STREAMING SERIALIZATION

USING SPARK

THIRUVENI B
2320446

Streaming Data Serialization: Serialize streaming data into various formats (e.g., JSON) for storage or exchange

METHOD: Streaming Data Serialization

ABSTRACT

In the realm of data manipulation and analysis using Apache Spark's DataFrame API, efficient access to metadata and manipulation of DataFrame attributes are crucial tasks. This abstract delves into the methodologies for accessing metadata such as column names, data types, and schema information, along with strategies for manipulating DataFrame data. The document explores techniques for retrieving DataFrame schema using the ‘printSchema()’method and accessing column names and data types using attributes like ‘schema’, ‘columns’, and ‘dtypes’. It also discusses advanced operations such as renaming columns, changing data types, adding new columns, and dropping columns using methods like ‘withColumnRenamed()’, ‘cast()’, ‘withColumn()’, and ‘drop()’. Through these methods, data engineers and analysts can effectively manage and transform DataFrame metadata, enhancing data processing workflows and facilitating insightful data analysis.

INTRODUCTION:

Apache Spark is a powerful framework for distributed data processing, offering extensive capabilities for working with structured data through its Data Frame API. One crucial aspect of data processing is managing Data Frame metadata, including column names, data types, and schema information. This report provides an overview of how to access and manipulate Data Frame metadata in Apache Spark using various methods from the Spark SQL and Data Frame API.

FEATURES OF APACHE SPARK

Apache Spark has following features.

- Speed − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- Supports multiple languages − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- Advanced Analytics − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

SPARK DATAFRAME

In Spark, DataFrames are the distributed collections of data, organized into rows and columns. Each column in a DataFrame has a name and an associated type.

DataFrames are similar to traditional database tables, which are structured and concise. We can say that DataFrames are relational databases with better optimization techniques.

Spark DataFrames can be created from various sources, such as Hive tables, log tables, external databases, or the existing RDDs. DataFrames allow the processing of huge amounts of data.

WHY DATAFRAME?

When there is not much storage space in memory or on disk, RDDs do not function properly as they get exhausted. Besides, Spark RDDs do not have the concept of schema—the structure of a database that defines its objects. RDDs store both structured and unstructured data together, which is not very efficient.

RDDs cannot modify the system in such a way that it runs more efficiently. RDDs do not allow us to debug errors during the runtime. They store the data as a collection of Java objects.

RDDs use serialization (converting an object into a stream of bytes to allow faster processing) and garbage collection (an automatic memory management technique that detects unused objects and frees them from memory) techniques. This increases the overhead on the memory of the system as they are very lengthy.

This was when DataFrames were introduced to overcome the limitations Spark RDDs had. Now, what makes Spark DataFrames so unique? Let's check out the features of Spark DataFrames that make them so popular.

STREAMING IN SPARK:

- Streaming in Apache Spark refers to the real-time processing of continuously flowing data, enabling the analysis of data as it arrives or changes over time. Spark Streaming, a component of the Apache Spark framework, provides support for processing live data streams with high throughput and fault tolerance.

- It allows developers to write streaming applications using familiar batch processing APIs, enabling seamless integration with existing Spark workflows.
- Spark Streaming ingests data from various sources such as Kafka, Flume, or TCP sockets, processes it using Spark's powerful distributed computing engine, and produces results in near real-time.

SERIALIZATION IN SPARK:

Serialization in Apache Spark refers to the process of converting data objects into a format suitable for transmission or storage, typically for distributed computing purposes.Format such as json, avro, csv,text etc. Spark uses serialization to efficiently transfer data between nodes in a distributed cluster and to persist data to disk or external storage systems.

PROJECT WORK FLOW:

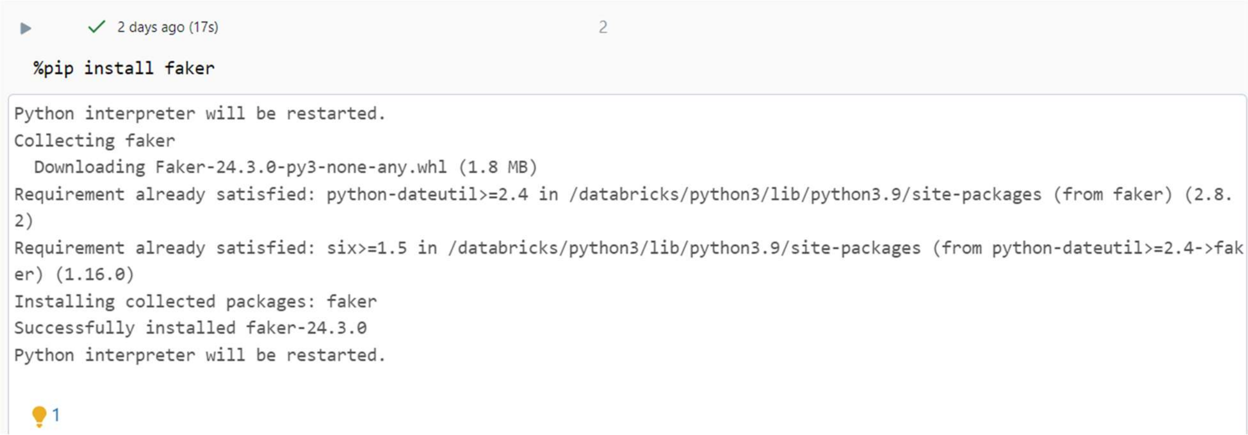
- Generate fake dataset by importing faker().
- Data cleaning- Removing unnecessary columns and removing special characters from column.
- Converting “duration” column from minutes to minutes-seconds using UDF.
- Converting “bpm” column from minutes to seconds using UDF.
- Merge the date, month and year column into single column as “date” using UDF.
- Load the dataframe into Google Cloud Bucket by creating Bucket instance and perform serialization.

❖ GENERATE DATA

Using Python, I generated the data for accessing and manipulating the Data Frame

Step 1:

The **%pip** magic command allows you to install packages from PyPI (Python Package Index) directly within your Databricks environment. Once installed, you can use the **faker** package to generate fake data for testing or simulation purposes.



```
%pip install faker
```

```
Python interpreter will be restarted.
Collecting faker
  Downloading Faker-24.3.0-py3-none-any.whl (1.8 MB)
Requirement already satisfied: python-dateutil>=2.4 in /databricks/python3/lib/python3.9/site-packages (from faker) (2.8.
2)
Requirement already satisfied: six>=1.5 in /databricks/python3/lib/python3.9/site-packages (from python-dateutil>=2.4->fak
er) (1.16.0)
Installing collected packages: faker
Successfully installed faker-24.3.0
Python interpreter will be restarted.
```

Step 2:

I have imported several libraries and modules in Python for data manipulation and visualization using Spark Data Frame (‘pyspark.sql’), Faker (‘faker’). Each of these libraries serves a specific purpose in your data analysis workflow. Here's a brief explanation of each import statement and its role:

1. Import Spark Session and Data Types from PySpark:

- **‘from pyspark.sql import SparkSession’:**

Imports the ‘SparkSession’ class from the ‘pyspark.sql’ module, which is used to interact with Spark SQL and create DataFrame objects.

- **‘from pyspark.sql. types import StructType, StructField, StringType, Integer Type, Timestamp Type, Array Type’:**

Imports various data types and structures from ‘pyspark.sql.types’ module, such as

- **StructType** - StructType is a class used to define the structure of a DataFrame schema. It represents a collection of StructField objects that define the columns and their data types.
- **StructField** - StructField is used within StructType to define individual columns of a DataFrame schema. It specifies the name, data type, and nullable property of each column.
- **StringType** - StringType is a data type used to represent string values in Spark Data Frames. It is used for columns containing text or alphanumeric data.
- **IntegerType** - IntegerType is a data type used to represent integer values in Spark DataFrames. It is used for columns containing whole numbers.
- **FloatType-** FloatType is a data type used to represent float values in Spark DataFrames. It is used for columns containing collections of values.
- **BooleanType** - BooleanType is a data type used to represent Boolean values in Spark DataFrames. It is used for columns containing collections of values.

which are commonly used for defining schema structures for Spark DataFrames.

2. Import DataFrame Functions from PySpark:

- ‘from pyspark.sql.Functions import col, date_format, to_date, to_timestamp’:

Imports DataFrame functions from ‘pyspark.sql. functions’ module, including ‘col ()’ for column selection, date formatting functions like ‘udf’, ‘regexp_replace’,’col,concat’,’date_format’,’lit’,’udf’,’regexp_replace’,’concat’,’d a te_format’.

3. Import Aggregate Functions and Others from PySpark:

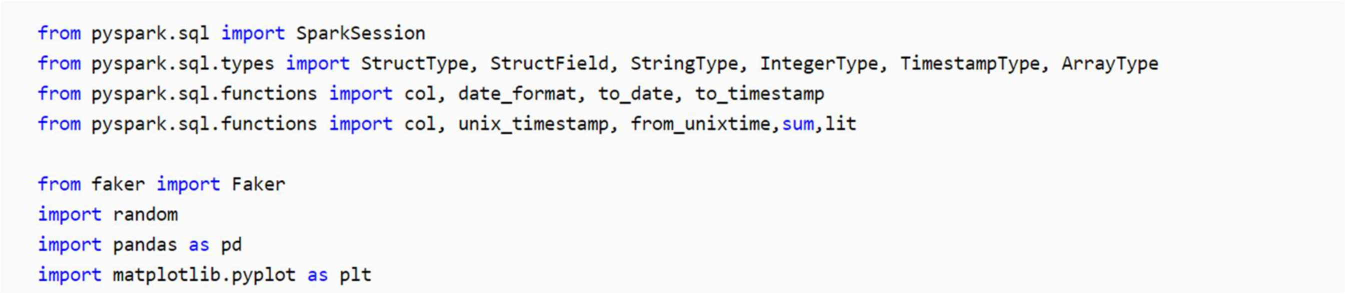
- **‘from pyspark.sql.functions import col lit’:** Imports additional DataFrame functions like ‘lit()’ for creating literal values.

4. Import Faker Library:

- **‘from faker import Faker’:**

Imports the ‘Faker’ class from the ‘faker’ library, which is used for generating fake data such as names, addresses, emails, etc. This is commonly used for testing and simulation purposes.

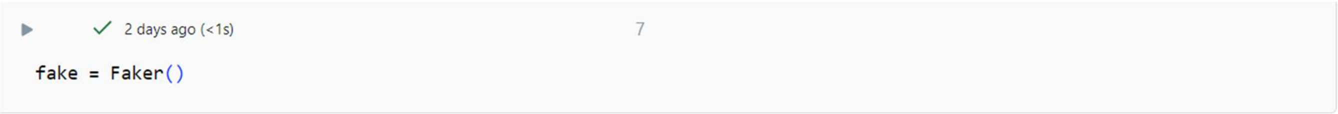
By importing these libraries and modules, you have access to a wide range of functionalities for working with data in Spark DataFrames, generating synthetic data with Faker, performing data analysis with Pandas, and creating visualizations using Matplotlib.



```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType, ArrayType
from pyspark.sql.functions import col, date_format, to_date, to_timestamp
from pyspark.sql.functions import col, unix_timestamp, from_unixtime,sum,lit

from faker import Faker
import random
import pandas as pd
import matplotlib.pyplot as plt
```

Step: 3

This initializes an instance of the Faker class from the faker library in Python. The faker library is commonly used for generating fake data, such as names, addresses, phone numbers, dates, and more, which can be useful for testing, prototyping, and generating sample datasets.



```
fake = Faker()
```

Step: 4

‘**date_time_this_year function** generates fake spotify data based on the specified number of rows (num_rows). It uses the random module for generating random numbers and the faker library for generating fake text, artist name, track name, album name, streaming, year, month, date, bpm, duration, dancibility.

Here's an explanation of each component in the function:

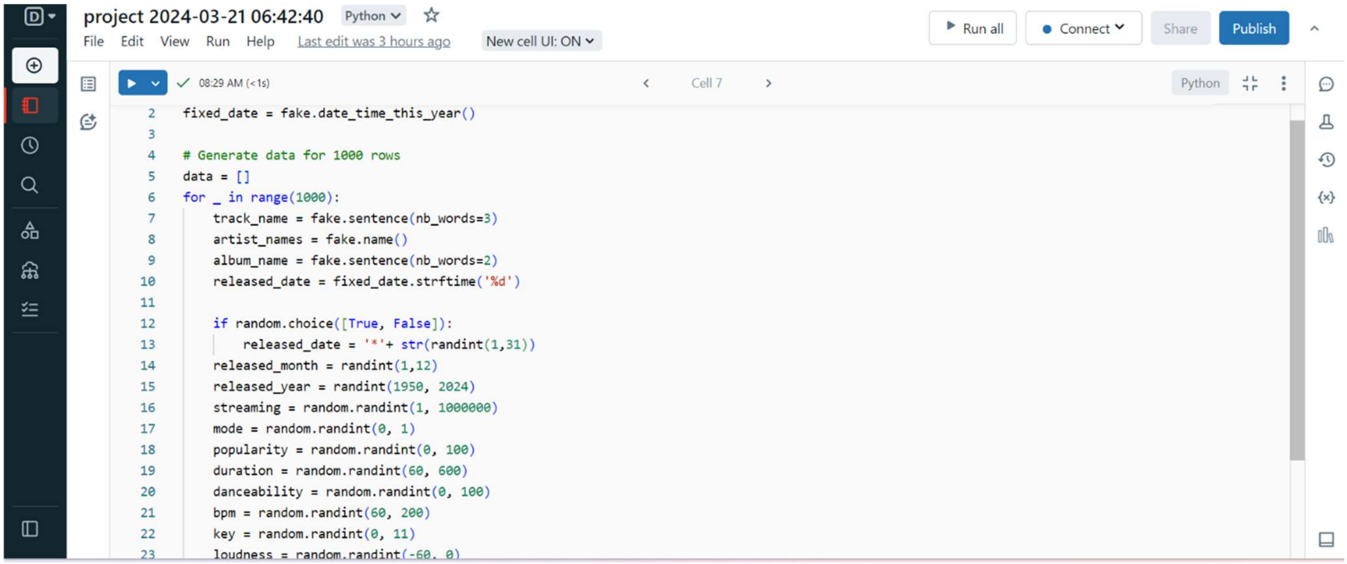
1. **Fake word and text:**

   - Track Name Generation: Uses the fake.sentence() function to generate a fake track name consisting of three words.
   - Artist Name Generation: Generates a fake artist name using the fake.name() function.

