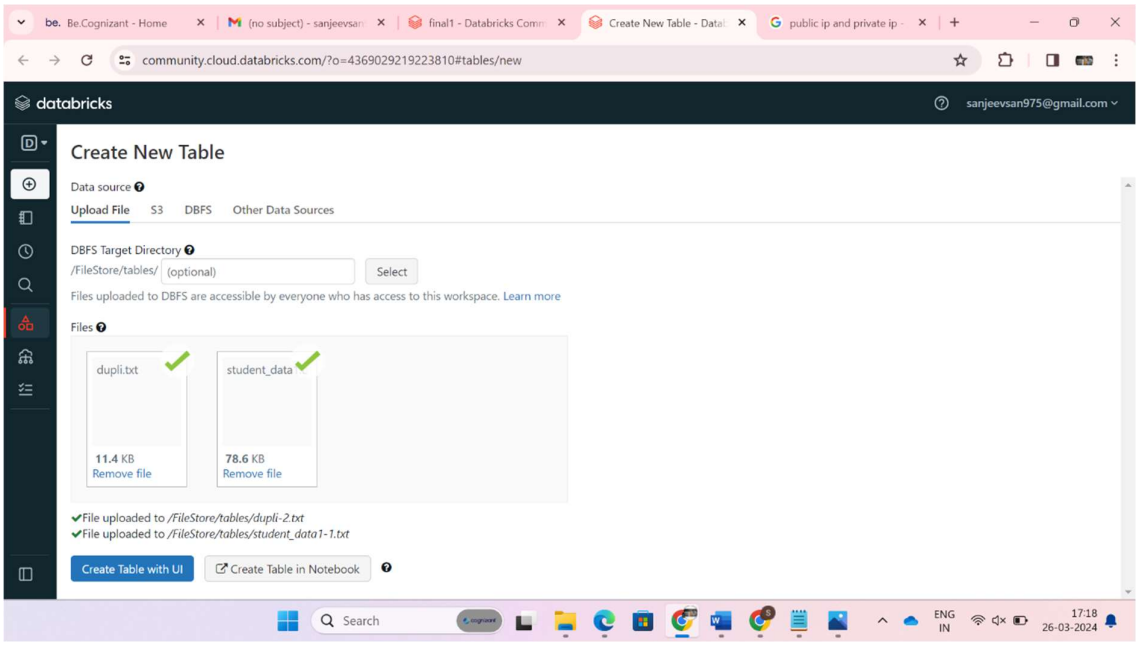
- Specifies connection parameters such as URL IP Address, table name, username, and password.

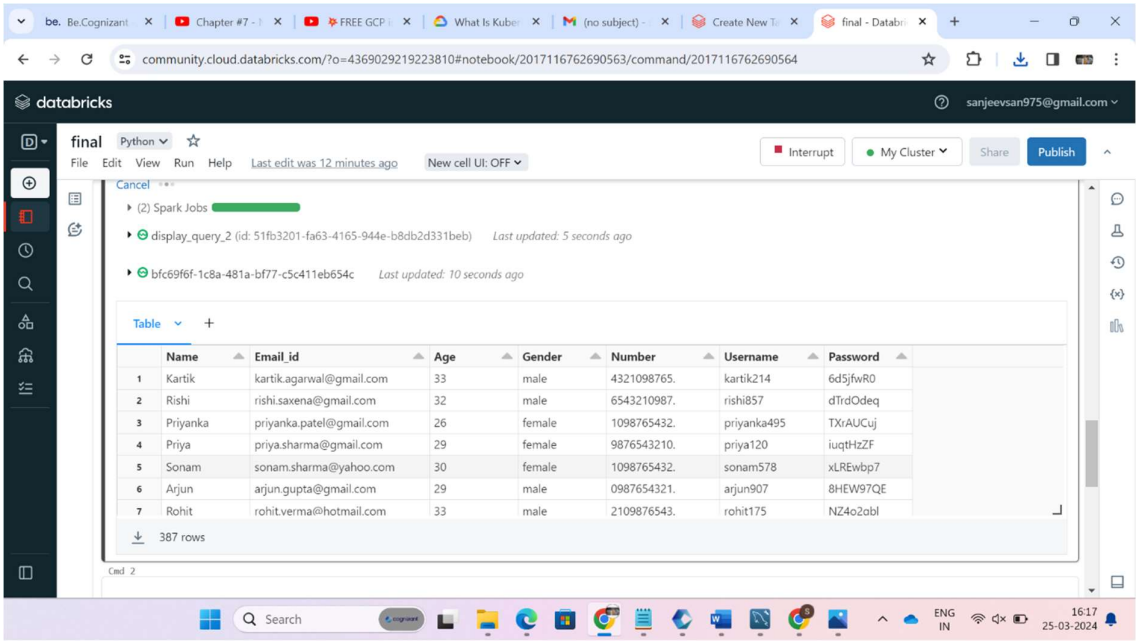
- Create mysql instance in cloud.

-In mysql workbench create new connection and link with Cloud data.

