DATA CLEANING: CLEANSE DATA BY REMOVING NOISE, HANDLING MISSING VALUES, OR CORRECTING ERROS USING EXTRACT-TRANSFORM-LOAD(ETL)

A Project Report Submitted in the fulfilment of the requirements for Interim-Project Evaluation

SUBMITTED BY:
NAME: VAISHNAVI R
EMP ID: 2319882
COHORT CODE: CSDAIA24GP002

ABSTRACT

In today's data-driven world, the integrity and quality of data are paramount for making informed decisions. However, raw data often contains noise, missing values, and errors that can significantly impact the accuracy of analyses and models. In this project, we employ the Extract, Transform, Load (ETL) process using Apache Spark and DataFrames to cleanse and preprocess the data.

The ETL process begins with extracting raw data from various sources, such as databases, CSV files, or streaming sources. Once extracted, the data undergoes transformation to remove noise, handle missing values, and correct errors. This transformation phase involves techniques such as data imputation, outlier detection, and standardization to ensure data consistency and accuracy.

Apache Spark provides a powerful platform for scalable data processing, and DataFrames offer a high-level abstraction for manipulating structured data. Leveraging Spark's distributed computing capabilities, we can efficiently cleanse large volumes of data in parallel, making the process scalable and efficient.

The final step of the ETL process is loading the cleansed data into a target destination, such as a data warehouse or a machine learning pipeline, for further analysis or modeling. By cleansing the data using Spark and DataFrames, we ensure that downstream analyses and models are built on reliable and accurate data, leading to more robust insights and decision-making.

Overall, this project demonstrates the importance of data cleansing in ensuring data quality and reliability, and showcases the effectiveness of using Apache Spark and DataFrames for scalable and efficient ETL operations in a big data environment.

TABLE OF CONTENTS

S.NO	CONTENT	PAGE NO
1	INTRODUCTION	4
2	OBJECTIVE	4
3	METHODOLOGY	5
4	SOFTWARE REQUIREMENT	6
5	PROJECT STATEMENT	6
6	PROJECT REQUIREMENT	7
7	IMPLEMENTATION	8
8	EXPERIMENTAL ANAYSIS	9
9	CONCLUSION	33

1. INTRODUCTION:

In this project, “data cleaning”, we used spark to perform some operations. Apache Spark is an open source analytics engine used for big data workloads. It can handle both batches as well as real-time analytics and data processing workloads. Apache Spark started in 2009 as a research project at the University of California, Berkeley. Researchers were looking for a way to speed up processing jobs in Hadoop systems. It is based on Hadoop Map Reduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. Spark provides native bindings for the Java, Scala, Python, and R programming languages. In addition, it includes several libraries to support build applications for machine learning [MLlib], stream processing [Spark Streaming], and graph processing [GraphX]. Apache Spark consists of Spark Core and a set of libraries. Spark Core is the heart of Apache Spark and it is responsible for providing distributed task transmission, scheduling, and I/O functionality. The Spark Core engine uses the concept of a Resilient Distributed Dataset (RDD) as its basic data type. The RDD is designed so it will hide most of the computational complexity from its users. Spark is intelligent on the way it operates on data; data and partitions are aggregated across a server cluster, where it can then be computed and either moved to a different data store or run through an analytic model. You will not be asked to specify the destination of the files or the computational resources that need to be used in order to store or retrieve files.

2. OBJECTIVE:

- **Noise Removal**: We identify and filter out noisy data points in real time. Noise can arise from sensor glitches or transmission errors. By applying statistical techniques and domain-specific rules, we reduce noise and enhance data quality.

- **Handling Missing Values**: Streaming data often contains missing values (nulls or NaNs). Our project imputes missing values using techniques such as mean imputation, forward/backward filling, or machine learning-based predictions. This ensures that downstream analytics receive complete data.
- **Error Correction**: Data errors (e.g., typos, incorrect units, or inconsistent formats) can propagate downstream. We implement custom validation rules and transformations to correct errors on the fly. For example, converting temperature units from Fahrenheit to Celsius or fixing timestamp discrepancies.

3. METHODOLOGY:

COLLECTION OF DATASET:

The project uses a sample dataset of health data of patients, including columns such as timestamp, patient id, gender, height, weight and other relevant metrics. The dataset is loaded into PySpark DataFrame for processing.

The dataset used in this project is a CSV file named **health_analysis.csv** containing seven fields:

Timestamp	Patient id	Gender	Height	Weight	Heart Rate

DATA PREPROCESSING:

Necessary data cleansing and other transformations are applied to extract relevant metrics and prepare the data health analysis. This may include filtering out irrelevant data, handling missing values, and converting data types .

- **Handling Missing Values:**
  Columns with missing values were present in the dataset and they were removed using PySpark DataFrame transformations.

- **Removing Duplicates:**
  Duplicate rows were present in the dataset and hence it has also been removed using PySpark DataFrame transformations.

SELECTION OF ATTRIBUTES:

Required fields from the dataset are selected and noise values are removed. The fields containing null values are removed using various dataframe transformations. Fields containing missing values are being selected by the user and then cleaned for the health analysis and visualization of data.

HEALTH ANALYSIS AND VISUALIZATION:

1. The CSV file was loaded into PySpark and processed using PySpark DataFrame transformations.
2. Columns containing noise, null and missing values were removed .
3. The cleaned health analysis data is loaded into a new CSV file – 'health_analysis_cleaned'.
4. The health analysis data is converted into pandas DataFrame for plotting using Matplotlib and seaborn.

4. SOFTWARE REQUIREMENT:

DATABRICKS:

Databricks is a unified, open analytics platform for building, deploying, sharing, and maintaining enterprise-grade data, analytics, and AI solutions at scale. The Databricks Data Intelligence Platform integrates with cloud storage and security in your cloud account, and manages and deploys cloud infrastructure on your behalf. Databricks uses generative AI with the data lakehouse to understand the unique semantics of your data. Then, it automatically optimizes performance and manages infrastructure to match your business needs.

5. PROJECT STATEMENT:

The objective of this project is to analyze health data of patients stored in CSV file using PySpark, focusing on various data manipulation operations such as reading the data filtering the data by gender, removing null values in gender column and filling missing values in heart_rate column by finding the mean of heart_rate column and performing dataframe transformations with the height, weight, heart_rate and other columns.

6. PROJECT REQUIREMENT:

STEPS:

- **Read Data from CSV File:** Implement code to read health data of patients from CSV file into PySpark data frames.
- **Drop null values using subset:** Remove null values in heart rate column using subset syntax.
- **Filter by Gender:** Analyze and separate the gender column for 'male and female' separately and 'other' gender separately.
- **Drop null values in Gender(male and female):** In gender column, for 'male' and 'female' convert the other gender to null values and drop it.
- **Drop null values in Gender(other):** In gender column, for 'other' convert the 'male' and 'female' values to null values and drop it.
- **Drop entire row using how:** Using how='any' function drops rows with atleast one null value and how='all' function drops rows with atleast one non null value.
- **Fill missing heart rate values:** Using fillna command we fill the missing heart rate values with mean values of heart rate column.
- **Drop column using threshold value:** Any column in the table has to have atleast the number of non null values given by threshold, otherwise it will get dropped.
- **Filter duplicates:** Using drop duplicates command filter rows in all columns.

DELIVERABLES:

- Python code implementing the above tasks using PySpark and DataFrame in DataBricks.
- Documentation detailing the abstract and need of Apache Spark in the project.
- Implementation screenshots summarizing the findings and insights from the data analysis.

CONSTRAINTS:

- Ensure compatibility with PySpark environment.
- Maintain data integrity throughout the manipulation process.
- Optimize code for performance when dealing with large datasets.

TECHNOLOGY USED AND DATA RESOURCES:

- Data Bricks
- PySpark with Python
- PySpark SQL Functions – DataFrame
- Health dataset of patients as health_analysis.csv

7. IMPLEMENTATION:

DATASET CREATION:

A dataset named health analysis has been created using the python and using its other libraries like faker, random and csv. Using these libraries a csv file is been created. Duplicate rows are being introduced at the time of creation of dataset**.**

DATA LOADING:

The health analysis dataset, containing columns like timestamp, patient id, gender, height, weight and other relevant metrics, is loaded into PySpark DataFrame. The dataset may be stored in a CSV or other suitable format.

DATA PREPROCESSING:

Data preprocessing steps are applied to the DataFrame to clean and transform the data. This may include removing duplicates, handling missing values, and converting data types to ensure compatibility .

VISUALIZATION:

The generated transformations are visualized using Matplotlib, providing a visual representation of the distribution patterns in the health analysis data. They are displayed within the Databricks for easy access and interpretation.

8. EXPERIMENTAL ANALYSIS:

DATASET GENERATOR CODE:

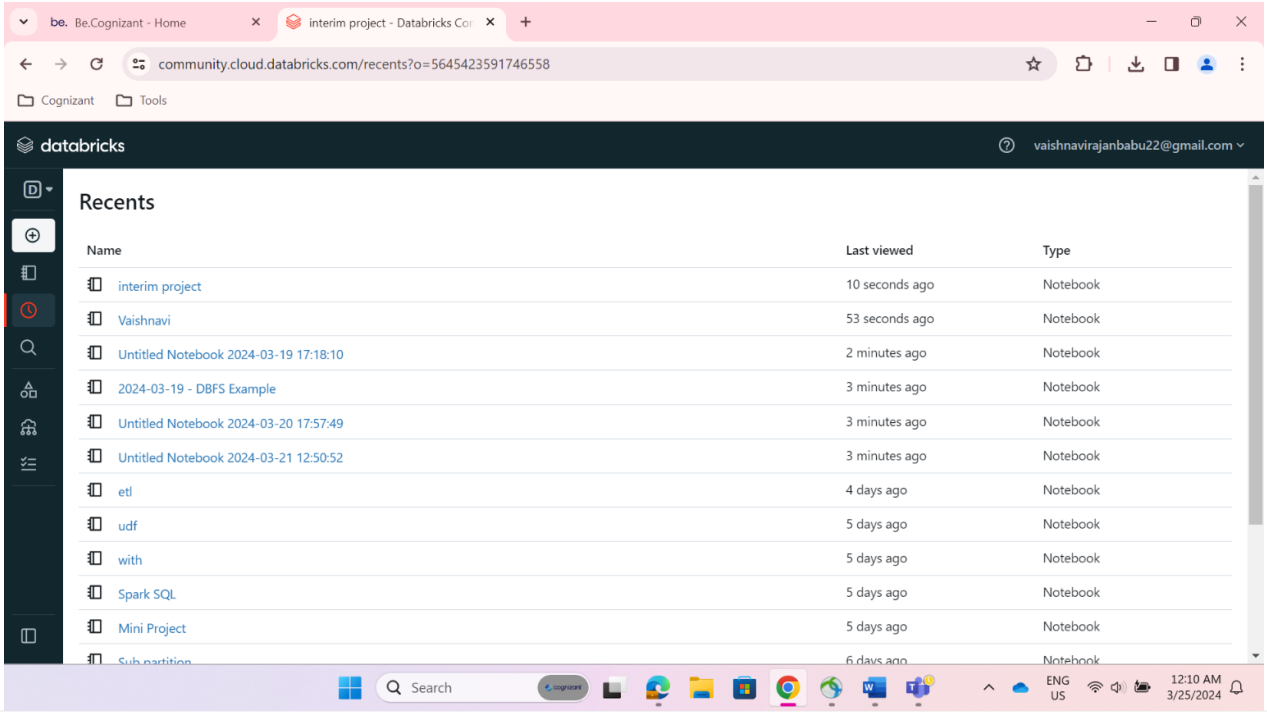
```
%pip install faker

import random

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, lit, rand, when

from faker import Faker


# Initialize Spark session

spark = SparkSession.builder.appName("HealthAnalysis").getOrCreate()


# Initialize Faker for generating random data

fake = Faker()


# Generate 1000 rows of data

data = []

for _ in range(1000):

    timestamp = fake.date_time_this_year()

    patient_id = fake.unique.random_number(digits=6)

    gender = random.choice(["Male", "Female", "Other"])

    weight = round(random.uniform(40, 120), 2)

    height = round(random.uniform(140, 200), 2)
```

```
    heart_rate = random.choice([random.randint(50, 100), None])

    data.append((timestamp, patient_id, gender, weight, height, heart_rate))


# Add duplicate rows

duplicate_rows = random.choices(data, k=50)
```