2. **Released date, month, year:**

   - Generates a fixed day of the month (released_date) based on a predetermined date.
   - Randomly appends '*' to the released_date if a random choice evaluates to True.
   - Generates a random month (released_month) between 1 and 12.
   - Generates a random year (released_year) between 1950 and the current year (2024).

3. **Streaming:**
   - Streaming Count: Simulates the number of times the track has been streamed, ranging from 1 to 1,000,000.
4. **Mode:** Indicates the mode of the track (major or minor) represented by a binary value (0 or 1).
5. **Popularity:** Represents the popularity of the track as a random integer between 0 and 100.
6. **Duration:** Specifies the duration of the track in seconds, ranging from 60 to 600 seconds.
7. **Danceability:** Assigns a danceability score to the track, ranging from 0 to 100.
8. Beats Per Minute (BPM): Sets the tempo of the track in beats per minute, ranging from 60 to 200.
9. **Key:** Specifies the musical key of the track, represented by a random integer between 0 and 11.
10. **Loudness:** Sets the loudness level of the track in decibels, ranging from -60 dB to 0 dB.



```
fixed_date = fake.date_time_this_year()

# Generate data for 1000 rows
data = []
for _ in range(1000):
    track_name = fake.sentence(nb_words=3)
    artist_names = fake.name()
    album_name = fake.sentence(nb_words=2)
    released_date = fixed_date.strftime('%d')

    if random.choice([True, False]):
        released_date = '*'+ str(randint(1,31))
    released_month = randint(1,12)
    released_year = randint(1950, 2024)
    streaming = random.randint(1, 1000000)
    mode = random.randint(0, 1)
    popularity = random.randint(0, 100)
    duration = random.randint(60, 600)
    danceability = random.randint(0, 100)
    bpm = random.randint(60, 200)
    key = random.randint(0, 11)
    loudness = random.randint(-60, 0)
```

‘the date_time function generate (1000)’ function call generates a list of 1000 tuples, with each tuple representing a piece of fake spotify data. You can adjust the number of rows (num_rows) as needed to generate more or fewer data entries.

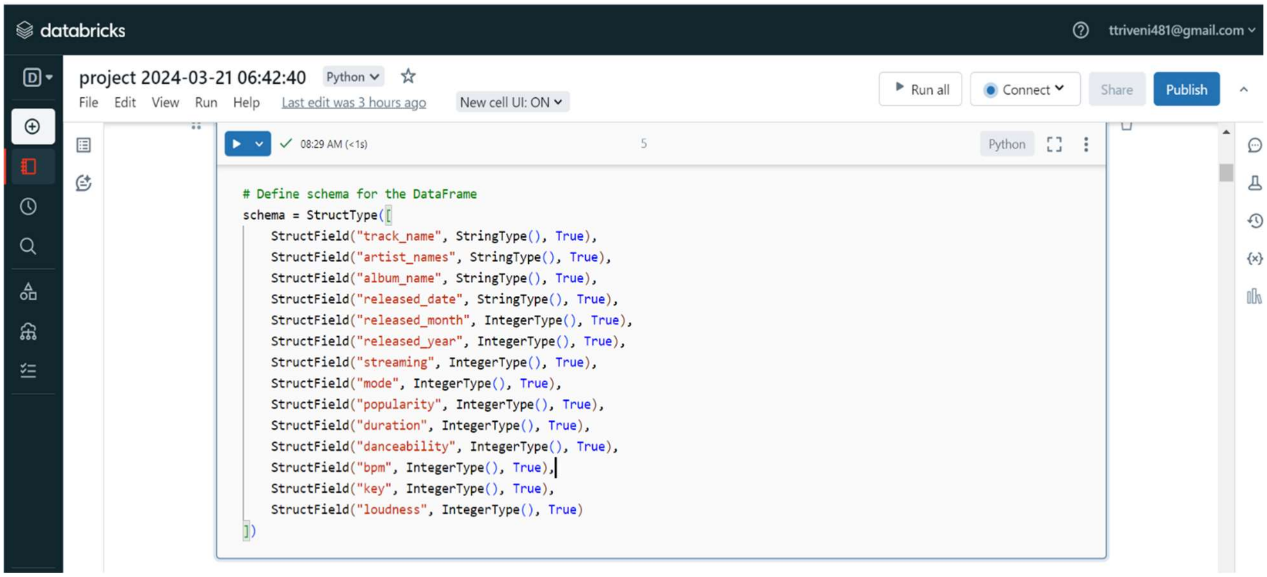
Step: 5

Defining a schema using the StructType and StructField classes from the pyspark.sql.types module in PySpark. This schema is used to define the structure and data types of the columns for a DataFrame in PySpark.

Here's an explanation of each field in the schema:

1. **track name"** (StringType): Represents the name of the music track.
2. **"artist names"** (StringType): Represents the name of the artist or artists associated with the track.
3. **"album name"** (StringType): Represents the name of the album to which the track belongs.
4. **"released date"** (StringType): Represents the day of the month when the track was released. This field is of StringType, indicating that the date is represented as a string.
5. **"released month"** (IntegerType): Represents the month when the track was released. This field is of IntegerType, indicating that the month is represented as an integer.
6. **"released year"** (IntegerType): Represents the year when the track was released. This field is of IntegerType, indicating that the year is represented as an integer.
7. **"streaming"** (IntegerType): Represents the number of times the track has been streamed.

8. **"mode"** (IntegerType): Represents the mode of the track (major or minor), typically encoded as 0 for minor and 1 for major.
9. **"popularity"** (IntegerType): Represents the popularity score of the track.
10. **"duration"** (IntegerType): Represents the duration of the track in seconds.
11. **"danceability"** (IntegerType): Represents the danceability score of the track.
12. **"bpm"** (IntegerType): Represents the tempo of the track in beats per minute (BPM).
13. **"key"** (IntegerType): Represents the musical key of the track, typically encoded as an integer value between 0 and 11.
14. **"loudness"** (IntegerType): Represents the loudness level of the track in decibels (dB)

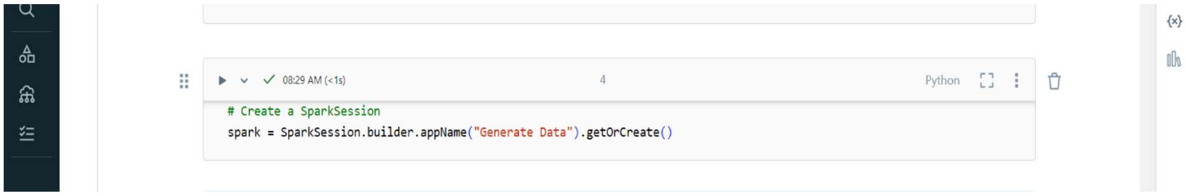


```
# Define schema for the DataFrame
schema = StructType([
    StructField("track_name", StringType(), True),
    StructField("artist_names", StringType(), True),
    StructField("album_name", StringType(), True),
    StructField("released_date", StringType(), True),
    StructField("released_month", IntegerType(), True),
    StructField("released_year", IntegerType(), True),
    StructField("streaming", IntegerType(), True),
    StructField("mode", IntegerType(), True),
    StructField("popularity", IntegerType(), True),
    StructField("duration", IntegerType(), True),
    StructField("danceability", IntegerType(), True),
    StructField("bpm", IntegerType(), True),
    StructField("key", IntegerType(), True),
    StructField("loudness", IntegerType(), True)
])
```

Step : 6

The line **spark=SparkSession.builder.appName("Generate Data").getOrCreate()** creates a Spark session named "Metadata" using the SparkSession builder in Apache Spark.

1. **SparkSession:**
   - SparkSession is the entry point to programming Spark with the Dataset and DataFrame API. It provides a unified interface for interacting with Spark functionality and allows you to work with structured data.
2. **builder:**
   - The builder method is used to create a Builder object for configuring and setting up the Spark session.
3. **appName("Generate Data"):**
   - The appName method sets the name of the Spark application to "Metadata". This name is displayed in the Spark UI and logs, making it easier to identify your application.
4. **getOrCreate ():**
   - The getOrCreate method checks if there is an existing Spark session available. If an active Spark session exists, it returns that session. Otherwise, it creates a new Spark session based on the configuration set using the builder object.



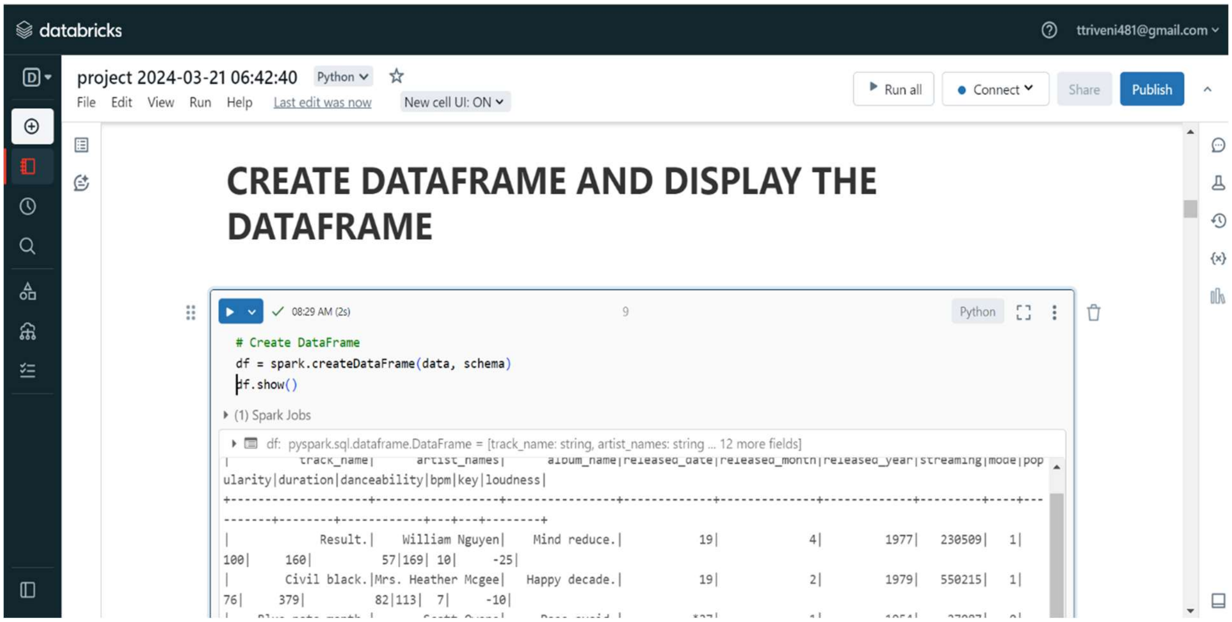
```
# Create a SparkSession
spark = SparkSession.builder.appName("Generate Data").getOrCreate()
```

Step : 7

Creating a PySpark DataFrame named df using the provided spotify dataset and schema.

Here's how the code works:

1. spotify_data: This is assumed to be a list of tuples, where each tuple represents a row of data for the DataFrame. Each tuple should follow the structure defined by the schema you provided earlier.
2. schema: This is the schema that defines the structure of the DataFrame, including the names, data types, and nullable settings of each column.
3. spark.createDataFrame(data, schema): This method creates a PySpark DataFrame (df) using the provided data (data) and schema (schema). The data is mapped to the columns defined in the schema, and the DataFrame is created accordingly.



CREATE DATAFRAME AND DISPLAY THE DATAFRAME

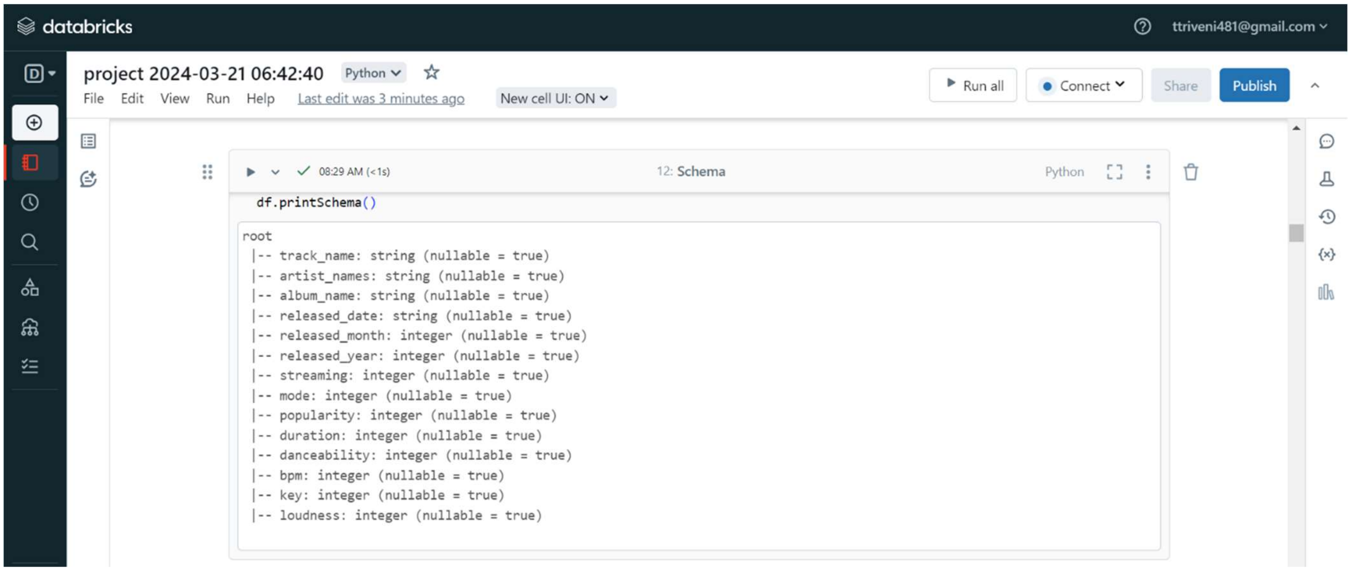
```
# Create DataFrame
df = spark.createDataFrame(data, schema)
df.show()
```

df: pyspark.sql.dataframe.DataFrame = [track_name: string, artist_names: string ... 12 more fields]

```
|         track_name|      artist_names|      album_name|released_date|released_month|released_year|streaming|mode|pop
ularity|duration|danceability|bpm|key|loudness|
+-------------------+------------------+----------------+-------------+--------------+-------------+---------+----+---
-------+--------+------------+---+---+--------+
|            Result.|    William Nguyen|    Mind reduce.|           19|             4|         1977|   230509|   1|
100|     160|          57|169| 10|     -25|
|       Civil black.|Mrs. Heather Mcgee|   Happy decade.|           19|             2|         1979|   550215|   1|
76|     379|          82|113|  7|     -10|
```

Step:8

To print the schema of a PySpark DataFrame before performing any manipulations, you can use the printSchema() method on the DataFrame object. it will print the schema of the Data Frame- df to the console or output window. The schema will include the names, data types of each column in the DataFrame.