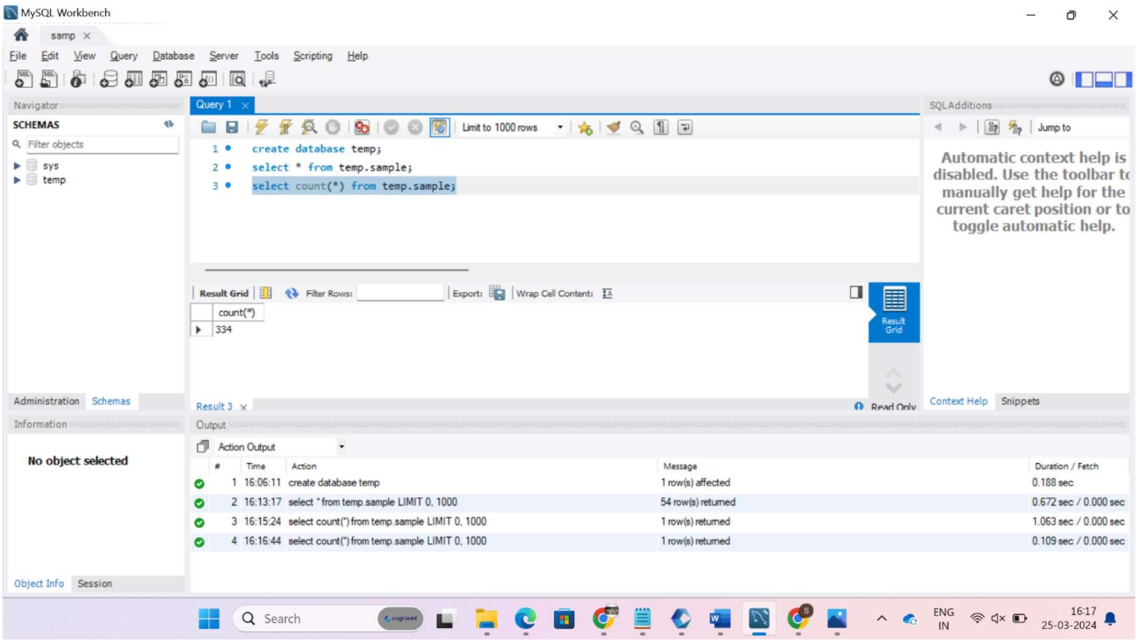


- Now the streaming is started. Now we will insert the file in data bricks as a steam process.

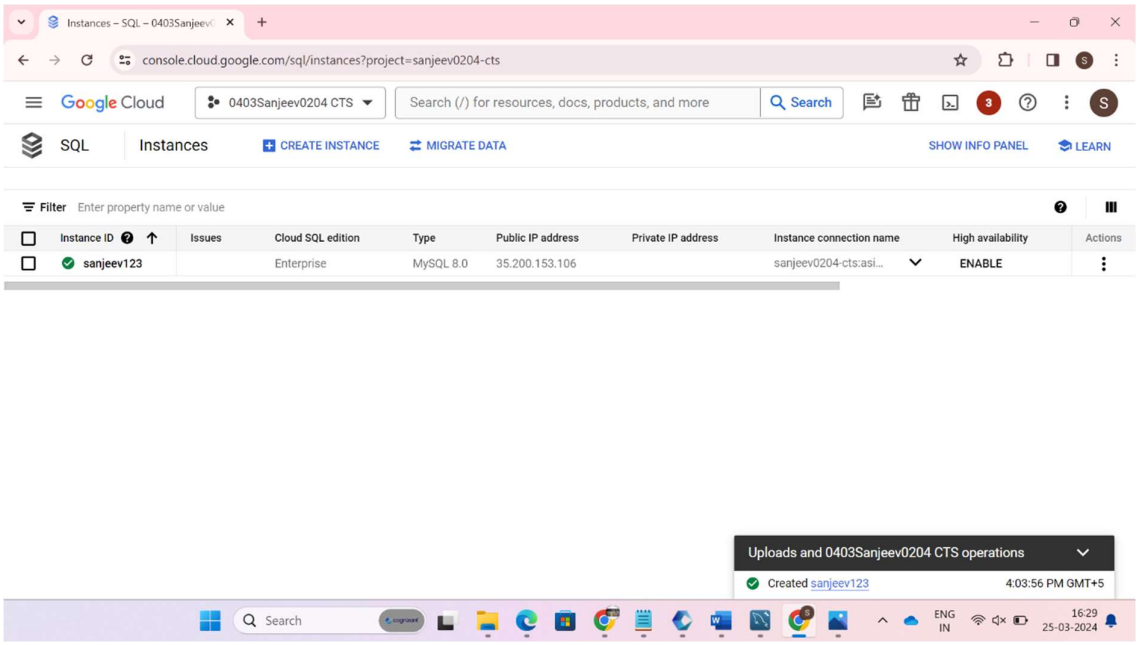




Now data will be stored in MySQL cloud. To get that data we need to open MySQL workbench and using select statement we can see that data.



- before creating a instance in MySQL cloud.



Conclusion:

Streaming and batch ETL are indispensable components of modern data processing architectures, offering organizations the flexibility to process data in real-time and batch modes according to their specific use cases and requirements. By leveraging both methodologies, organizations can derive actionable insights, optimize decision-making processes, and unlock new opportunities for innovation and growth in today's data-driven landscape.