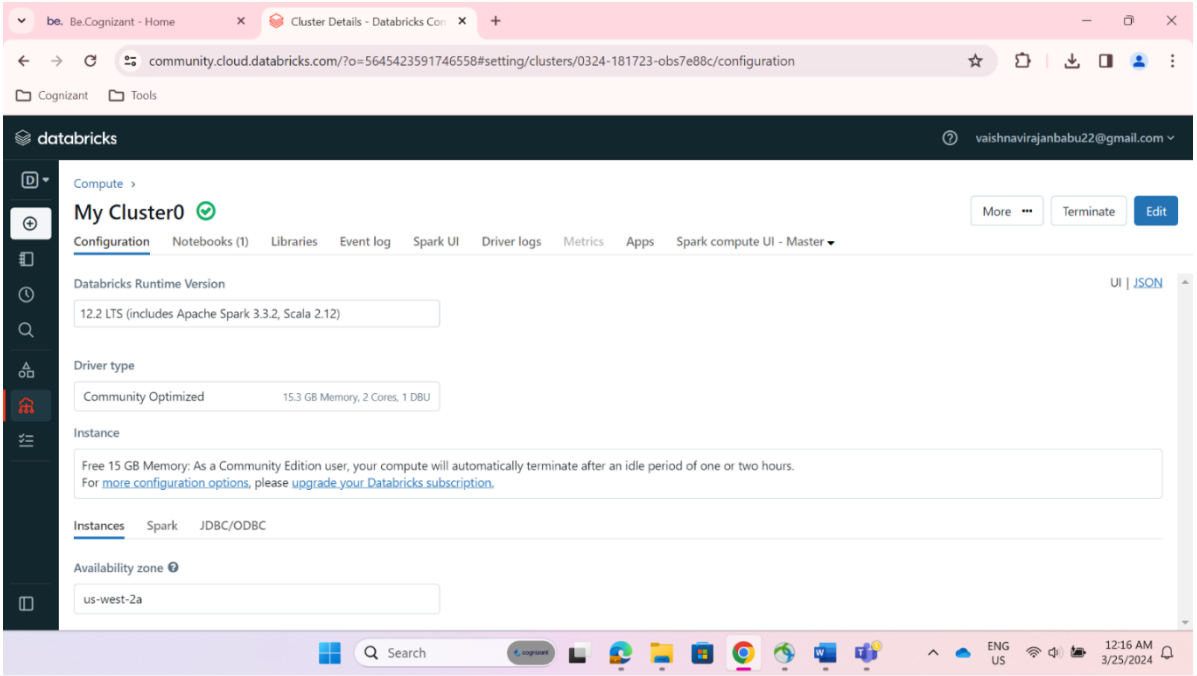
```
data.extend(duplicate_rows)
```

IMPLEMENTATION SCREENSHOTS:

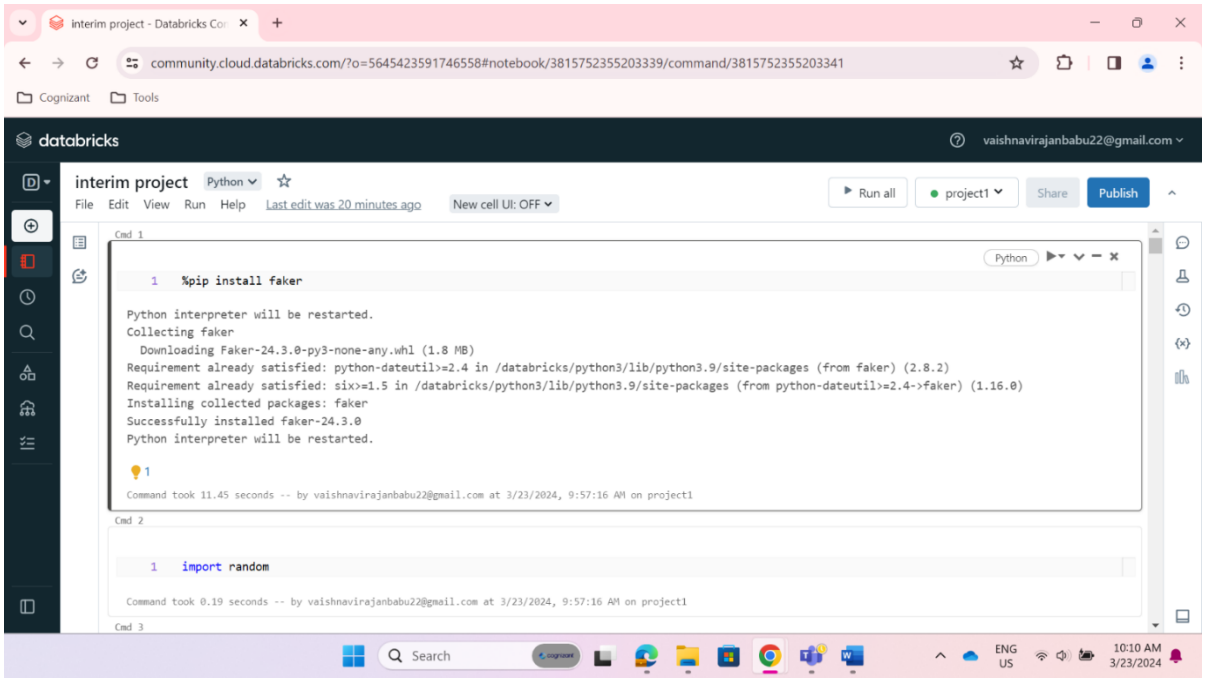
STEP 1: Setup Data Bricks Community edition to your profile and create a new notebook as interim project.



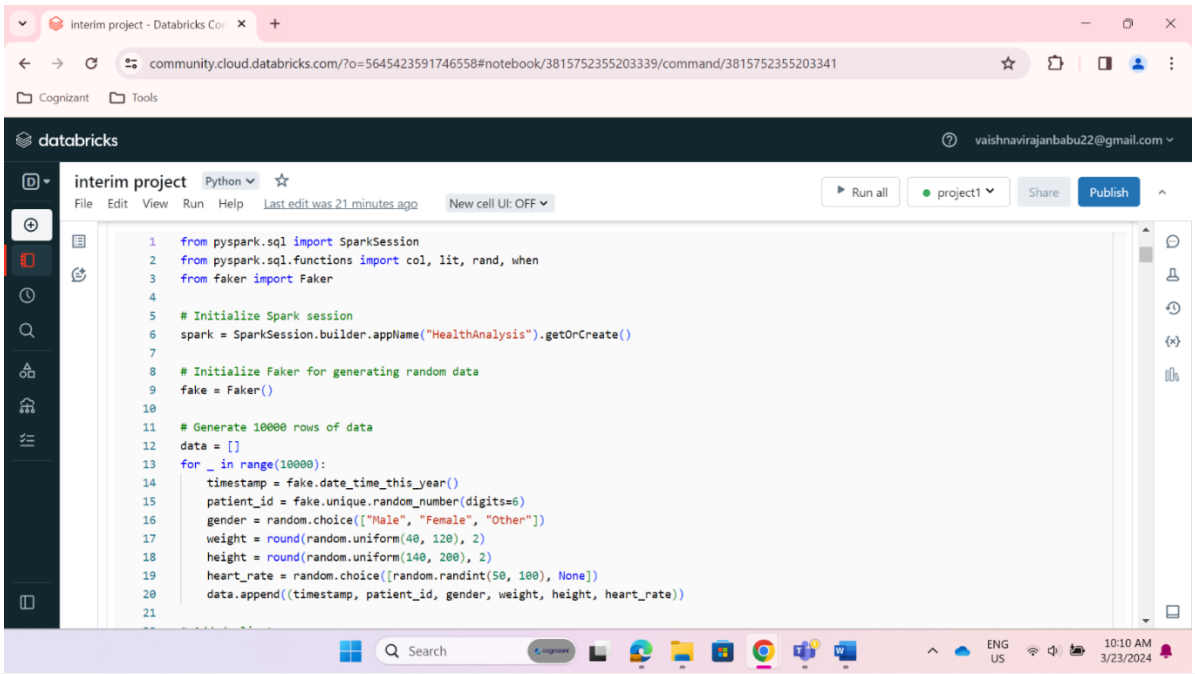
STEP 2: Create and configure resource cluster, for your notebook. Use DataBricks default configuration for your cluster.



STEP 3: Installing and importing required libraries.

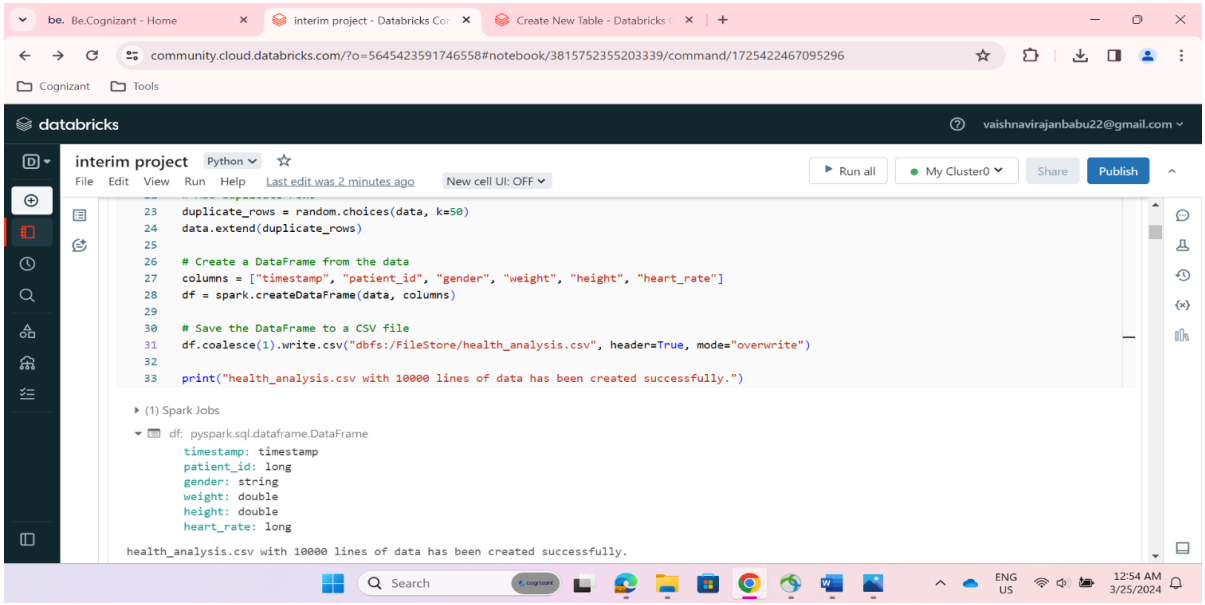


STEP 4: Import pyspark, and all pyspark.sql.functions and create a SparkSession.