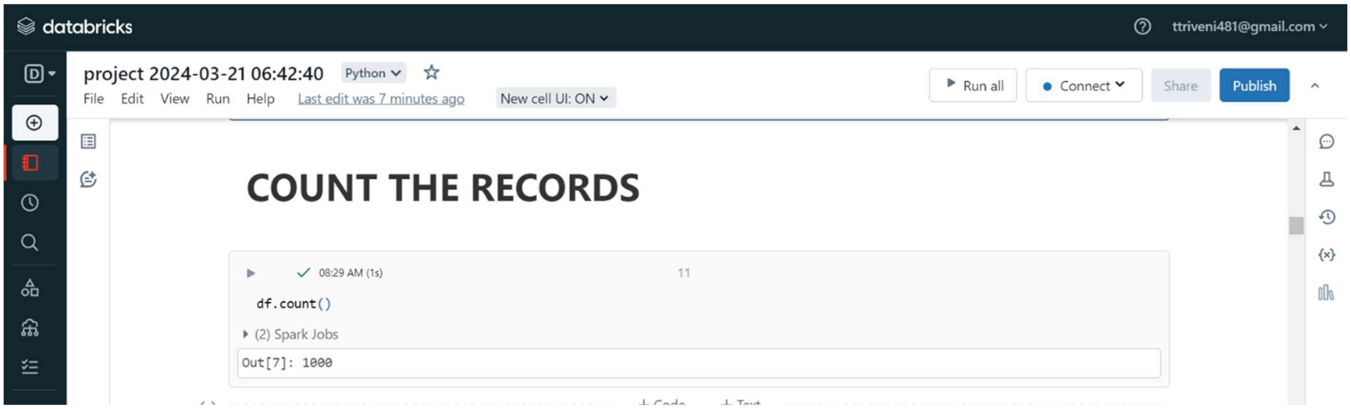


```
df.printSchema()
```

```
root
 |-- track_name: string (nullable = true)
 |-- artist_names: string (nullable = true)
 |-- album_name: string (nullable = true)
 |-- released_date: string (nullable = true)
 |-- released_month: integer (nullable = true)
 |-- released_year: integer (nullable = true)
 |-- streaming: integer (nullable = true)
 |-- mode: integer (nullable = true)
 |-- popularity: integer (nullable = true)
 |-- duration: integer (nullable = true)
 |-- danceability: integer (nullable = true)
 |-- bpm: integer (nullable = true)
 |-- key: integer (nullable = true)
 |-- loudness: integer (nullable = true)
```

Step: 9

Using the count function, total records of the dataframe will be showed.



COUNT THE RECORDS

```
df.count()
```

```
Out[7]: 1000
```

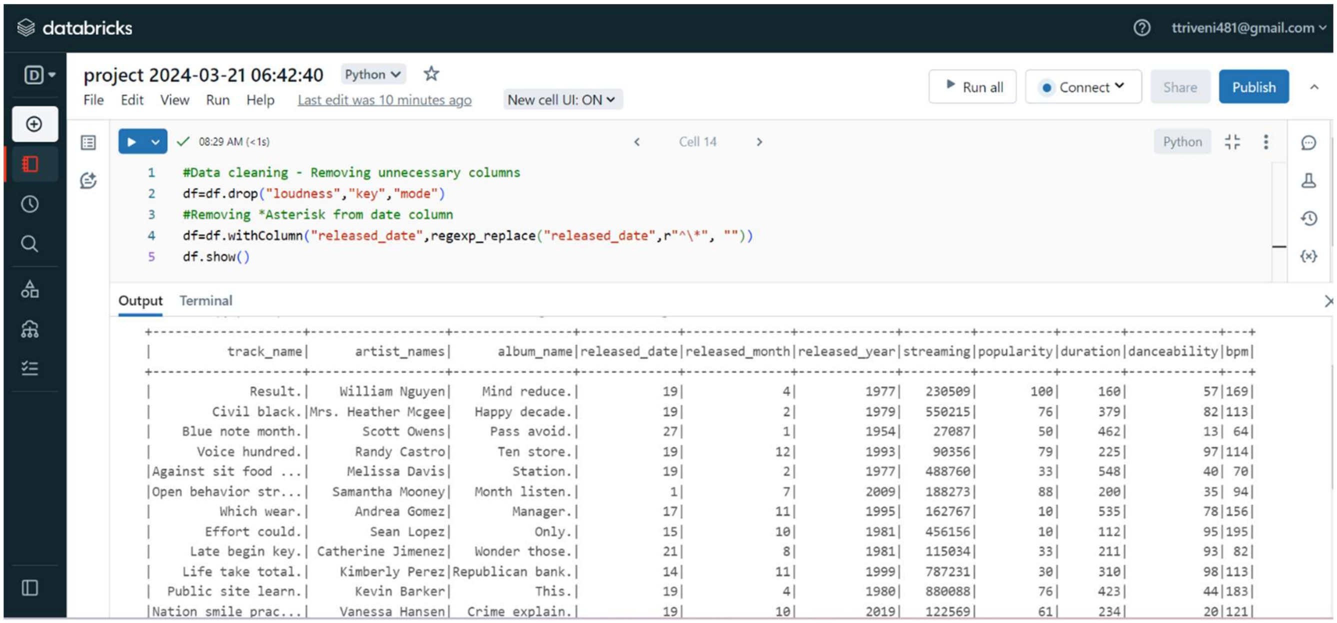
Step: 10

Handle missing or null values by imputing them with appropriate values, dropping unnecessary rows or columns. Removing special characters from column.

‘drop’: The drop function is used to remove one or more columns from a DataFrame. It takes column names as arguments and returns a new DataFrame with the specified columns removed.

‘withColumn’: The withColumn function is used to add a new column to a DataFrame or replace an existing column with a modified version. It takes a column name and an expression as arguments, where the expression computes the values for the new column based on the existing columns.

‘regexp’: regexp is not a function in Spark directly. However, regular expressions (regex) can be used with various functions like filter, where, select, like, rlike, etc., for pattern matching and string manipulation.



```
#Data cleaning - Removing unnecessary columns
df=df.drop("loudness","key","mode")
#Removing *Asterisk from date column
df=df.withColumn("released_date",regexp_replace("released_date",r"^\*", ""))
df.show()
```

```
+--------------------+------------------+----------------+-------------+--------------+-------------+---------+----------+--------+------------+---+
|          track_name|      artist_names|      album_name|released_date|released_month|released_year|streaming|popularity|duration|danceability|bpm|
+--------------------+------------------+----------------+-------------+--------------+-------------+---------+----------+--------+------------+---+
|             Result.|    William Nguyen|    Mind reduce.|           19|             4|         1977|   230509|       100|     160|          57|169|
|        Civil black.|Mrs. Heather Mcgee|   Happy decade.|           19|             2|         1979|   550215|        76|     379|          82|113|
|    Blue note month.|       Scott Owens|     Pass avoid.|           27|             1|         1954|    27087|        50|     462|          13| 64|
|      Voice hundred.|      Randy Castro|      Ten store.|           19|            12|         1993|    90356|        79|     225|          97|114|
|Against sit food ...|     Melissa Davis|        Station.|           19|             2|         1977|   488760|        33|     548|          40| 70|
|Open behavior str...|   Samantha Mooney|   Month listen.|            1|             7|         2009|   188273|        88|     200|          35| 94|
|         Which wear.|      Andrea Gomez|        Manager.|           17|            11|         1995|   162767|        10|     535|          78|156|
|       Effort could.|        Sean Lopez|           Only.|           15|            10|         1981|   456156|        10|     112|          95|195|
|     Late begin key.| Catherine Jimenez|   Wonder those.|           21|             8|         1981|   115034|        33|     211|          93| 82|
|    Life take total.|    Kimberly Perez|Republican bank.|           14|            11|         1999|   787231|        30|     310|          98|113|
|  Public site learn.|      Kevin Barker|           This.|           19|             4|         1980|   880088|        76|     423|          44|183|
|Nation smile prac...|    Vanessa Hansen|  Crime explain.|           19|            10|         2019|   122569|        61|     234|          20|121|
```

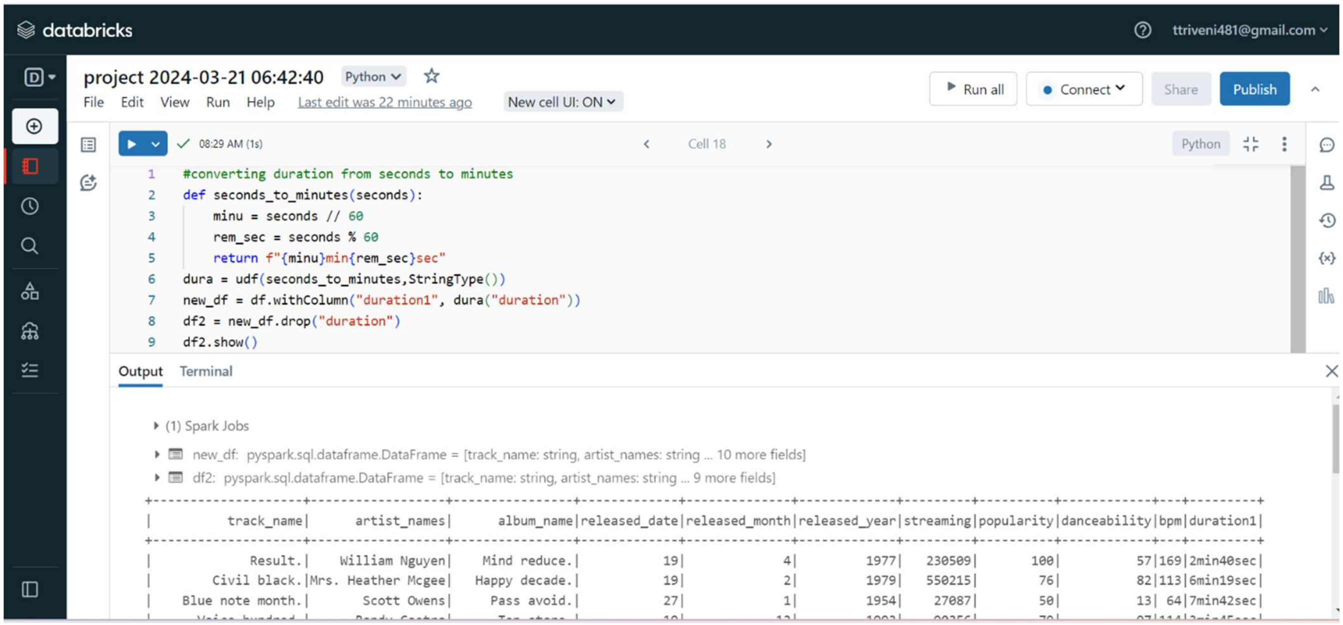
Step: 11

Converting the duration column from minutes to minutes -seconds by performing UDF functions.

1. **Function Definition (seconds_to_minutes):**
   - Defines a Python function named seconds_to_minutes that takes a number of seconds as input and returns a string representing the duration in minutes and seconds.
   - Inside the function, it calculates the number of minutes (minu) and the remaining seconds (rem_sec) after converting the input seconds to minutes.
   - Returns a formatted string representing the duration in the format "XminYsec".
2. **User Defined Function (UDF) Creation (dura):**
   - Creates a User Defined Function (UDF) named dura using the seconds_to_minutes function defined above.
   - The UDF is defined to operate on the duration column of the DataFrame and convert each value from seconds to minutes using the seconds_to_minutes function.
   - Specifies the return type of the UDF as StringType().
3. **DataFrame Transformation (withColumn):**
   - Applies the UDF dura to the duration column of the DataFrame (df) using the withColumn function.
   - Creates a new DataFrame (new_df) with the modified duration column representing the duration in minutes and seconds.
4. **DataFrame Manipulation (drop):**

- Creates another new DataFrame (df2) by dropping the original duration column from new_df using the drop function.
- This operation effectively removes the original duration column, leaving only the modified duration column representing the duration in minutes and seconds.

5. **Displaying DataFrame (show):**
    - Displaying the dataframe.



```
#converting duration from seconds to minutes
def seconds_to_minutes(seconds):
    minu = seconds // 60
    rem_sec = seconds % 60
    return f"{minu}min{rem_sec}sec"
dura = udf(seconds_to_minutes,StringType())
new_df = df.withColumn("duration1", dura("duration"))
df2 = new_df.drop("duration")
df2.show()
```

```
(1) Spark Jobs
new_df:  pyspark.sql.dataframe.DataFrame = [track_name: string, artist_names: string ... 10 more fields]
df2:  pyspark.sql.dataframe.DataFrame = [track_name: string, artist_names: string ... 9 more fields]
+------------------+------------------+-----------------+-------------+--------------+-------------+---------+----------+------------+---+---------+
|        track_name|      artist_names|       album_name|released_date|released_month|released_year|streaming|popularity|danceability|bpm|duration1|
+------------------+------------------+-----------------+-------------+--------------+-------------+---------+----------+------------+---+---------+
|           Result.|    William Nguyen|     Mind reduce.|           19|             4|         1977|   230509|       100|          57|169|2min40sec|
|      Civil black.|Mrs. Heather Mcgee|    Happy decade.|           19|             2|         1979|   550215|        76|          82|113|6min19sec|
|  Blue note month.|       Scott Owens|      Pass avoid.|           27|             1|         1954|    27087|        50|          13| 64|7min42sec|
```

Step: 12

Converting the bpm(Beat Per Minute) to seconds and make a new column name ‘bpm_sec’ and dropping the existing bpm column.