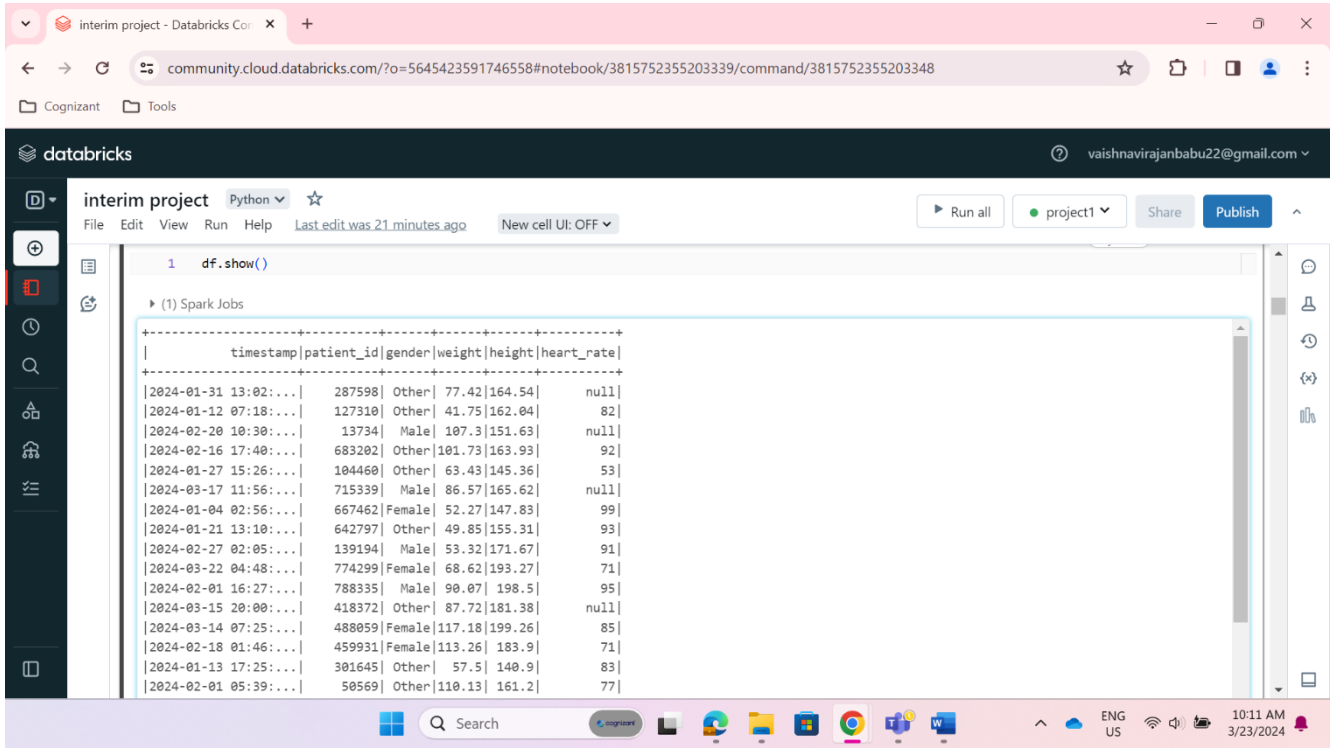


STEP 5: DATA INGESTION**:** Upload health_analysis.csv into dbfs of Data Bricks and import the file into your notebook, by using the dbfs path. Display imported file.

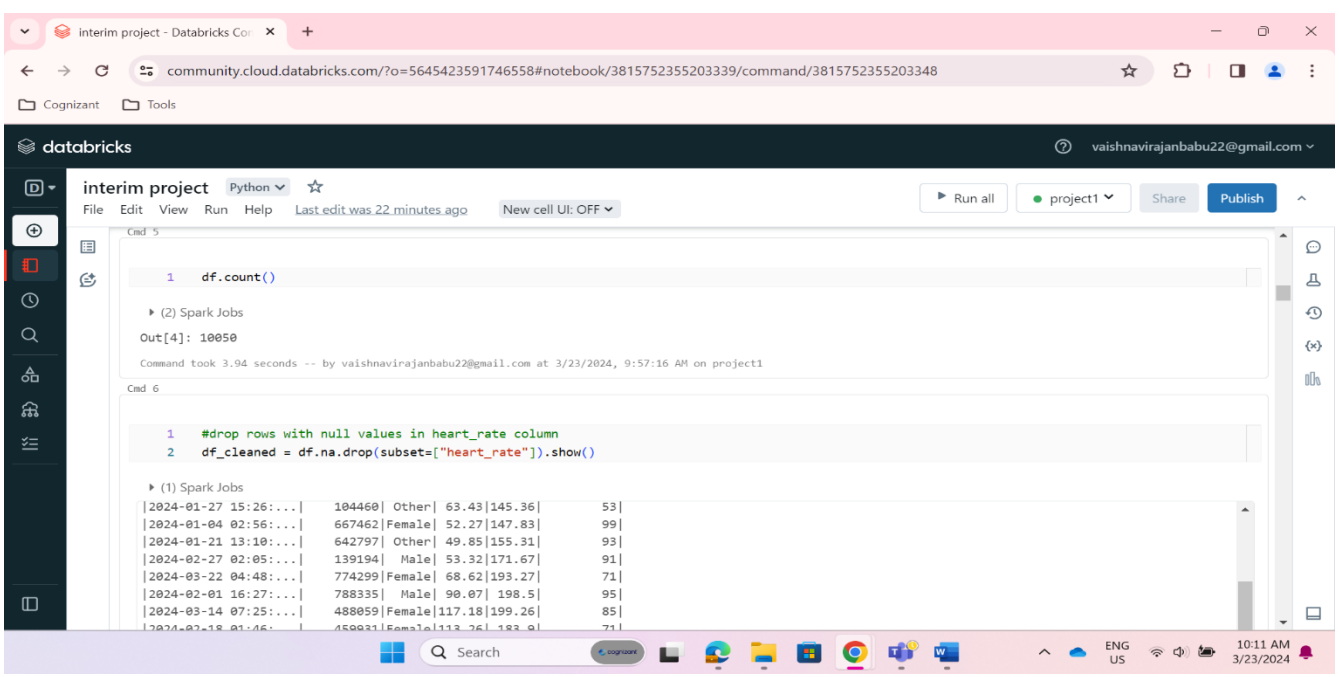
Path: dbfs:FileStore/health_analysis.csv



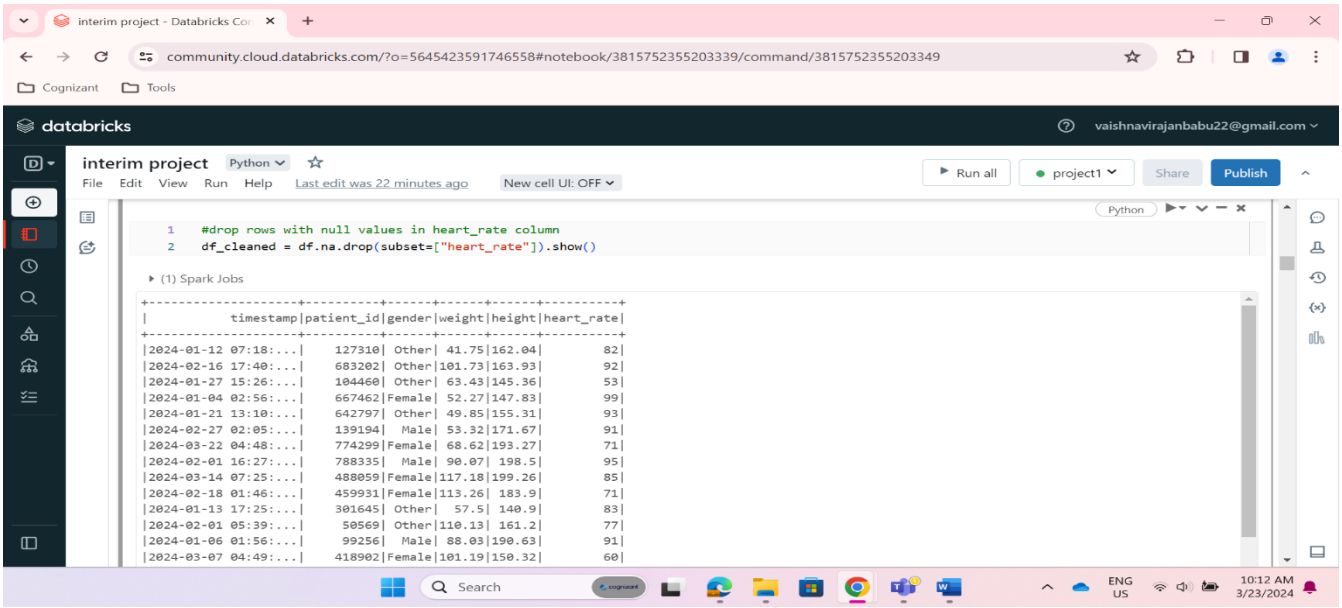
STEP 6: Showing the loaded input dataset using show() function.



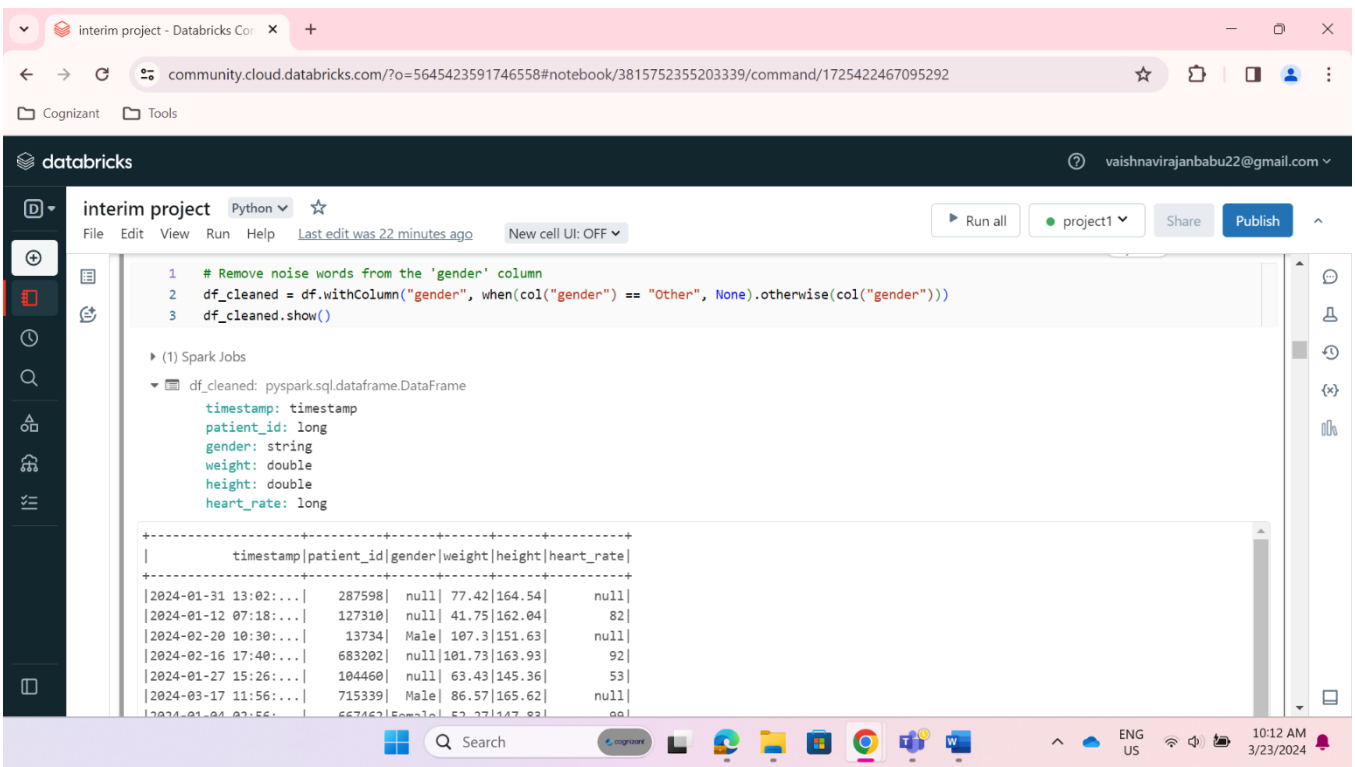
STEP 7: Count function is used to count the number of rows in the dataset.



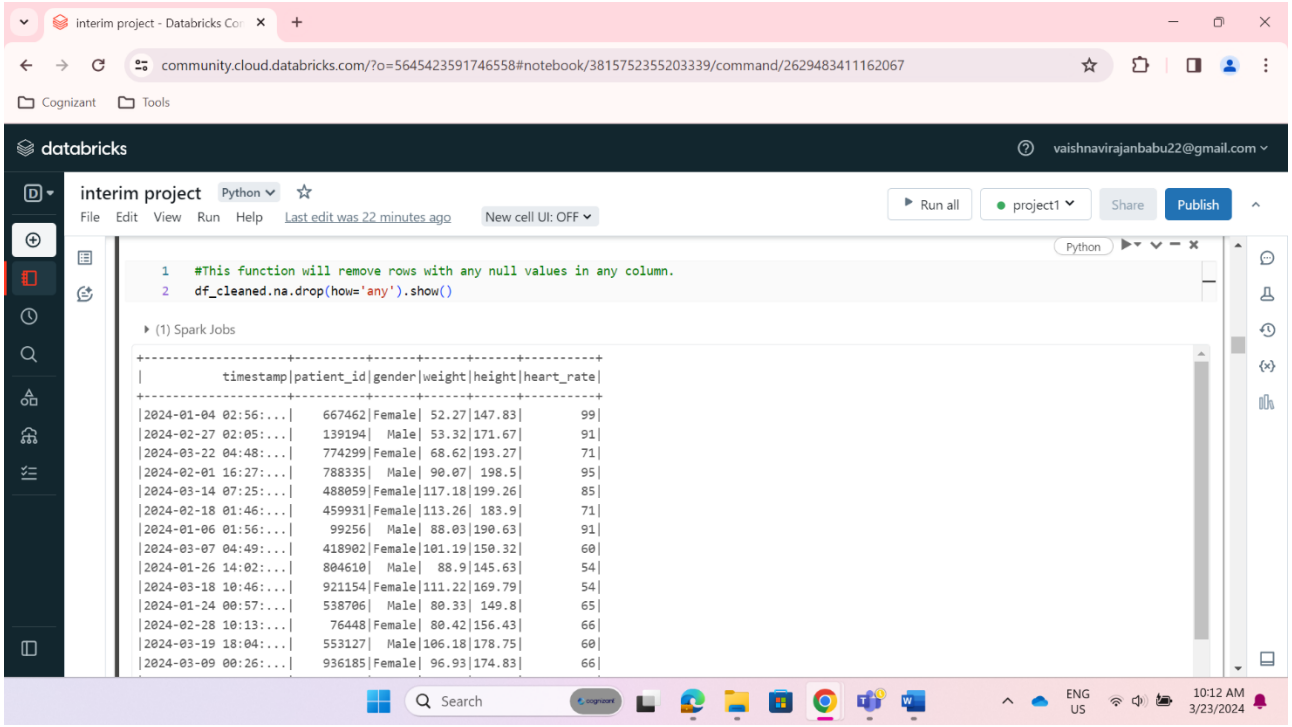
STEP 8: Removing null values from heart rate column.



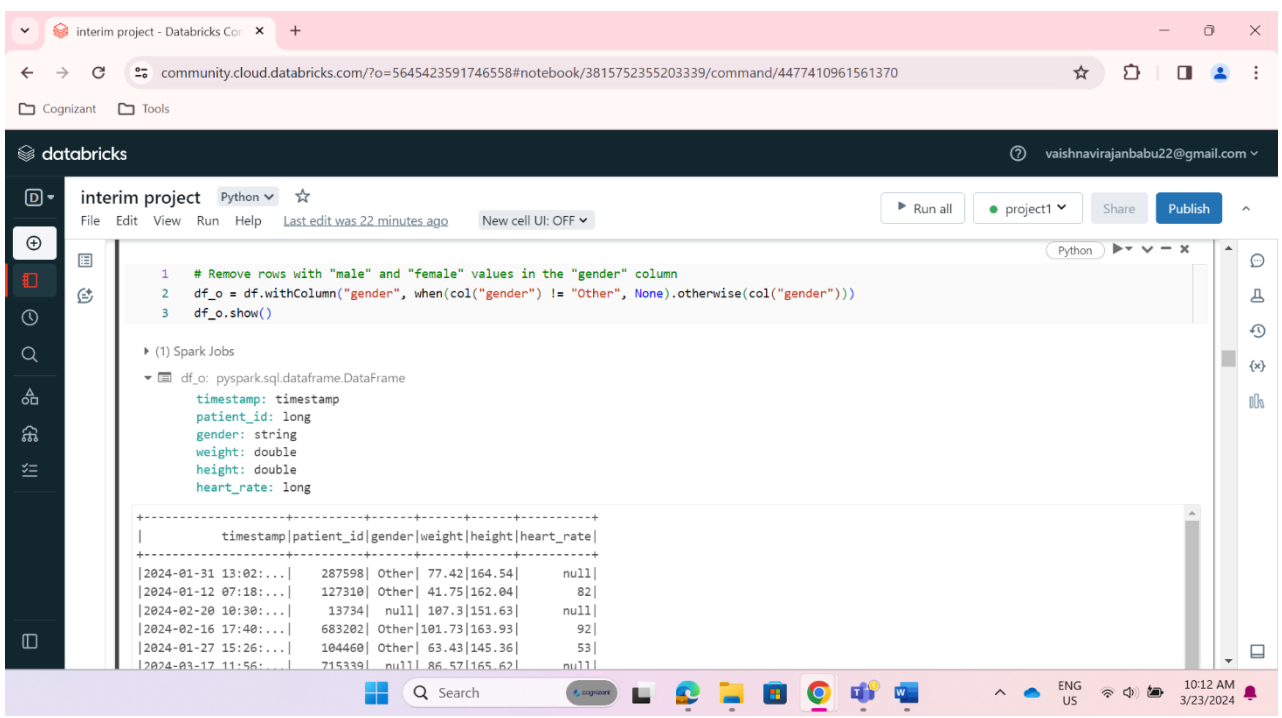
STEP 9: Remove noise words from gender column using withcolumn function.



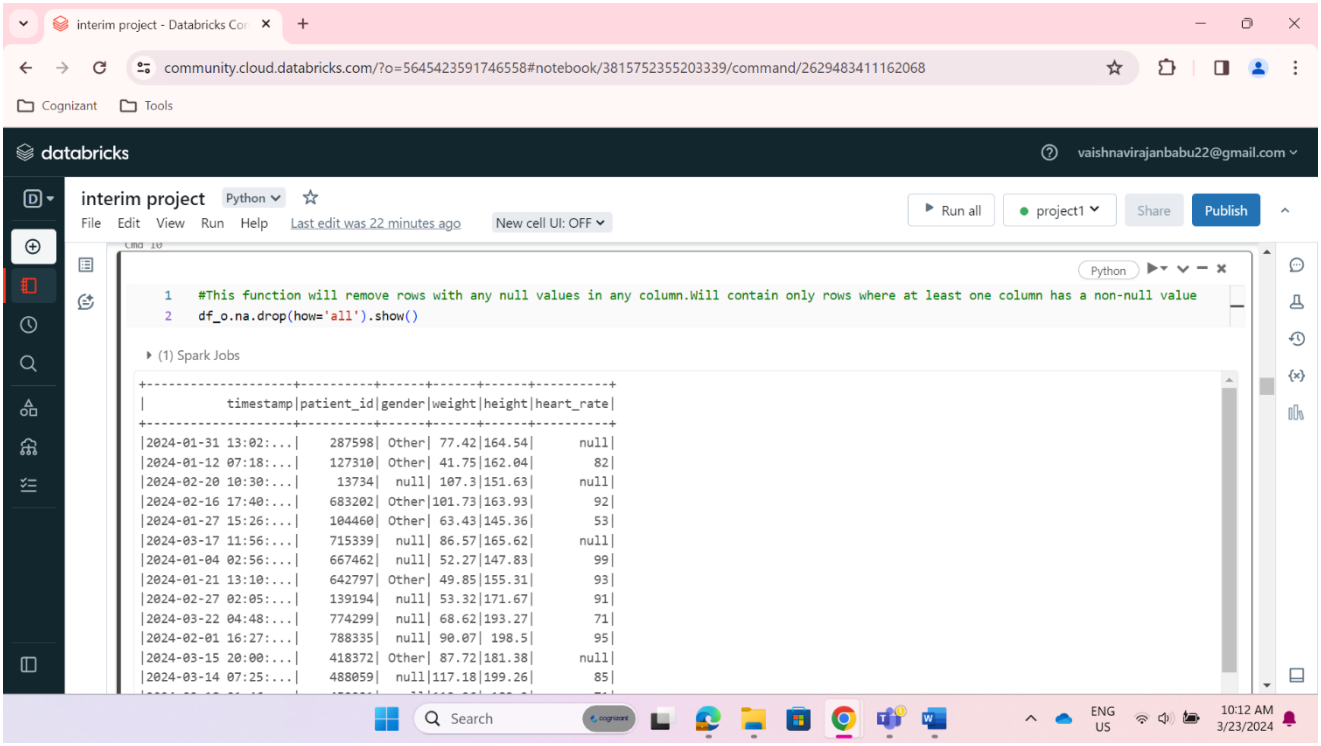
STEP 10: Remove rows with any null value in any column.



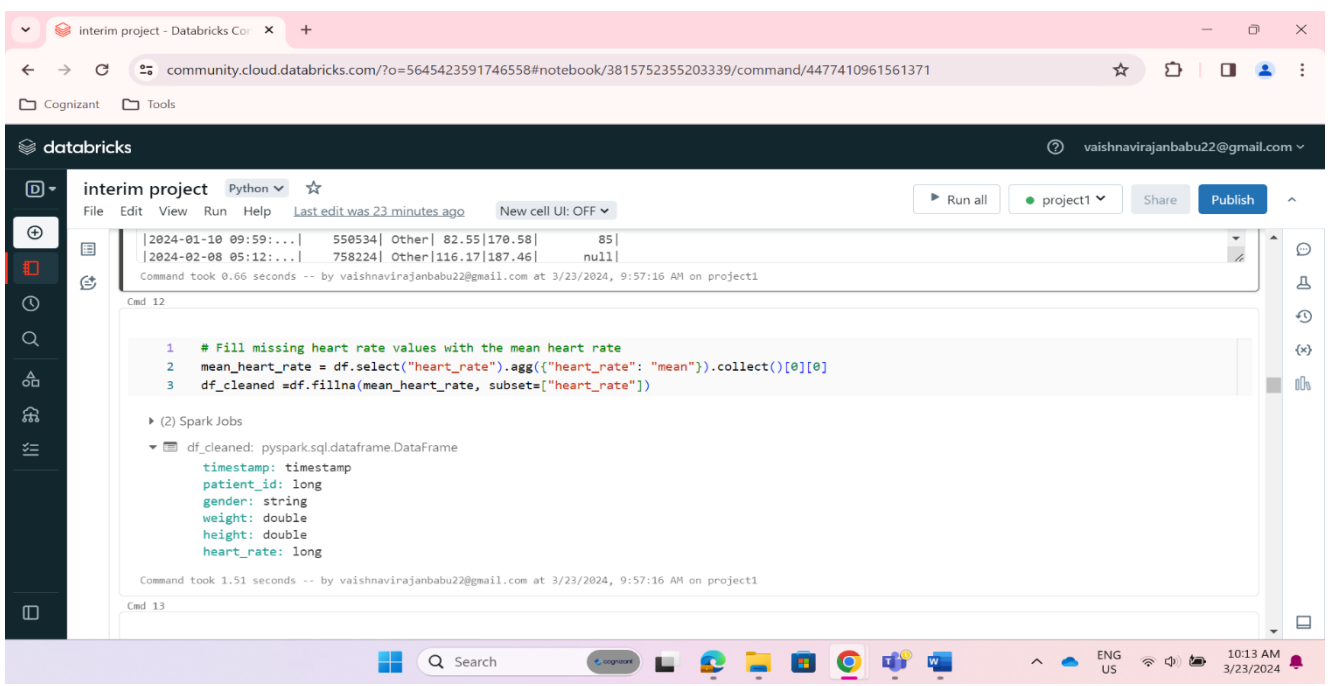
STEP 11: Create a gender column only for ‘other’ gender.