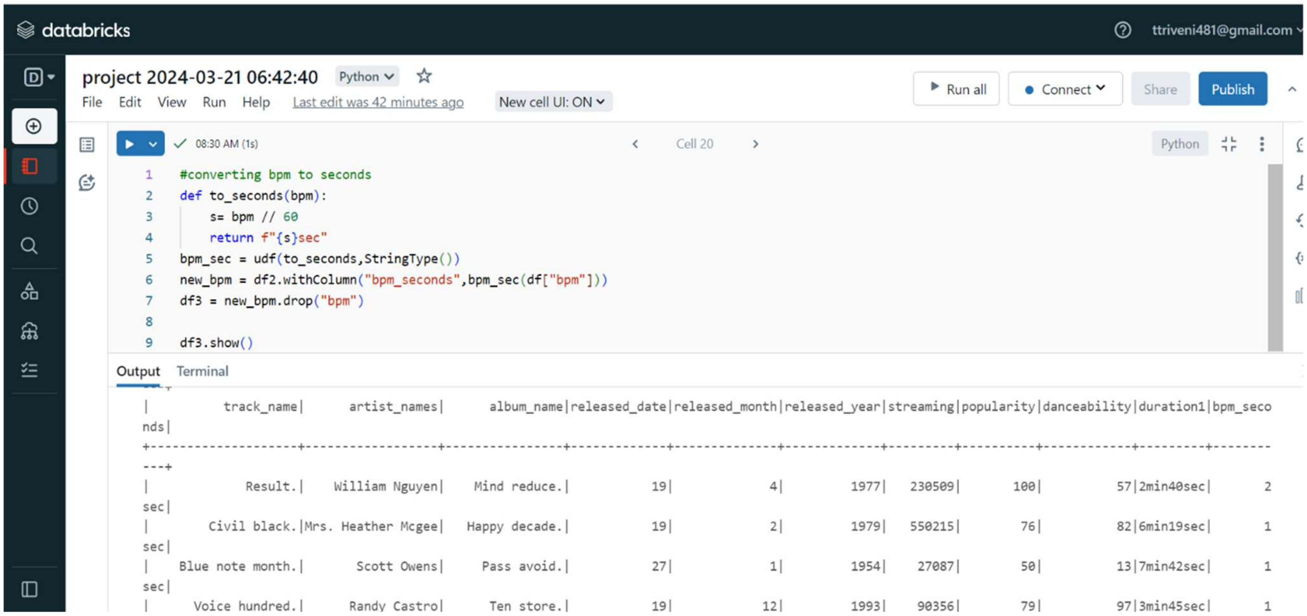
Function Definition (to_seconds):

- Defines a Python function named to_seconds that takes a BPM value as input and returns a string representing the BPM converted to beats per second.

- Inside the function, it calculates the number of beats per second (s) by dividing the BPM by 60 (since there are 60 seconds in a minute).
- Returns a formatted string representing the BPM in beats per second.

User Defined Function (UDF) Creation (bpm_sec):

- Creates a User Defined Function (UDF) named bpm_sec using the to_seconds function.
- The UDF is defined to operate on the BPM column of the DataFrame and convert each value from beats per minute to beats per second using the to_seconds function.
- Specifies the return type of the UDF as StringType().

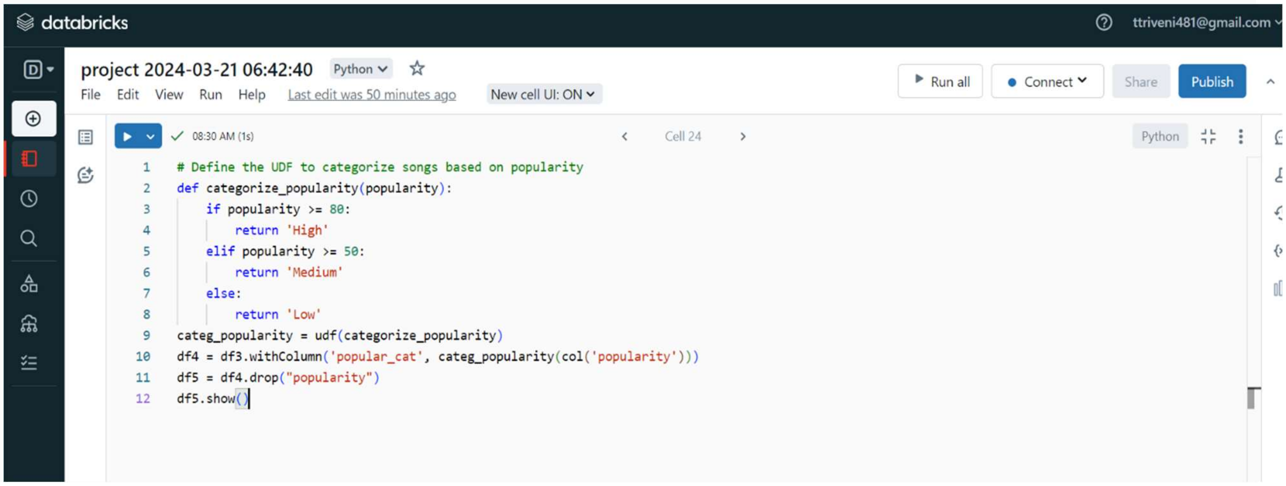


```
#converting bpm to seconds
def to_seconds(bpm):
    s= bpm // 60
    return f"{s}sec"
bpm_sec = udf(to_seconds,StringType())
new_bpm = df2.withColumn("bpm_seconds",bpm_sec(df["bpm"]))
df3 = new_bpm.drop("bpm")

df3.show()
```

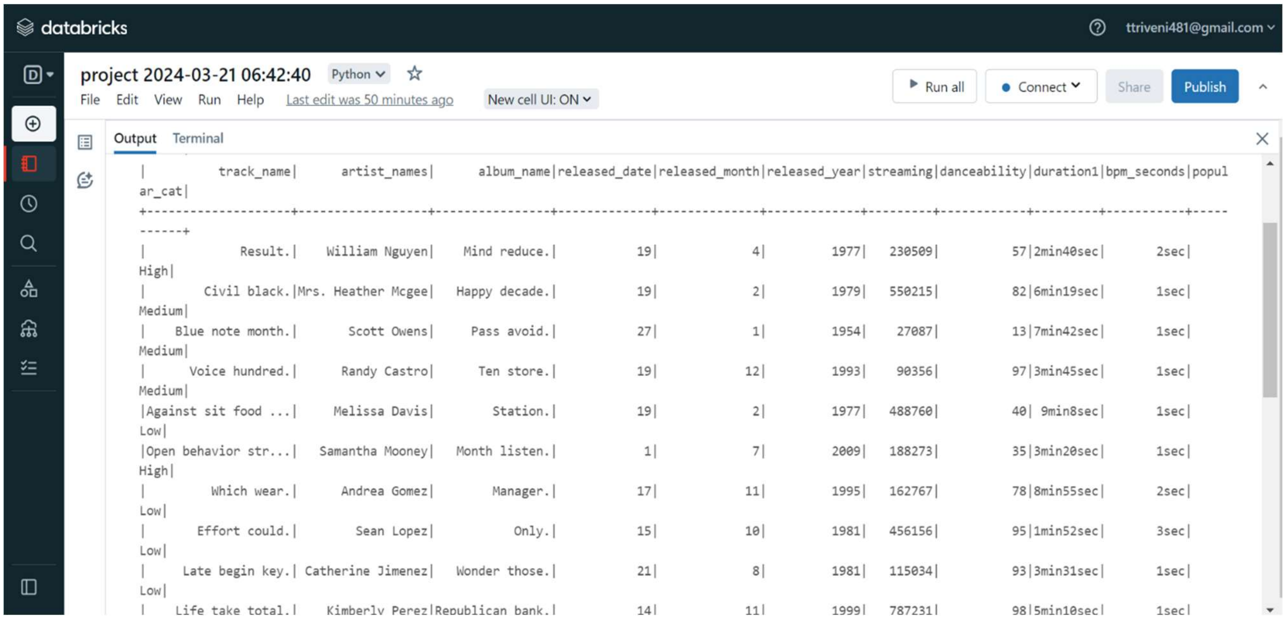
```
|         track_name|      artist_names|      album_name|released_date|released_month|released_year|streaming|popularity|danceability|duration1|bpm_seco
nds|
+-------------------+------------------+----------------+-------------+--------------+-------------+---------+----------+------------+---------+--------
---+
|            Result.|    William Nguyen|    Mind reduce.|           19|             4|         1977|   230509|       100|          57|2min40sec|       2
sec|
|       Civil black.|Mrs. Heather Mcgee|   Happy decade.|           19|             2|         1979|   550215|        76|          82|6min19sec|       1
sec|
|   Blue note month.|       Scott Owens|     Pass avoid.|           27|             1|         1954|    27087|        50|          13|7min42sec|       1
sec|
|     Voice hundred.|      Randy Castro|      Ten store.|           19|            12|         1993|    90356|        79|          97|3min45sec|       1
```

Step: 13

Define the UDF to categorize songs based on popularity and shows the dataframe.



```python
# Define the UDF to categorize songs based on popularity
def categorize_popularity(popularity):
    if popularity >= 80:
        return 'High'
    elif popularity >= 50:
        return 'Medium'
    else:
        return 'Low'
categ_popularity = udf(categorize_popularity)
df4 = df3.withColumn('popular_cat', categ_popularity(col('popularity')))
df5 = df4.drop("popularity")
df5.show()
```

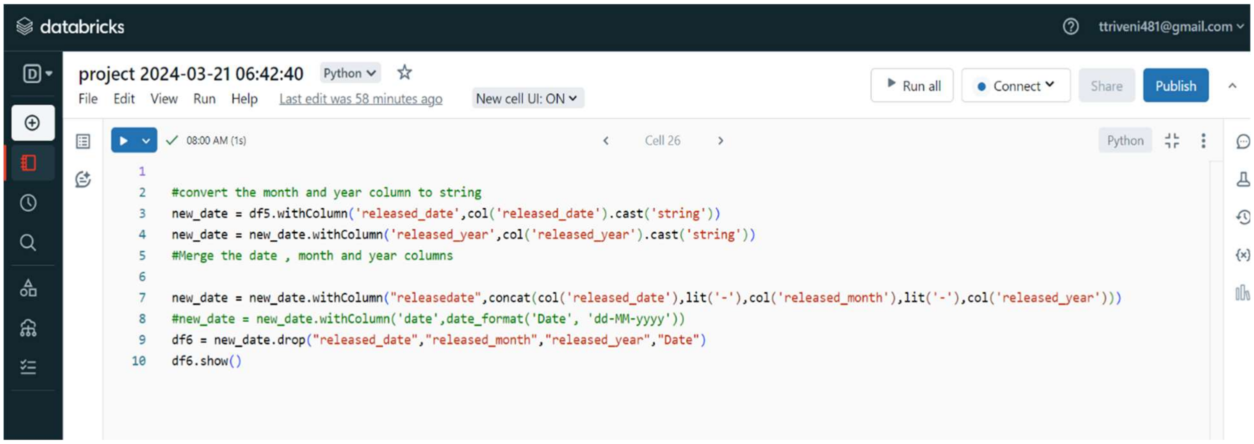


```
|          track_name|      artist_names|      album_name|released_date|released_month|released_year|streaming|danceability|duration1|bpm_seconds|popul
ar_cat|
+--------------------+------------------+----------------+-------------+--------------+-------------+---------+------------+---------+-----------+-----
------+
|             Result.|    William Nguyen|    Mind reduce.|           19|             4|         1977|   230509|          57|2min40sec|       2sec|
High|
|        Civil black.|Mrs. Heather Mcgee|   Happy decade.|           19|             2|         1979|   550215|          82|6min19sec|       1sec|
Medium|
|    Blue note month.|       Scott Owens|     Pass avoid.|           27|             1|         1954|    27087|          13|7min42sec|       1sec|
Medium|
|      Voice hundred.|      Randy Castro|      Ten store.|           19|            12|         1993|    90356|          97|3min45sec|       1sec|
Medium|
|Against sit food ...|     Melissa Davis|        Station.|           19|             2|         1977|   488760|          40| 9min8sec|       1sec|
Low|
|Open behavior str...|   Samantha Mooney|   Month listen.|            1|             7|         2009|   188273|          35|3min20sec|       1sec|
High|
|         Which wear.|      Andrea Gomez|        Manager.|           17|            11|         1995|   162767|          78|8min55sec|       2sec|
Low|
|       Effort could.|        Sean Lopez|           Only.|           15|            10|         1981|   456156|          95|1min52sec|       3sec|
Low|
|     Late begin key.| Catherine Jimenez|   Wonder those.|           21|             8|         1981|   115034|          93|3min31sec|       1sec|
Low|
|    Life take total.|    Kimberly Perez|Republican bank.|           14|            11|         1999|   787231|          98|5min10sec|       1sec|
```

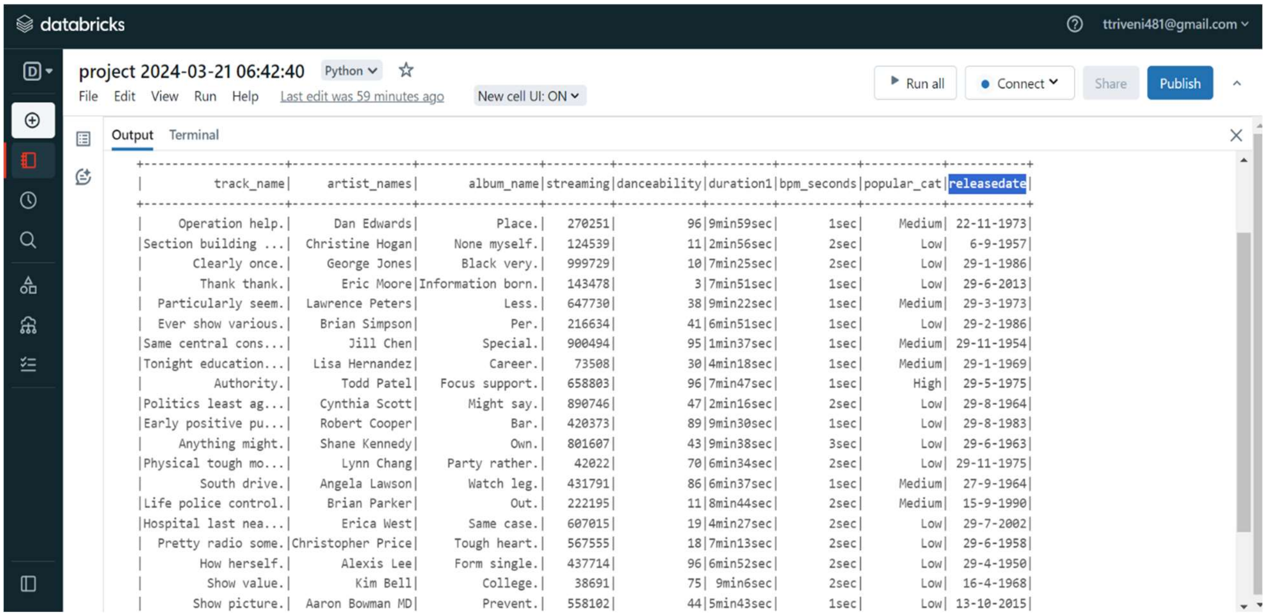
Step: 14

To convert the month and year to string by casting.