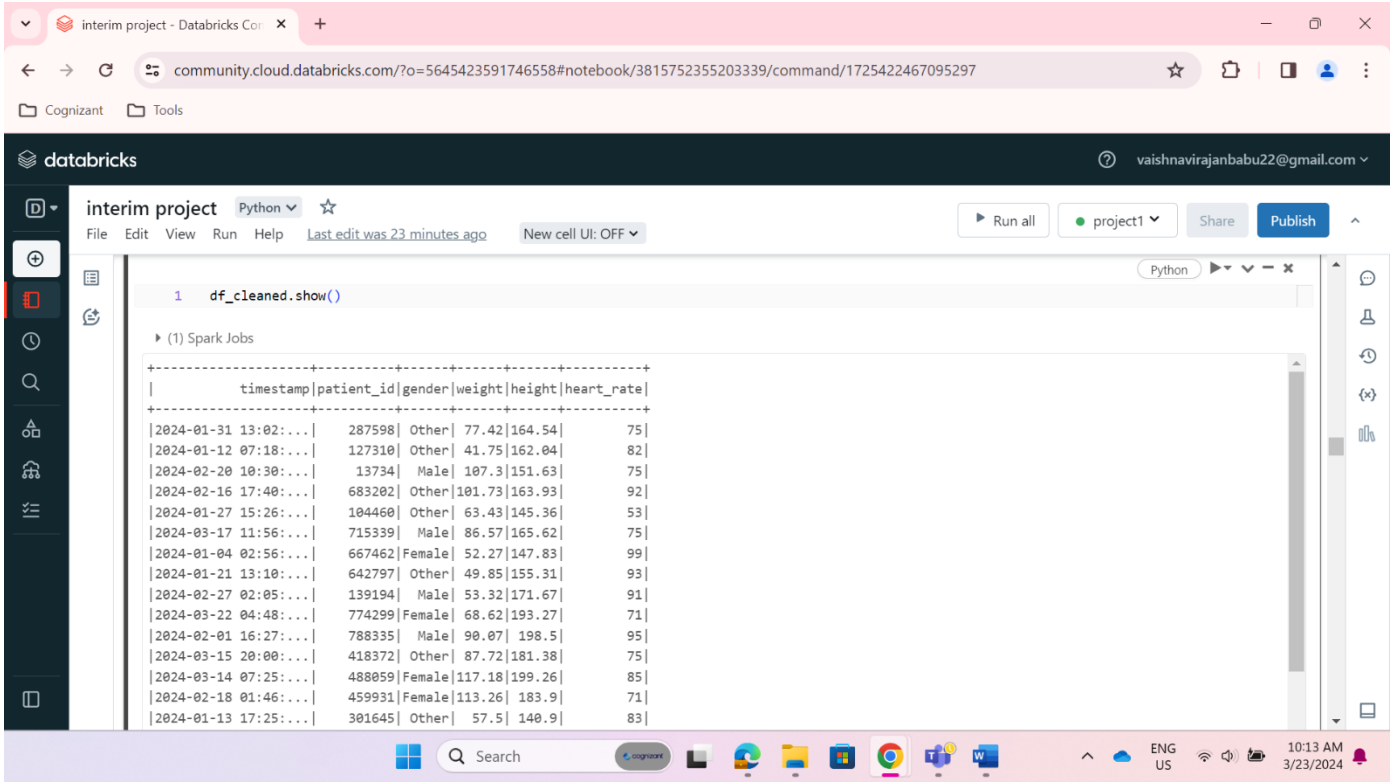


STEP 12: This function will filter rows where atleast one column has a non-null value.

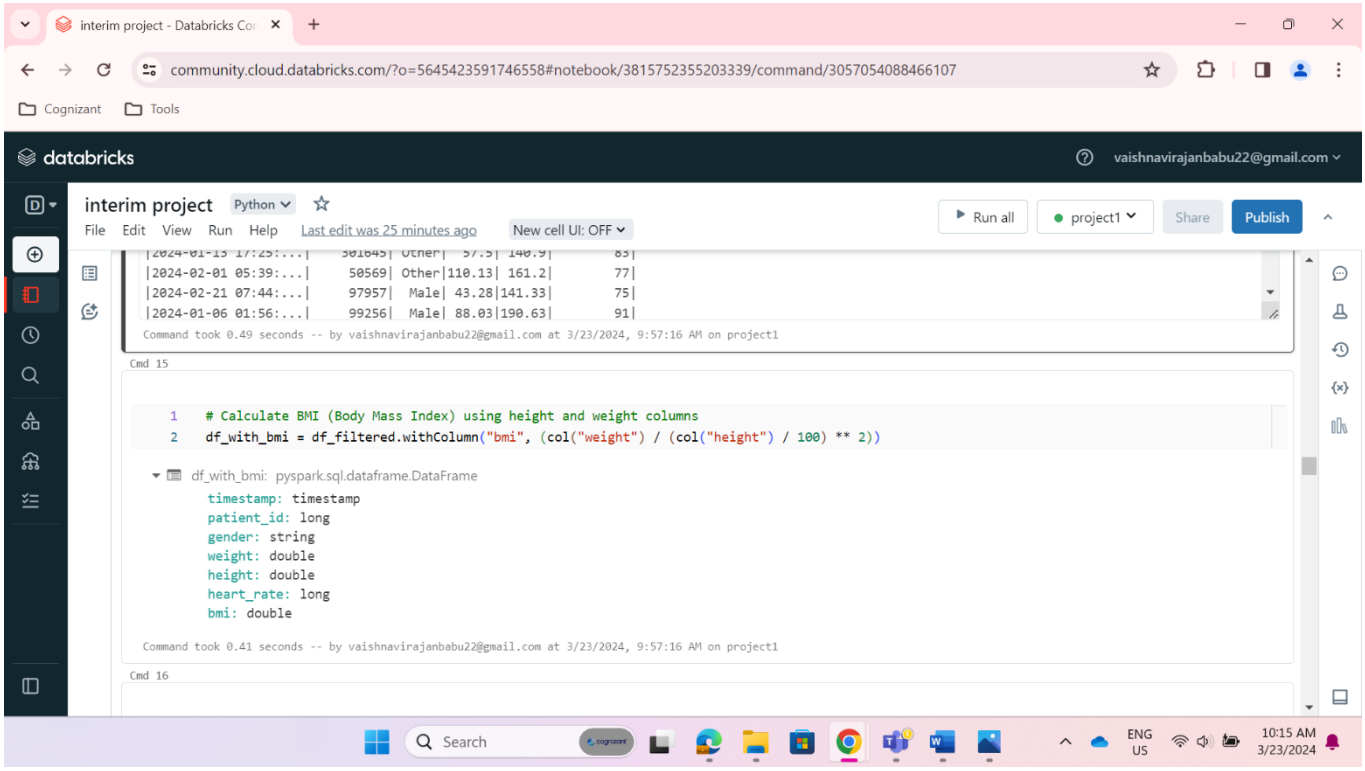


STEP 13: This function is used to fill the missing heart rate values with mean heart rate value.





STEP 14: Calculate BMI using height and weight columns, and set a threshold for the same. Now categorize weight status of patients using that threshold.



interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180704

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 25 minutes ago New cell UI: OFF

Run all project1 Share Publish

```python
# Round off the "bmi" column values to 2 decimal places
from pyspark.sql.functions import round
df_with_bmi_rounded = df_with_bmi.withColumn("bmi", round(col("bmi"), 2))
df_with_bmi_rounded.show()
```

(1) Spark Jobs

df_with_bmi_rounded: pyspark.sql.dataframe.DataFrame = [timestamp: timestamp, patient_id: long ... 5 more fields]

```
+--------------------+----------+------+------+------+----------+-----+
|           timestamp|patient_id|gender|weight|height|heart_rate|  bmi|
+--------------------+----------+------+------+------+----------+-----+
|2024-01-31 13:02:...|    287598| Other| 77.42|164.54|        75| 28.6|
|2024-01-12 07:18:...|    127310| Other| 41.75|162.04|        82| 15.9|
|2024-02-20 10:30:...|     13734|  Male| 107.3|151.63|        75|46.67|
|2024-02-16 17:40:...|    683202| Other|101.73|163.93|        92|37.86|
|2024-01-27 15:26:...|    104460| Other| 63.43|145.36|        53|30.02|
|2024-03-17 11:56:...|    715339|  Male| 86.57|165.62|        75|31.56|
|2024-01-04 02:56:...|    667462|Female| 52.27|147.83|        99|23.92|
|2024-01-21 13:10:...|    642797| Other| 49.85|155.31|        93|20.67|
|2024-02-27 02:05:...|    139194|  Male| 53.32|171.67|        91|18.09|
|2024-03-22 04:48:...|    774299|Female| 68.62|193.27|        71|18.37|
```

Search ENG US 10:15 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180706

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 26 minutes ago New cell UI: OFF

Run all project1 Share Publish

```
|2024-02-27 02:05:...|    139194|  Male| 53.32|171.67|        91|18.09|
|2024-03-22 04:48:...|    774299|Female| 68.62|193.27|        71|18.37|
|2024-02-01 16:27:...|    788335|  Male| 90.07| 198.5|        95|22.86|
|2024-03-15 20:00:...|    418372| Other| 87.72|181.38|        75|26.66|
|2024-03-14 07:25:...|    488059|Female|117.18|199.26|        85|29.51|
|2024-02-18 01:46:...|    459931|Female|113.26| 183.9|        71|33.49|
|2024-01-13 17:25:...|    301645| Other|  57.5| 140.9|        83|28.96|
|2024-02-01 05:39:...|     50569| Other|110.13| 161.2|        77|42.38|
|2024-02-21 07:44:...|     97957|  Male| 43.28|141.33|        75|21.67|
|2024-01-06 01:56:...|     99256|  Male| 88.03|190.63|        91|24.22|
```

Command took 0.48 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 18

```python
# Define BMI thresholds for weight status
underweight_threshold = 18.5
normal_weight_threshold = 24.9
overweight_threshold = 29.9
```

Command took 0.08 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 19

Search ENG US 10:16 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180706

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 26 minutes ago New cell UI: OFF

Run all project1 Share Publish

```python
# Categorize weight status based on BMI
df_with_weight_status = df_with_bmi_rounded.withColumn(
    "weight_status",
    when(col("bmi") < underweight_threshold, "Underweight")
    .when((col("bmi") >= underweight_threshold) & (col("bmi") <= normal_weight_threshold), "Normal weight")
    .when((col("bmi") > normal_weight_threshold) & (col("bmi") <= overweight_threshold), "Overweight")
    .otherwise("Obesity")
)
```

df_with_weight_status: pyspark.sql.dataframe.DataFrame

```
timestamp: timestamp
patient_id: long
gender: string
weight: double
height: double
heart_rate: long
bmi: double
weight_status: string
```

Command took 0.38 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 20

Search ENG US 10:16 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180707

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 26 minutes ago New cell UI: OFF

Run all project1 Share Publish

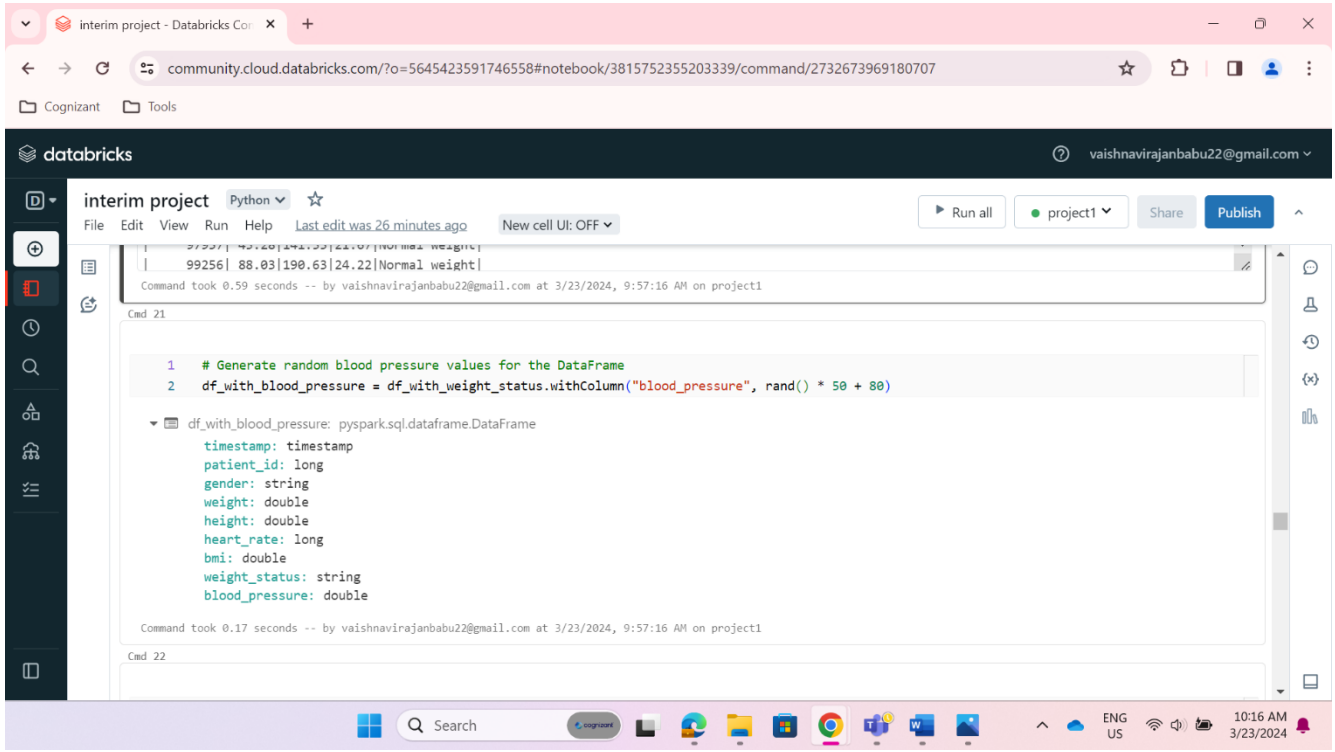
```python
# Show the first few rows of the DataFrame with weight status
df_with_weight_status.select("patient_id","weight","height","bmi","weight_status").show()
```