1. **‘col_name().cast(Datatype)’:**This command is used for convert datatype for the column.
2. **‘concat’ :** Using concat function merge the date, month, year column.



```
#convert the month and year column to string
new_date = df5.withColumn('released_date',col('released_date').cast('string'))
new_date = new_date.withColumn('released_year',col('released_year').cast('string'))
#Merge the date , month and year columns

new_date = new_date.withColumn("releasedate",concat(col('released_date'),lit('-'),col('released_month'),lit('-'),col('released_year')))
#new_date = new_date.withColumn('date',date_format('Date', 'dd-MM-yyyy'))
df6 = new_date.drop("released_date","released_month","released_year","Date")
df6.show()
```

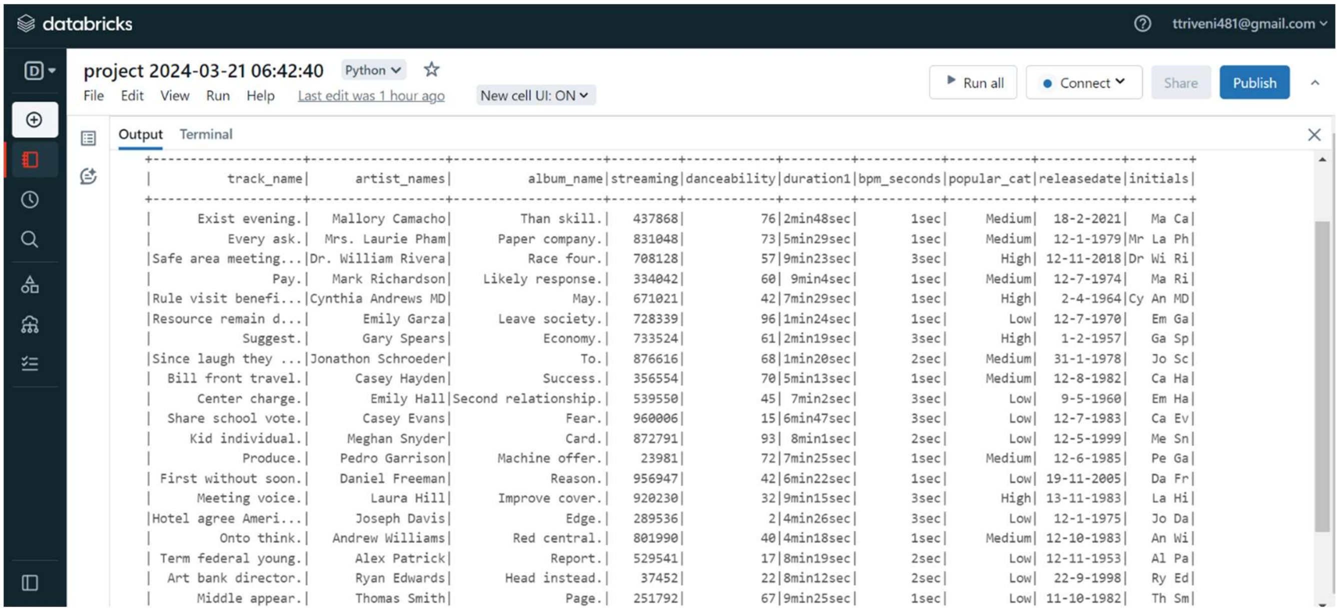


```
+--------------------+-----------------+-----------------+---------+------------+---------+-----------+-----------+-----------+
|          track_name|     artist_names|       album_name|streaming|danceability|duration1|bpm_seconds|popular_cat|releasedate|
+--------------------+-----------------+-----------------+---------+------------+---------+-----------+-----------+-----------+
|     Operation help.|      Dan Edwards|           Place.|   270251|          96|9min59sec|       1sec|     Medium| 22-11-1973|
|Section building ...|  Christine Hogan|     None myself.|   124539|          11|2min56sec|       2sec|        Low|   6-9-1957|
|       Clearly once.|     George Jones|      Black very.|   999729|          10|7min25sec|       2sec|        Low|  29-1-1986|
|        Thank thank.|       Eric Moore|Information born.|   143478|           3|7min51sec|       1sec|        Low|  29-6-2013|
|  Particularly seem.|  Lawrence Peters|            Less.|   647730|          38|9min22sec|       1sec|     Medium|  29-3-1973|
|  Ever show various.|    Brian Simpson|             Per.|   216634|          41|6min51sec|       1sec|        Low|  29-2-1986|
|Same central cons...|        Jill Chen|         Special.|   900494|          95|1min37sec|       1sec|     Medium| 29-11-1954|
|Tonight education...|   Lisa Hernandez|          Career.|    73508|          30|4min18sec|       1sec|     Medium|  29-1-1969|
|          Authority.|       Todd Patel|   Focus support.|   658803|          96|7min47sec|       1sec|       High|  29-5-1975|
|Politics least ag...|    Cynthia Scott|       Might say.|   890746|          47|2min16sec|       2sec|        Low|  29-8-1964|
|Early positive pu...|    Robert Cooper|             Bar.|   420373|          89|9min30sec|       1sec|        Low|  29-8-1983|
|     Anything might.|    Shane Kennedy|             Own.|   801607|          43|9min38sec|       3sec|        Low|  29-6-1963|
|Physical tough mo...|       Lynn Chang|    Party rather.|    42022|          70|6min34sec|       2sec|        Low| 29-11-1975|
|        South drive.|    Angela Lawson|       Watch leg.|   431791|          86|6min37sec|       1sec|     Medium|  27-9-1964|
|Life police control.|     Brian Parker|             Out.|   222195|          11|8min44sec|       2sec|     Medium|  15-9-1990|
|Hospital last nea...|       Erica West|       Same case.|   607015|          19|4min27sec|       2sec|        Low|  29-7-2002|
|  Pretty radio some.|Christopher Price|     Tough heart.|   567555|          18|7min13sec|       2sec|        Low|  29-6-1958|
|        How herself.|       Alexis Lee|     Form single.|   437714|          96|6min52sec|       2sec|        Low|  29-4-1950|
|         Show value.|         Kim Bell|         College.|    38691|          75| 9min6sec|       2sec|        Low|  16-4-1968|
|       Show picture.|  Aaron Bowman MD|         Prevent.|   558102|          44|5min43sec|       1sec|        Low| 13-10-2015|
```

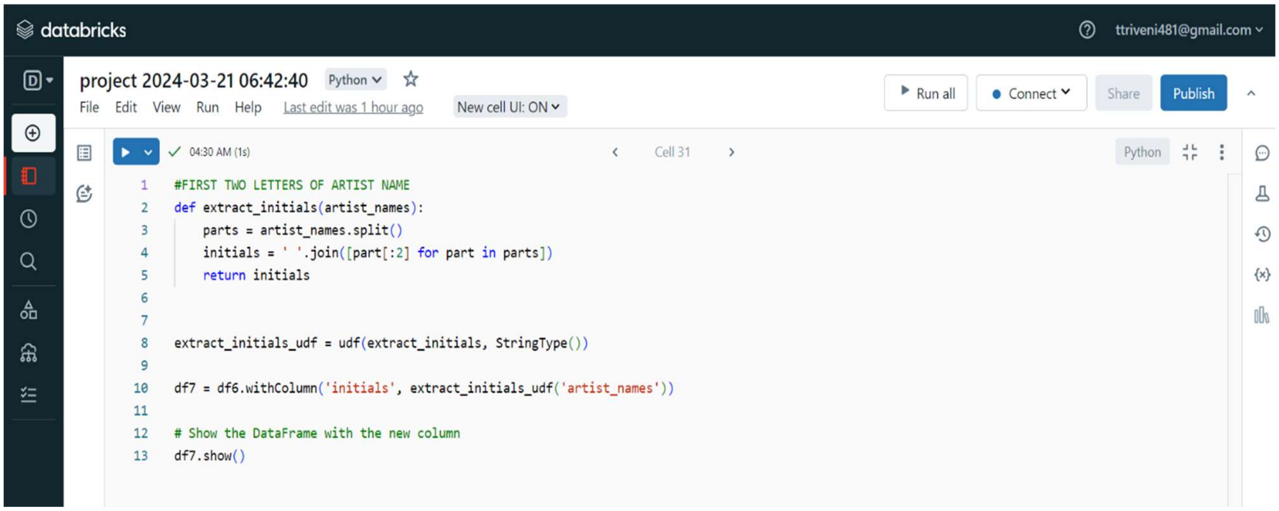
Step: 15

Using the withColumn method in PySpark to add columns in a DataFrame.

1. **‘df.withColumn ("Initials")’:**This creates the "Initials" column in the DataFrame df.



```
+--------------------+------------------+--------------------+---------+------------+---------+-----------+-----------+-----------+--------+
|          track_name|      artist_names|          album_name|streaming|danceability|duration1|bpm_seconds|popular_cat|releasedate|initials|
+--------------------+------------------+--------------------+---------+------------+---------+-----------+-----------+-----------+--------+
|      Exist evening.|   Mallory Camacho|         Than skill.|   437868|          76|2min48sec|       1sec|     Medium|  18-2-2021|   Ma Ca|
|          Every ask.|  Mrs. Laurie Pham|      Paper company.|   831048|          73|5min29sec|       1sec|     Medium|  12-1-1979|Mr La Ph|
|Safe area meeting...|Dr. William Rivera|          Race four.|   708128|          57|9min23sec|       3sec|       High| 12-11-2018|Dr Wi Ri|
|                Pay.|   Mark Richardson|    Likely response.|   334042|          60| 9min4sec|       1sec|     Medium|  12-7-1974|   Ma Ri|
|Rule visit benefi...|Cynthia Andrews MD|                May.|   671021|          42|7min29sec|       1sec|       High|   2-4-1964|Cy An MD|
|Resource remain d...|       Emily Garza|      Leave society.|   728339|          96|1min24sec|       1sec|        Low|  12-7-1970|   Em Ga|
|            Suggest.|       Gary Spears|            Economy.|   733524|          61|2min19sec|       3sec|       High|   1-2-1957|   Ga Sp|
|Since laugh they ...|Jonathon Schroeder|                 To.|   876616|          68|1min20sec|       2sec|     Medium|  31-1-1978|   Jo Sc|
|  Bill front travel.|      Casey Hayden|            Success.|   356554|          70|5min13sec|       1sec|     Medium|  12-8-1982|   Ca Ha|
|      Center charge.|        Emily Hall|Second relationship.|   539550|          45| 7min2sec|       3sec|        Low|   9-5-1960|   Em Ha|
|  Share school vote.|       Casey Evans|               Fear.|   960006|          15|6min47sec|       3sec|        Low|  12-7-1983|   Ca Ev|
|     Kid individual.|     Meghan Snyder|               Card.|   872791|          93| 8min1sec|       2sec|        Low|  12-5-1999|   Me Sn|
|            Produce.|    Pedro Garrison|      Machine offer.|    23981|          72|7min25sec|       1sec|     Medium|  12-6-1985|   Pe Ga|
| First without soon.|    Daniel Freeman|             Reason.|   956947|          42|6min22sec|       1sec|        Low| 19-11-2005|   Da Fr|
|      Meeting voice.|        Laura Hill|      Improve cover.|   920230|          32|9min15sec|       3sec|       High| 13-11-1983|   La Hi|
|Hotel agree Ameri...|      Joseph Davis|               Edge.|   289536|           2|4min26sec|       3sec|        Low|  12-1-1975|   Jo Da|
|         Onto think.|   Andrew Williams|        Red central.|   801990|          40|4min18sec|       1sec|     Medium| 12-10-1983|   An Wi|
| Term federal young.|      Alex Patrick|             Report.|   529541|          17|8min19sec|       2sec|        Low| 12-11-1953|   Al Pa|
|  Art bank director.|      Ryan Edwards|       Head instead.|    37452|          22|8min12sec|       2sec|        Low|  22-9-1998|   Ry Ed|
|      Middle appear.|      Thomas Smith|               Page.|   251792|          67|9min25sec|       1sec|        Low| 11-10-1982|   Th Sm|
```