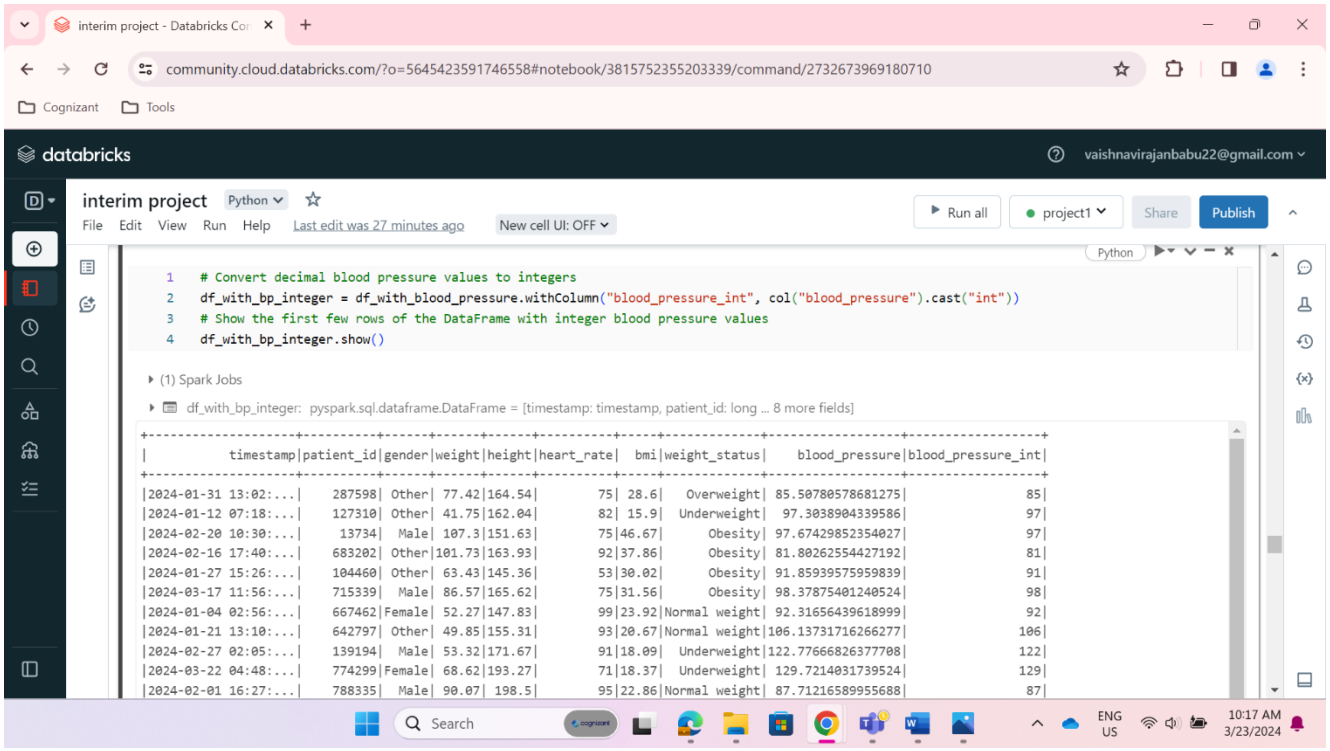
(1) Spark Jobs

```
+----------+------+------+-----+-------------+
|patient_id|weight|height|  bmi|weight_status|
+----------+------+------+-----+-------------+
|    287598| 77.42|164.54| 28.6|   Overweight|
|    127310| 41.75|162.04| 15.9|  Underweight|
|     13734| 107.3|151.63|46.67|      Obesity|
|    683202|101.73|163.93|37.86|      Obesity|
|    104460| 63.43|145.36|30.02|      Obesity|
|    715339| 86.57|165.62|31.56|      Obesity|
|    667462| 52.27|147.83|23.92|Normal weight|
|    642797| 49.85|155.31|20.67|Normal weight|
|    139194| 53.32|171.67|18.09|  Underweight|
|    774299| 68.62|193.27|18.37|  Underweight|
|    788335| 90.07| 198.5|22.86|Normal weight|
|    418372| 87.72|181.38|26.66|   Overweight|
|    488059|117.18|199.26|29.51|   Overweight|
|    459931|113.26| 183.9|33.49|      Obesity|
```

Search ENG US 10:16 AM 3/23/2024

STEP 15: Generate blood pressure of patients using rand function and categorize blood pressure of patients.





interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180711

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 30 minutes ago New cell UI: OFF

Run all project1 Share Publish

```
|2024-03-22 04:48:...|    774299|Female| 68.62|193.27|        71|18.37|  Underweight|              129|
|2024-02-01 16:27:...|    788335|  Male| 90.07| 198.5|        95|22.86|Normal weight|               87|
|2024-03-15 20:00:...|    418372| Other| 87.72|181.38|        75|26.66|   Overweight|              125|
|2024-03-14 07:25:...|    488059|Female|117.18|199.26|        85|29.51|   Overweight|              105|
|2024-02-18 01:46:...|    459931|Female|113.26| 183.9|        71|33.49|      Obesity|              125|
|2024-01-13 17:25:...|    301645| Other|  57.5| 140.9|        83|28.96|   Overweight|               89|
|2024-02-01 05:39:...|     50569| Other|110.13| 161.2|        77|42.38|      Obesity|              116|
|2024-02-21 07:44:...|     97957|  Male| 43.28|141.33|        75|21.67|Normal weight|              123|
|2024-01-06 01:56:...|     99256|  Male| 88.03|190.63|        91|24.22|Normal weight|              101|
```

Command took 0.70 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 25

```python
# Define blood pressure thresholds for categorization
normal_bp_threshold = 80
elevated_bp_threshold = 120
high_bp_stage1_threshold = 130
high_bp_stage2_threshold = 140
```

Command took 0.09 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 26

Search ENG US 10:19 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180711

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 30 minutes ago New cell UI: OFF

Run all project1 Share Publish

```python
# Categorize blood pressure based on the thresholds
df_with_bp_category = df_with_bp_integer.withColumn(
    "bp_category",
    when(col("blood_pressure_int") < normal_bp_threshold, "Low")
    .when((col("blood_pressure_int") >= normal_bp_threshold) & (col("blood_pressure_int") < elevated_bp_threshold), "Normal")
    .when((col("blood_pressure_int") >= elevated_bp_threshold) & (col("blood_pressure_int") < high_bp_stage1_threshold), "Elevated")
    .when((col("blood_pressure_int") >= high_bp_stage1_threshold) & (col("blood_pressure_int") < high_bp_stage2_threshold), "High (Stage 1)
    ")
    .otherwise("High (Stage 2 or higher)")
)
```

df_with_bp_category: pyspark.sql.dataframe.DataFrame

```
timestamp: timestamp
patient_id: long
gender: string
weight: double
height: double
heart_rate: long
bmi: double
weight_status: string
blood_pressure_int: integer
bp_category: string
```

Search ENG US 10:20 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180714

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 30 minutes ago New cell UI: OFF

Run all project1 Share Publish

```python
# Show the first few rows of the DataFrame with bp_category column
df_with_bp_category.select("patient_id","blood_pressure_int","bp_category").show()
```

(1) Spark Jobs

```
+----------+------------------+-----------+
|patient_id|blood_pressure_int|bp_category|
+----------+------------------+-----------+
|    287598|                85|     Normal|
|    127310|                97|     Normal|
|     13734|                97|     Normal|
|    683202|                81|     Normal|
|    104460|                91|     Normal|
|    715339|                98|     Normal|
|    667462|                92|     Normal|
|    642797|               106|     Normal|
|    139194|               122|   Elevated|
|    774299|               129|   Elevated|
|    788335|                87|     Normal|
|    418372|               125|   Elevated|
|    488059|               105|     Normal|
|    459931|               125|   Elevated|
```

Search ENG US 10:20 AM 3/23/2024

STEP 16: Set threshold values for heart rate and categorize heart rate of patients.

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180714

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 30 minutes ago New cell UI: OFF

Run all project1 Share Publish

```
|    642797|               106|     Normal|
|    139194|               122|   Elevated|
|    774299|               129|   Elevated|
|    788335|                87|     Normal|
|    418372|               125|   Elevated|
|    488059|               105|     Normal|
|    459931|               125|   Elevated|
|    301645|                89|     Normal|
|     50569|               116|     Normal|
|     97957|               123|   Elevated|
|     99256|               101|     Normal|
```

Command took 0.59 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 28

```python
# Define heart rate thresholds for categorization
low_hr_threshold = 60
normal_hr_threshold = 100
```

Command took 0.09 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Cmd 29

Search ENG US 10:20 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180714

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 30 minutes ago New cell UI: OFF

Run all project1 Share Publish

```python
# Categorize heart rate based on the thresholds
df_with_hr_category = df_with_bp_category.withColumn(
    "heart_rate_category",
    when(col("heart_rate").isNull(), "Unknown")
    .when(col("heart_rate") < low_hr_threshold, "Low")
    .when((col("heart_rate") >= low_hr_threshold) & (col("heart_rate") <= normal_hr_threshold), "Normal")
    .otherwise("Elevated")
)
```

df_with_hr_category: pyspark.sql.dataframe.DataFrame

```
timestamp: timestamp
patient_id: long
gender: string
weight: double
height: double
heart_rate: long
bmi: double
weight_status: string
blood_pressure_int: integer
bp_category: string
heart_rate_category: string
```

Command took 0.19 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 9:57:16 AM on project1

Search ENG US 10:20 AM 3/23/2024

interim project - Databricks Co

community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2732673969180717

Cognizant Tools

databricks vaishnavirajanbabu22@gmail.com

interim project Python

File Edit View Run Help Last edit was 31 minutes ago New cell UI: OFF

Run all project1 Share Publish

Python

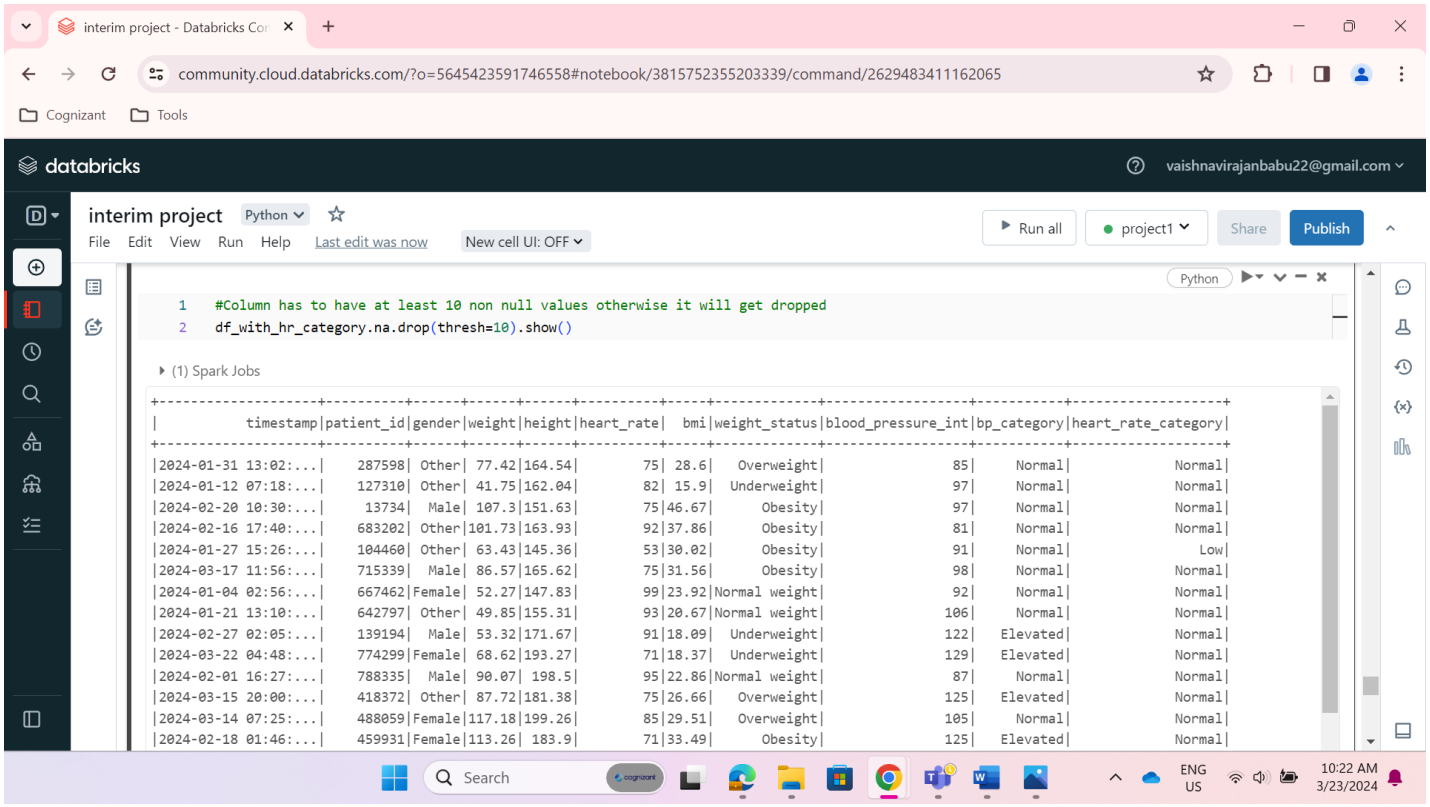
```python
# Show the first few rows of the DataFrame with heart rate category
df_with_hr_category.select("patient_id","heart_rate","heart_rate_category").show()
```