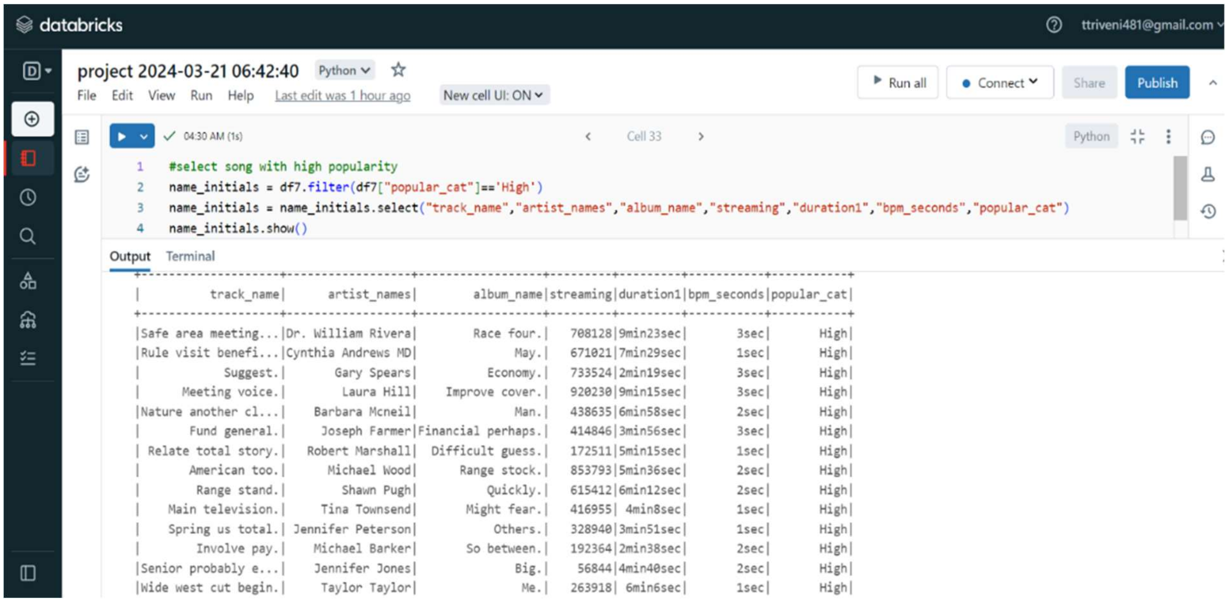
```python
#FIRST TWO LETTERS OF ARTIST NAME
def extract_initials(artist_names):
    parts = artist_names.split()
    initials = ' '.join([part[:2] for part in parts])
    return initials


extract_initials_udf = udf(extract_initials, StringType())

df7 = df6.withColumn('initials', extract_initials_udf('artist_names'))

# Show the DataFrame with the new column
df7.show()
```

Step 16:

Select the song with high popularity and also selecting the particular columns.

1.‘select’: This select function is used to selecting the particular columns from the dataframe.

2. ‘Filter’ : Filter function is used to filtering the column by the given condition.



```
#select song with high popularity
name_initials = df7.filter(df7["popular_cat"]=='High')
name_initials = name_initials.select("track_name","artist_names","album_name","streaming","duration1","bpm_seconds","popular_cat")
name_initials.show()
```

```
+--------------------+------------------+------------------+---------+---------+-----------+-----------+
|          track_name|      artist_names|        album_name|streaming|duration1|bpm_seconds|popular_cat|
+--------------------+------------------+------------------+---------+---------+-----------+-----------+
|Safe area meeting...|Dr. William Rivera|        Race four.|   708128|9min23sec|       3sec|       High|
|Rule visit benefi...|Cynthia Andrews MD|              May.|   671021|7min29sec|       1sec|       High|
|            Suggest.|       Gary Spears|          Economy.|   733524|2min19sec|       3sec|       High|
|      Meeting voice.|        Laura Hill|    Improve cover.|   920230|9min15sec|       3sec|       High|
|Nature another cl...|    Barbara Mcneil|              Man.|   438635|6min58sec|       2sec|       High|
|       Fund general.|     Joseph Farmer|Financial perhaps.|   414846|3min56sec|       3sec|       High|
| Relate total story.|   Robert Marshall|  Difficult guess.|   172511|5min15sec|       1sec|       High|
|       American too.|      Michael Wood|      Range stock.|   853793|5min36sec|       2sec|       High|
|        Range stand.|        Shawn Pugh|          Quickly.|   615412|6min12sec|       2sec|       High|
|    Main television.|     Tina Townsend|       Might fear.|   416955| 4min8sec|       1sec|       High|
|    Spring us total.| Jennifer Peterson|           Others.|   328940|3min51sec|       1sec|       High|
|        Involve pay.|    Michael Barker|       So between.|   192364|2min38sec|       2sec|       High|
|Senior probably e...|    Jennifer Jones|              Big.|    56844|4min40sec|       2sec|       High|
|Wide west cut begin.|     Taylor Taylor|               Me.|   263918| 6min6sec|       1sec|       High|
```

Step: 17

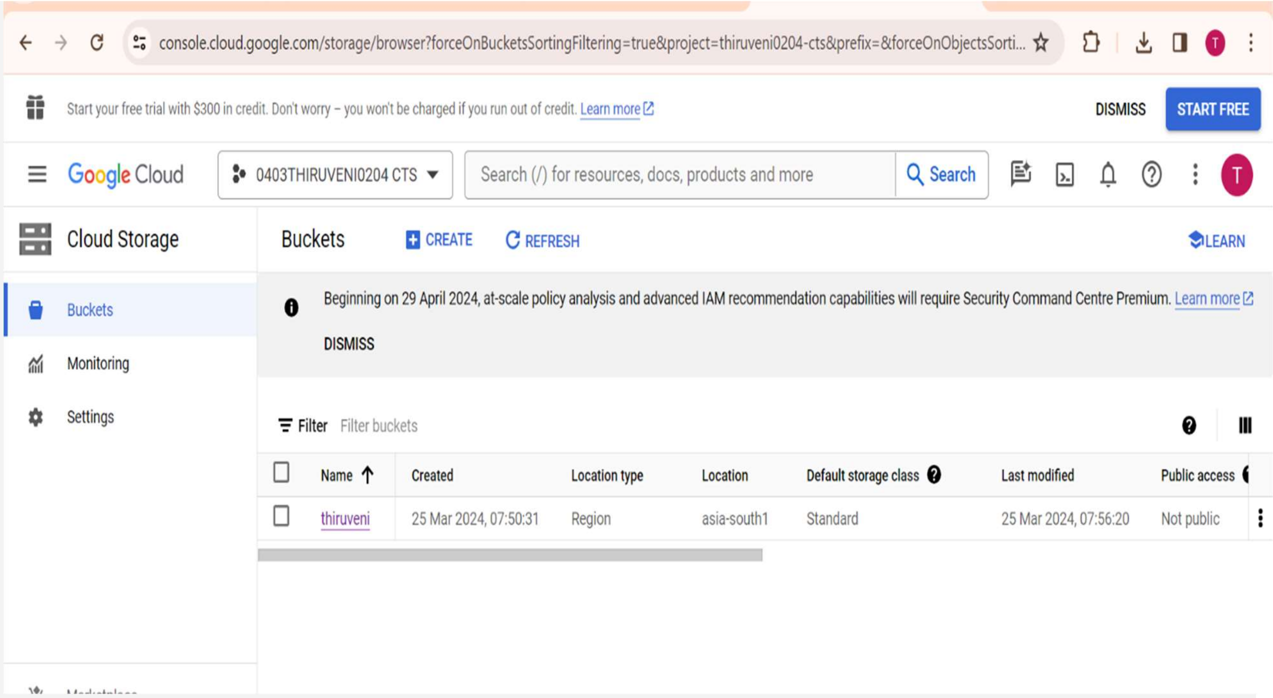
LOAD THE DF TO BUCKET

- Buckets are the basic containers that hold your data within Cloud Storage.
- Everything you store in Cloud Storage must be contained in a bucket.
- Bucket names are publicly visible, so avoid using personally identifiable information (PII) or sensitive data in them.

1.‘Create Bucket in Google Cloud’:

Cloud storage services allows users to create buckets to store their data. Each bucket typically has a unique name within the cloud storage.

- Go to cloud storage – Click Bucket then click Create Bucket and give Bucket name
- Regions and zone (asia-Mumbai)
- Create bucket
- Now the created bucket is displayed in bucket page.


console.cloud.google.com/storage/browser?forceOnBucketsSortingFiltering=true&project=thiruveni0204-cts&prefix=&forceOnObjectsSorti...
Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more
DISMISS
START FREE
Google Cloud
0403THIRUVENI0204 CTS
Search (/) for resources, docs, products and more
Search
Cloud Storage
Buckets
CREATE
REFRESH
LEARN
Buckets
Monitoring
Settings
Filter
Filter buckets

Beginning on 29 April 2024, at-scale policy analysis and advanced IAM recommendation capabilities will require Security Command Centre Premium. Learn more

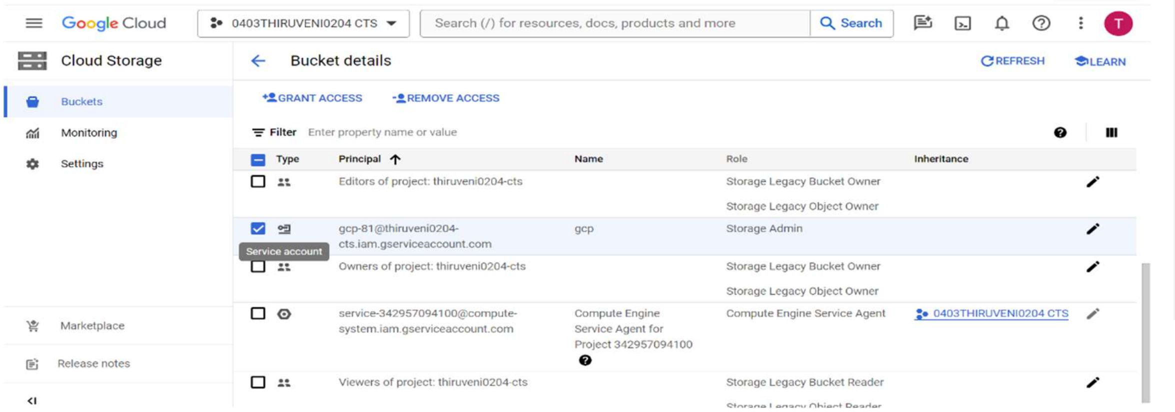
DISMISS

□	Name ↑	Created	Location type	Location	Default storage class	Last modified	Public access
□	thiruveni	25 Mar 2024, 07:50:31	Region	asia-south1	Standard	25 Mar 2024, 07:56:20	Not public

Step 18:

CREATING SERVICE ACCOUNT:

A service account represents google cloud identity such as code running on the compute engine VMs, App Engine running outside Google.

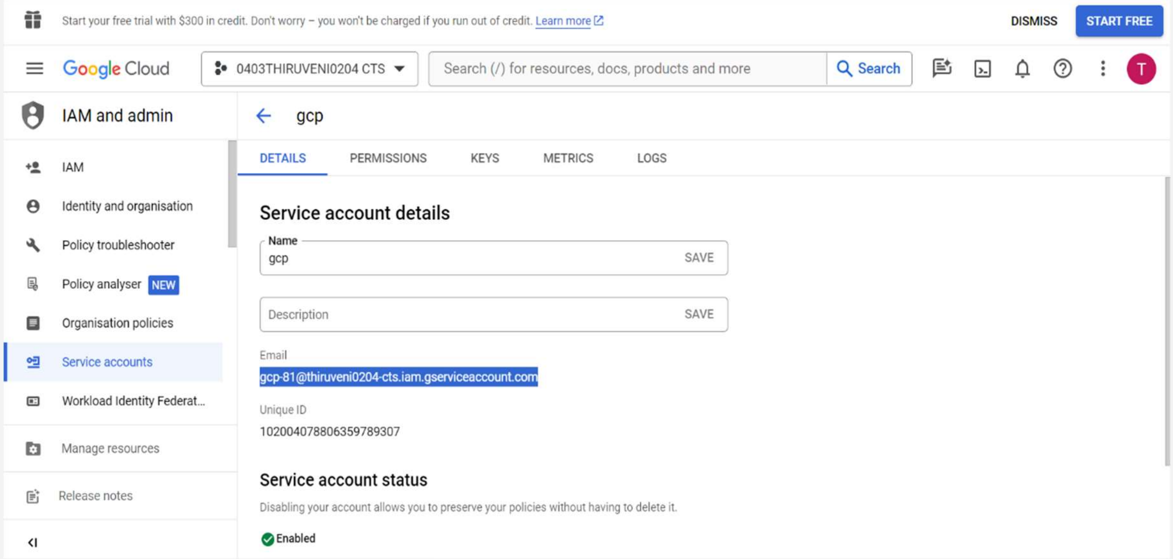


☐	Type	Principal ↑	Name	Role	Inheritance
□		Editors of project: thiruveni0204-cts		Storage Legacy Bucket Owner Storage Legacy Object Owner	
☑		gcp-81@thiruveni0204-cts.iam.gserviceaccount.com	gcp	Storage Admin	
□		Owners of project: thiruveni0204-cts		Storage Legacy Bucket Owner Storage Legacy Object Owner	
□		service-342957094100@compute-system.iam.gserviceaccount.com	Compute Engine Service Agent for Project 342957094100	Compute Engine Service Agent	0403THIRUVENI0204 CTS
□		Viewers of project: thiruveni0204-cts		Storage Legacy Bucket Reader Storage Legacy Object Reader	

Step 19:

COPYING THE SERVICE ACCOUNT DETAILS

Go to service account and copy the email that was generated in the Details tab.



Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more

Service account details

Name: gcp

Description

Email: gcp-81@thiruveni0204-cts.iam.gserviceaccount.com

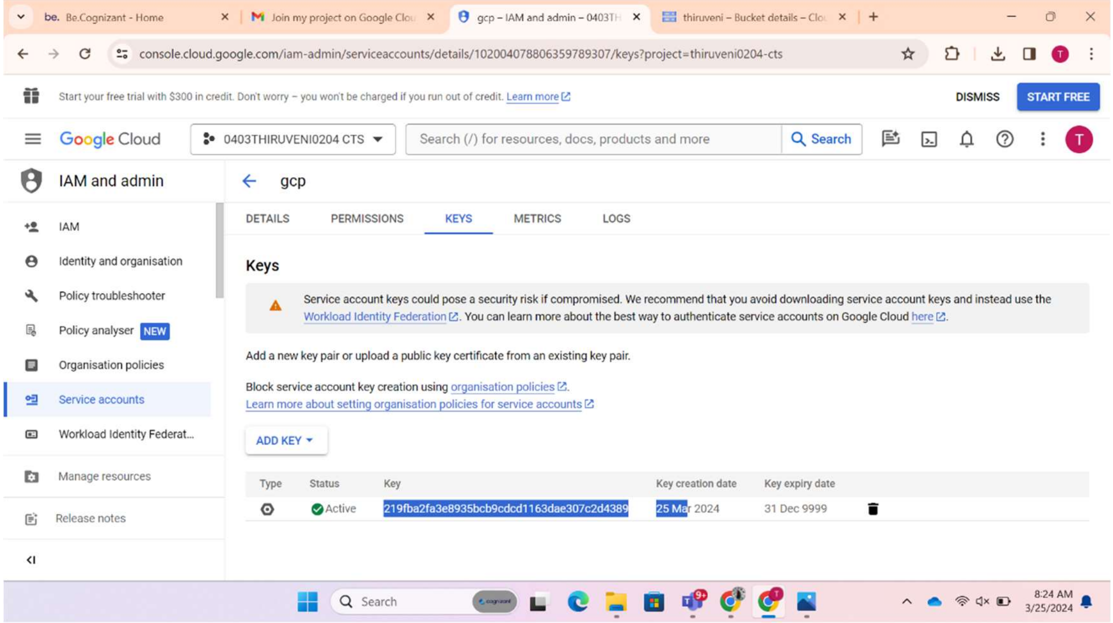
Unique ID: 102004078806359789307

Service account status

Disabling your account allows you to preserve your policies without having to delete it.

✓ Enabled

In key tab we have downlod JSON file that was stored in our local system.



console.cloud.google.com/iam-admin/serviceaccounts/details/102004078806359789307/keys?project=thiruveni0204-cts

Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more

Keys

Service account keys could pose a security risk if compromised. We recommend that you avoid downloading service account keys and instead use the Workload Identity Federation. You can learn more about the best way to authenticate service accounts on Google Cloud here.

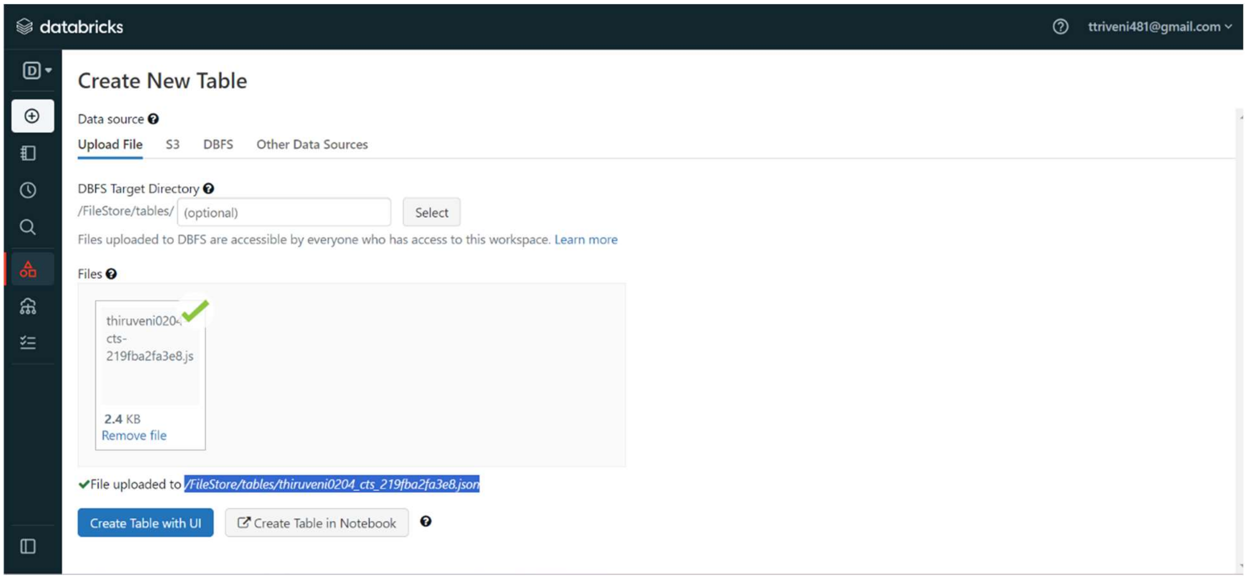
Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organisation policies.
Learn more about setting organisation policies for service accounts

Type	Status	Key	Key creation date	Key expiry date
⚙	✓ Active	219fba2fa3e8935bcb9cdcd1163dae307c2d4389	25 Mar 2024	31 Dec 9999

Step 20:

Upload the key JSON file in the Databricks DBFS.



DBFS Target Directory
/FileStore/tables/ (optional)

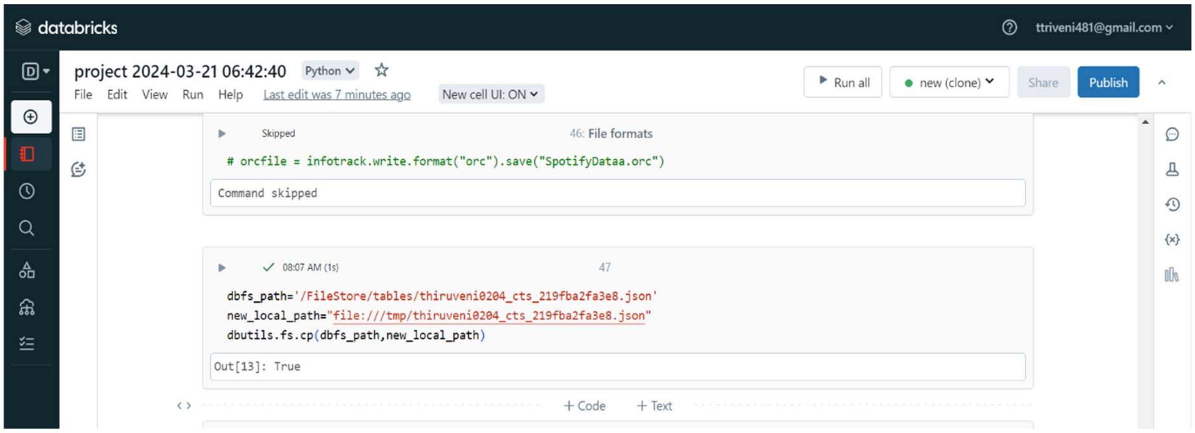
Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

thiruveni0204_cts-219fba2fa3e8.js
2.4 KB

✓File uploaded to /FileStore/tables/thiruveni0204_cts_219fba2fa3e8.json

Create path from local to DBFS by uploading the key Json file in DBFS Databricks.



Skipped 46: File formats

```
# orcfile = infotrack.write.format("orc").save("SpotifyDataa.orc")
```

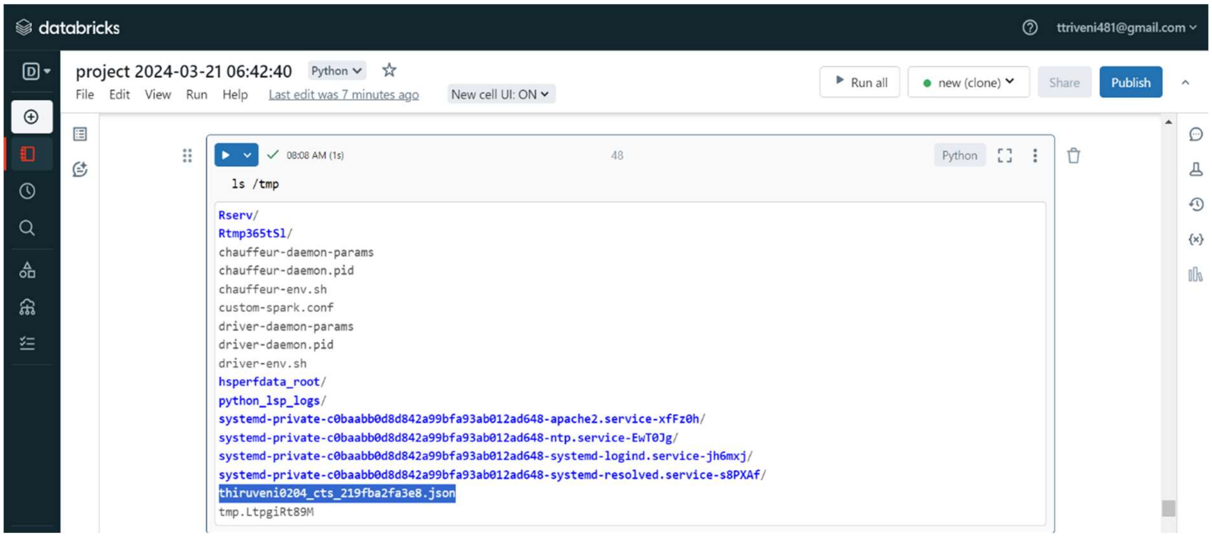
```
Command skipped
```

✓ 08:07 AM (1s) 47

```
dbfs_path='/FileStore/tables/thiruveni0204_cts_219fba2fa3e8.json'
new_local_path="file:///tmp/thiruveni0204_cts_219fba2fa3e8.json"
dbutils.fs.cp(dbfs_path,new_local_path)
```

```
Out[13]: True
```

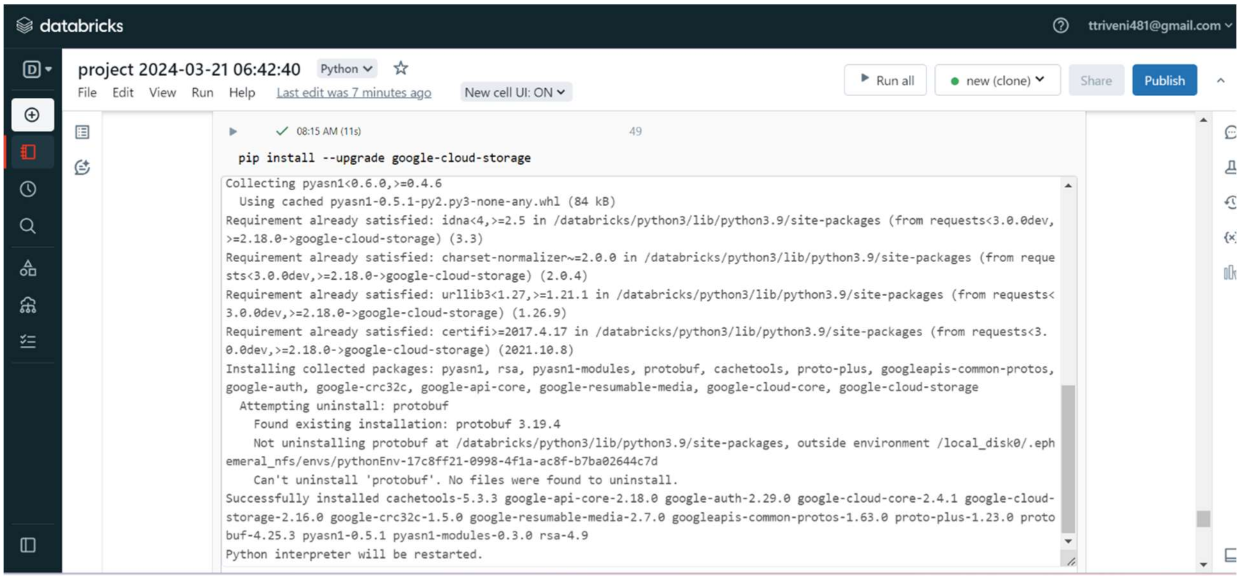
'LS' command is used to list the folders in the DBFS.



```
ls /tmp
```

```
Rserv/
Rtmp365tSl/
chauffeur-daemon-params
chauffeur-daemon.pid
chauffeur-env.sh
custom-spark.conf
driver-daemon-params
driver-daemon.pid
driver-env.sh
hsperfdata_root/
python_lsp_logs/
systemd-private-c0baabb0d8d842a99bfa93ab012ad648-apache2.service-xfFz0h/
systemd-private-c0baabb0d8d842a99bfa93ab012ad648-ntp.service-EwT0Jg/
systemd-private-c0baabb0d8d842a99bfa93ab012ad648-systemd-logind.service-jh6mxj/
systemd-private-c0baabb0d8d842a99bfa93ab012ad648-systemd-resolved.service-s8PXAf/
thiruveni0204_cts_219fba2fa3e8.json
tmp.LtpgiRt89M
```

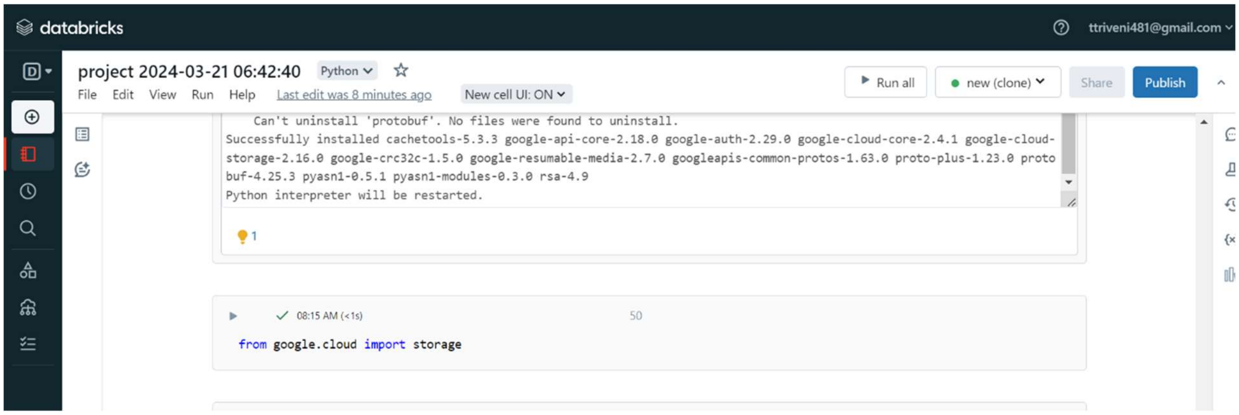
Install the Google Cloud storage in Databricks using pip command.



```
pip install --upgrade google-cloud-storage
```

```
Collecting pyasn1<0.6.0,>=0.4.6
  Using cached pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
Requirement already satisfied: idna<4,>=2.5 in /databricks/python3/lib/python3.9/site-packages (from requests<3.0.0dev,
>=2.18.0->google-cloud-storage) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /databricks/python3/lib/python3.9/site-packages (from reque
sts<3.0.0dev,>=2.18.0->google-cloud-storage) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /databricks/python3/lib/python3.9/site-packages (from requests<
3.0.0dev,>=2.18.0->google-cloud-storage) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in /databricks/python3/lib/python3.9/site-packages (from requests<3.
0.0dev,>=2.18.0->google-cloud-storage) (2021.10.8)
Installing collected packages: pyasn1, rsa, pyasn1-modules, protobuf, cachetools, proto-plus, googleapis-common-protos,
google-auth, google-crc32c, google-api-core, google-resumable-media, google-cloud-core, google-cloud-storage
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.19.4
    Not uninstalling protobuf at /databricks/python3/lib/python3.9/site-packages, outside environment /local_disk0/.eph
emeral_nfs/envs/pythonEnv-17c8ff21-0998-4f1a-ac8f-b7ba02644c7d
    Can't uninstall 'protobuf'. No files were found to uninstall.
Successfully installed cachetools-5.3.3 google-api-core-2.18.0 google-auth-2.29.0 google-cloud-core-2.4.1 google-cloud-
storage-2.16.0 google-crc32c-1.5.0 google-resumable-media-2.7.0 googleapis-common-protos-1.63.0 proto-plus-1.23.0 proto
buf-4.25.3 pyasn1-0.5.1 pyasn1-modules-0.3.0 rsa-4.9
Python interpreter will be restarted.
```

‘Import google cloud’ package in the Databricks.