(1) Spark Jobs

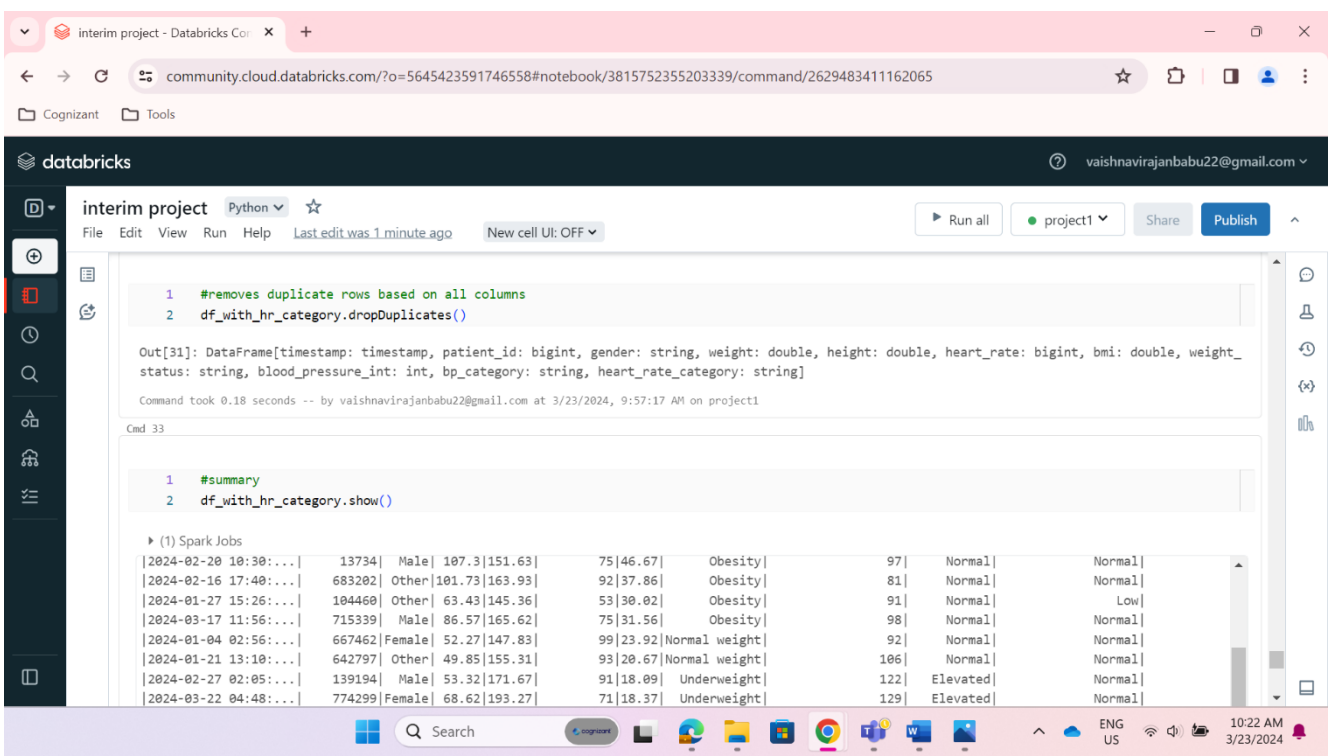
```
+----------+----------+------------------+
|patient_id|heart_rate|heart_rate_category|
+----------+----------+------------------+
|    287598|        75|            Normal|
|    127310|        82|            Normal|
|     13734|        75|            Normal|
|    683202|        92|            Normal|
|    104460|        53|               Low|
|    715339|        75|            Normal|
|    667462|        99|            Normal|
|    642797|        93|            Normal|
|    139194|        91|            Normal|
|    774299|        71|            Normal|
|    788335|        95|            Normal|
|    418372|        75|            Normal|
|    488059|        85|            Normal|
|    459931|        71|            Normal|
|    301645|        83|            Normal|
```

Search ENG US 10:20 AM 3/23/2024

STEP 17: This function will drop the entire column if it does not contain atleast 10 non null values. Since 10 is the threshold value.

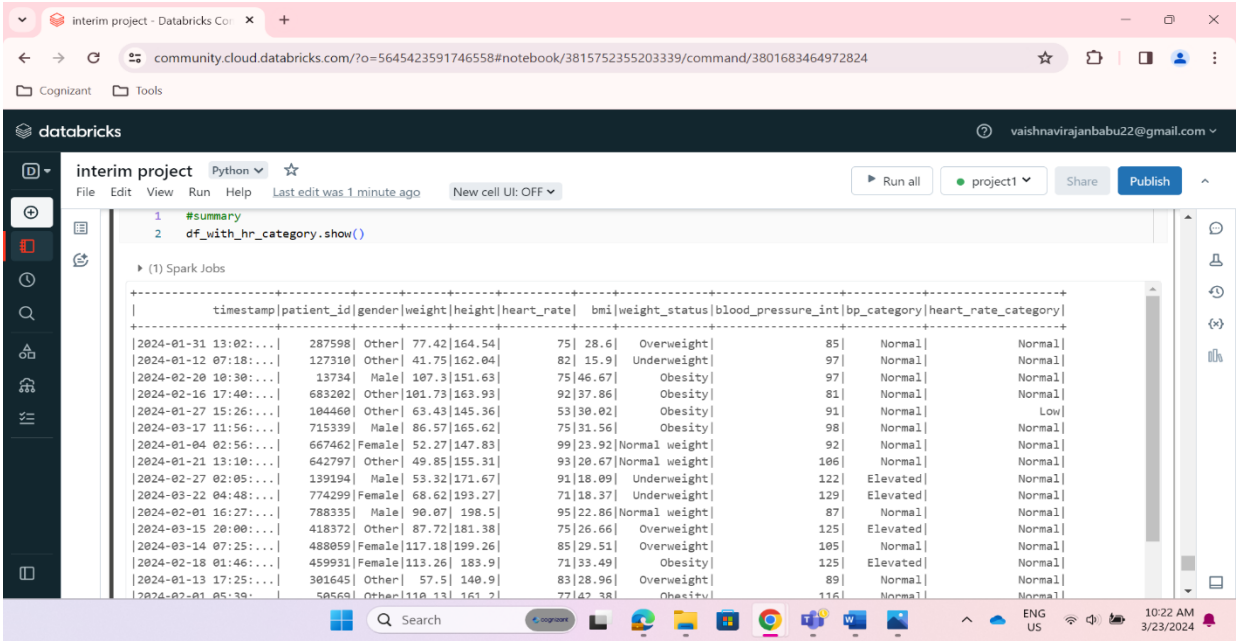


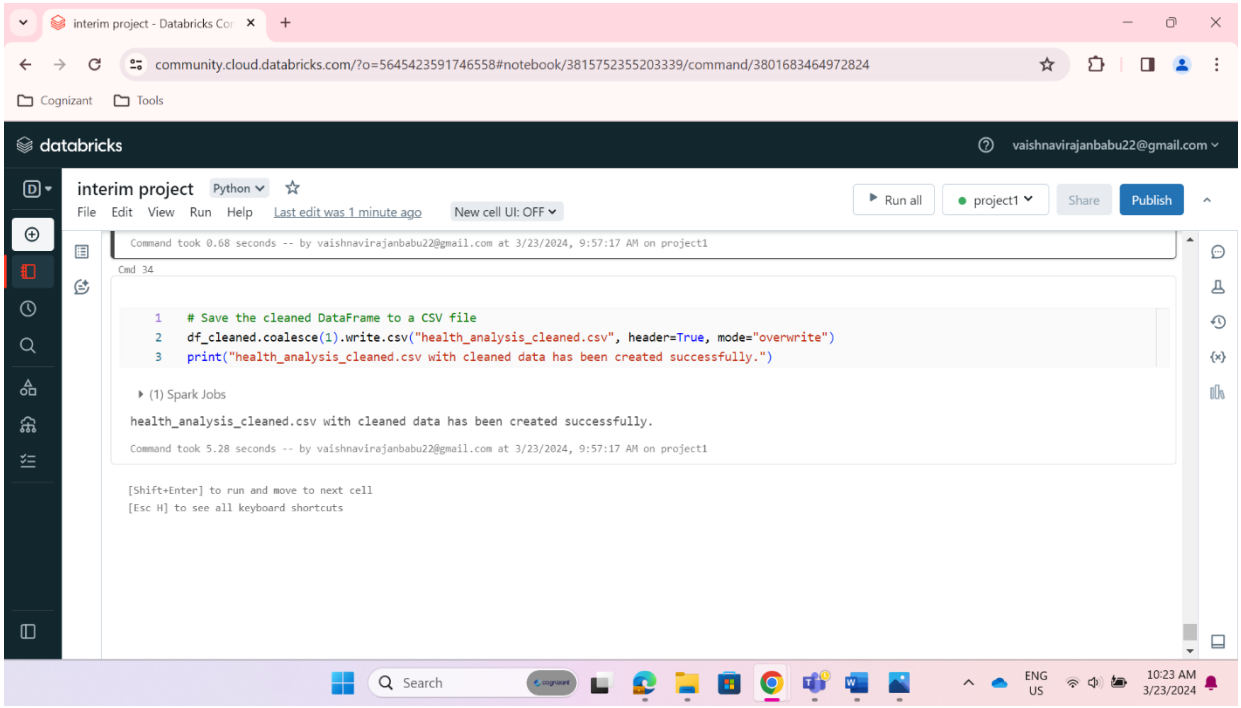
STEP 18: Remove duplicates in the dataset.