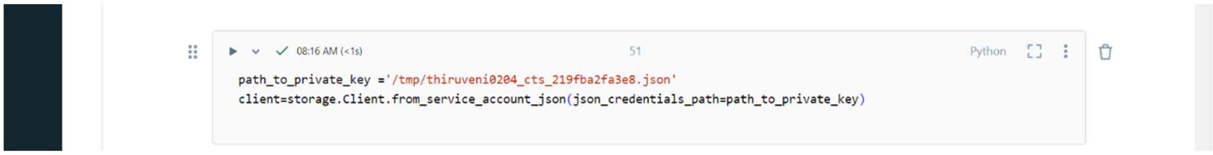


```
    Can't uninstall 'protobuf'. No files were found to uninstall.
Successfully installed cachetools-5.3.3 google-api-core-2.18.0 google-auth-2.29.0 google-cloud-core-2.4.1 google-cloud-
storage-2.16.0 google-crc32c-1.5.0 google-resumable-media-2.7.0 googleapis-common-protos-1.63.0 proto-plus-1.23.0 proto
buf-4.25.3 pyasn1-0.5.1 pyasn1-modules-0.3.0 rsa-4.9
Python interpreter will be restarted.
```

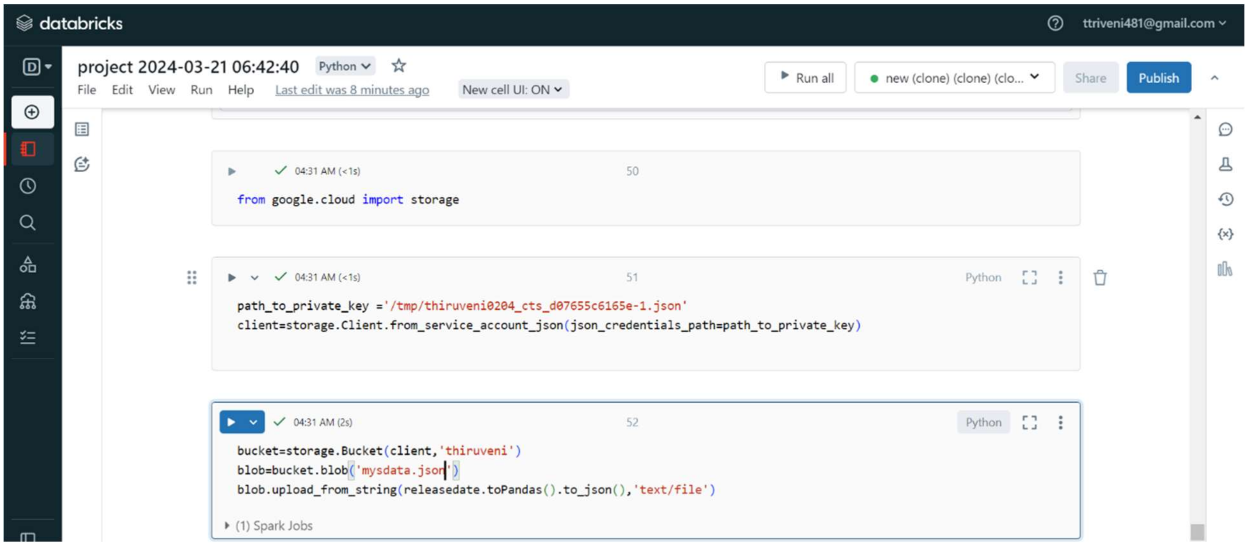
```
from google.cloud import storage
```

SERIALIZATION:

Loading my DBFS file to Google Cloud Bucket as **JSON file format**.



```
path_to_private_key ='/tmp/thiruveni0204_cts_219fba2fa3e8.json'
client=storage.Client.from_service_account_json(json_credentials_path=path_to_private_key)
```



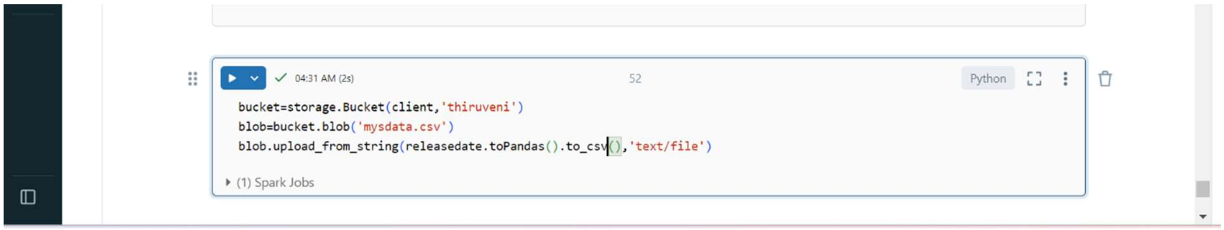
```
from google.cloud import storage
```

```
path_to_private_key ='/tmp/thiruveni0204_cts_d07655c6165e-1.json'
client=storage.Client.from_service_account_json(json_credentials_path=path_to_private_key)
```

```
bucket=storage.Bucket(client,'thiruveni')
blob=bucket.blob('mysdata.json')
blob.upload_from_string(releasedate.toPandas().to_json(),'text/file')
```

(1) Spark Jobs

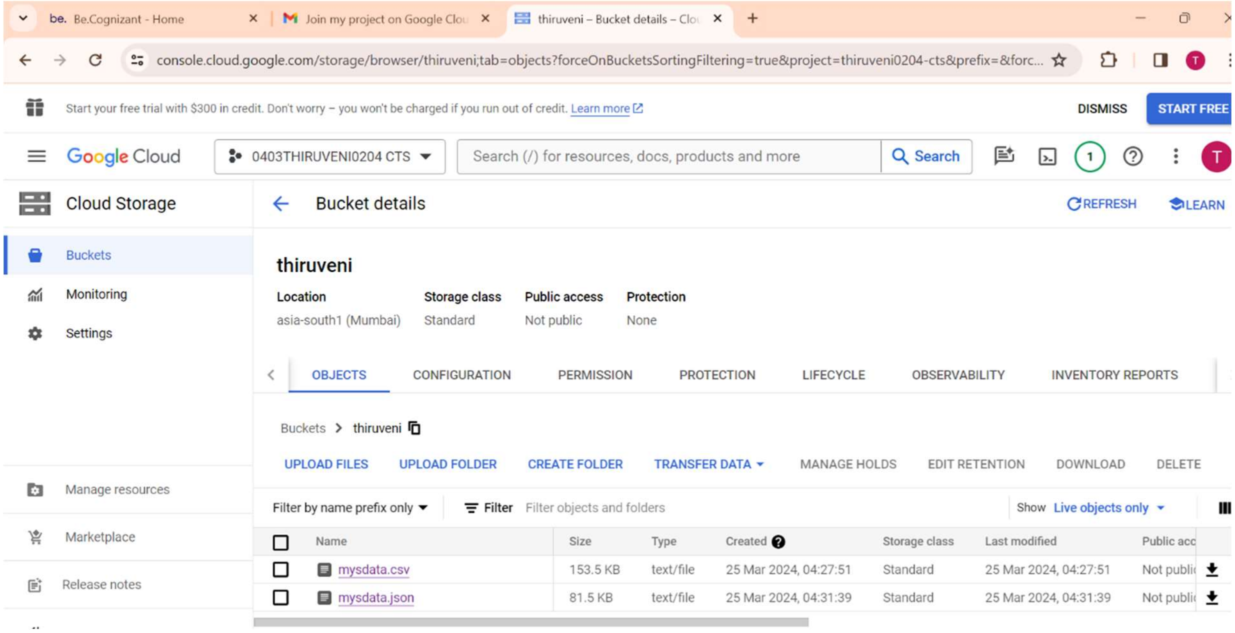
Loading my DBFS file to Google Cloud Bucket as **csv file format**.



```
bucket=storage.Bucket(client,'thiruveni')
blob=bucket.blob('mysdata.csv')
blob.upload_from_string(releasedate.toPandas().to_csv(),'text/file')
```

▸ (1) Spark Jobs

Now JSON and CSV files are uploaded in the Google Cloud Bucket.

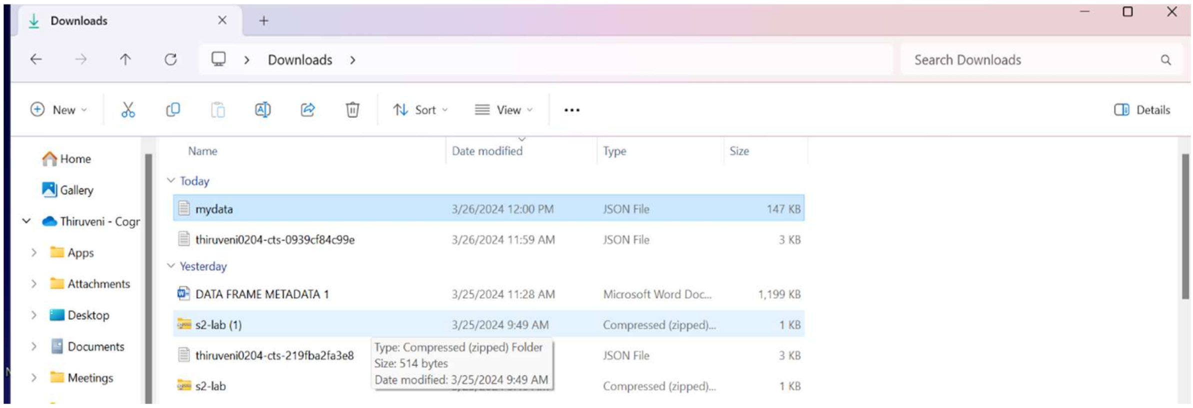

Be.Cognizant - Home
Join my project on Google Clou
thiruveni – Bucket details – Clo
console.cloud.google.com/storage/browser/thiruveni;tab=objects?forceOnBucketsSortingFiltering=true&project=thiruveni0204-cts&prefix=&forc...
Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more
DISMISS
START FREE
Google Cloud
0403THIRUVENI0204 CTS
Search (/) for resources, docs, products and more
Search
Cloud Storage
Buckets
Monitoring
Settings
Manage resources
Marketplace
Release notes
Bucket details
REFRESH
LEARN
OBJECTS
CONFIGURATION
PERMISSION
PROTECTION
LIFECYCLE
OBSERVABILITY
INVENTORY REPORTS
Buckets > thiruveni
UPLOAD FILES
UPLOAD FOLDER
CREATE FOLDER
TRANSFER DATA
MANAGE HOLDS
EDIT RETENTION
DOWNLOAD
DELETE
Filter by name prefix only
Filter
Filter objects and folders
Show Live objects only

thiruveni

Location	Storage class	Public access	Protection
asia-south1 (Mumbai)	Standard	Not public	None

Name	Size	Type	Created	Storage class	Last modified	Public acc
mysdata.csv	153.5 KB	text/file	25 Mar 2024, 04:27:51	Standard	25 Mar 2024, 04:27:51	Not public
mysdata.json	81.5 KB	text/file	25 Mar 2024, 04:31:39	Standard	25 Mar 2024, 04:31:39	Not public

Download the mydata.json from google cloud bucket and stored in the local file system.



Name	Date modified	Type	Size
Today			
mydata	3/26/2024 12:00 PM	JSON File	147 KB
thiruveni0204-cts-0939cf84c99e	3/26/2024 11:59 AM	JSON File	3 KB
Yesterday			
DATA FRAME METADATA 1	3/25/2024 11:28 AM	Microsoft Word Doc...	1,199 KB
s2-lab (1)	3/25/2024 9:49 AM	Compressed (zipped)...	1 KB
thiruveni0204-cts-219fba2fa3e8		JSON File	3 KB
s2-lab		Compressed (zipped)...	1 KB

CONCLUSION:

In conclusion, streaming serialization in Apache Spark on Databricks offers a powerful solution for real-time data processing and analysis. By leveraging Spark's streaming capabilities and Databricks' cloud-based platform, organizations can efficiently ingest, process, and analyze data streams in a scalable and fault-tolerant manner. Streaming serialization enables the seamless conversion of streaming data into a structured format, facilitating easy integration with Spark's DataFrame and Dataset APIs for further analysis and manipulation. This process ensures that data is efficiently serialized and deserialized during streaming operations, optimizing performance and reducing latency.