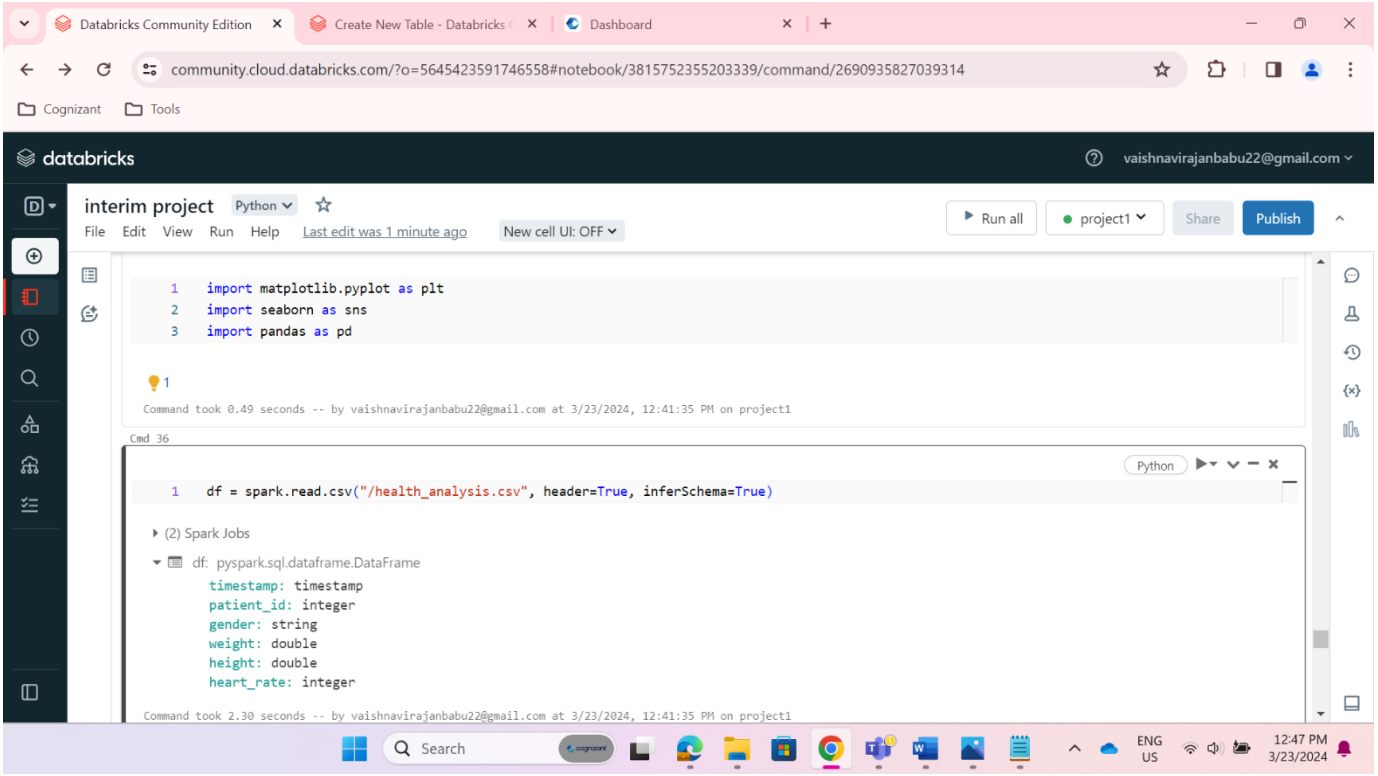
RESULT: The cleaned dataset is saved in a new CSV file ‘health_analysis_cleaned.csv’

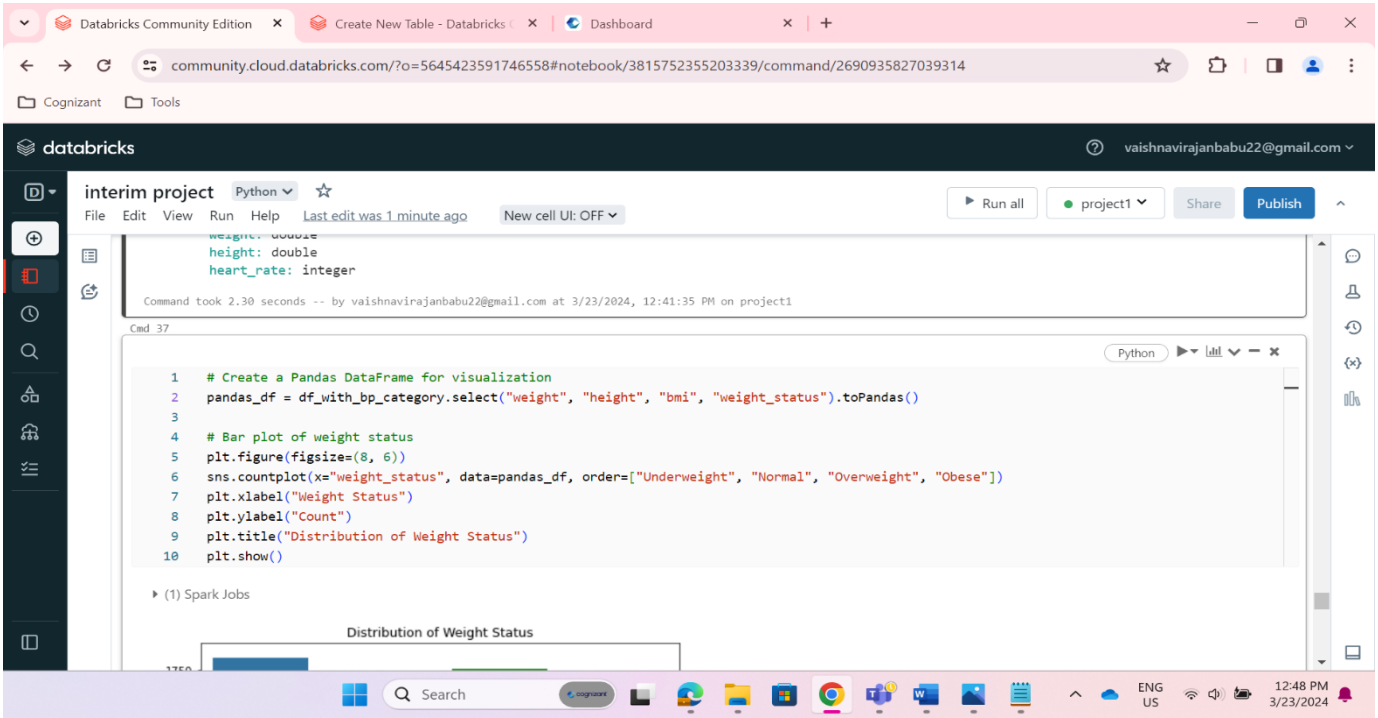
Path: dbfs:FileStore/health_analysis_cleaned.csv

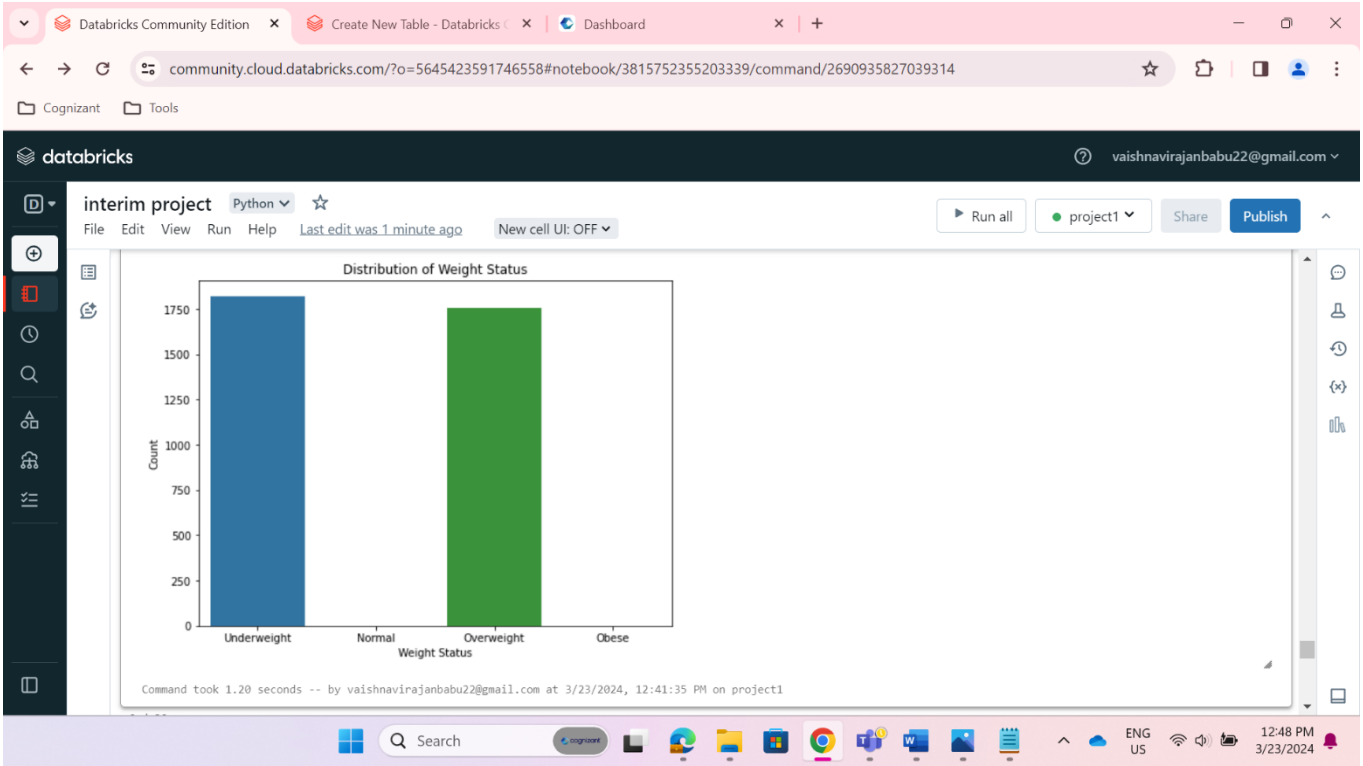


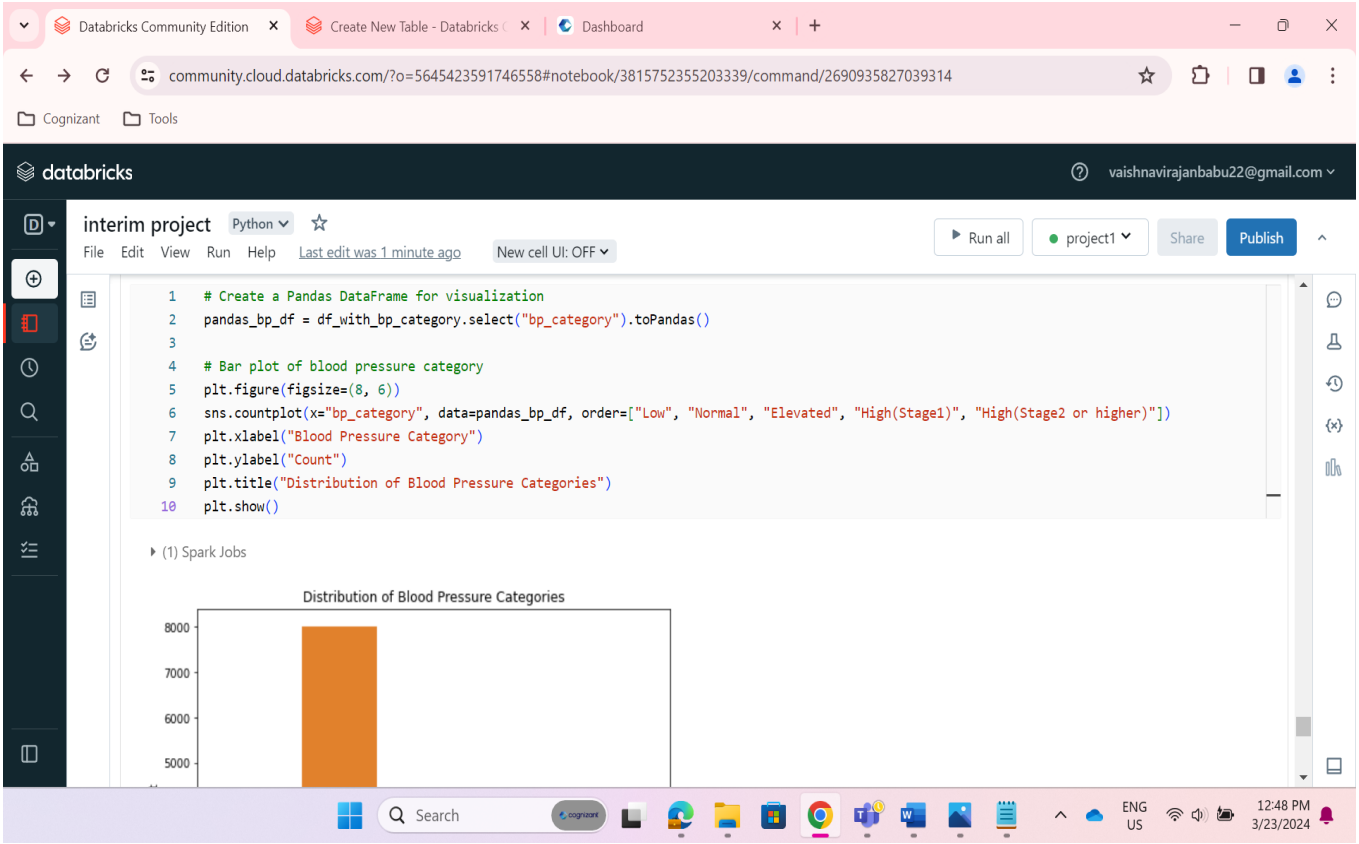


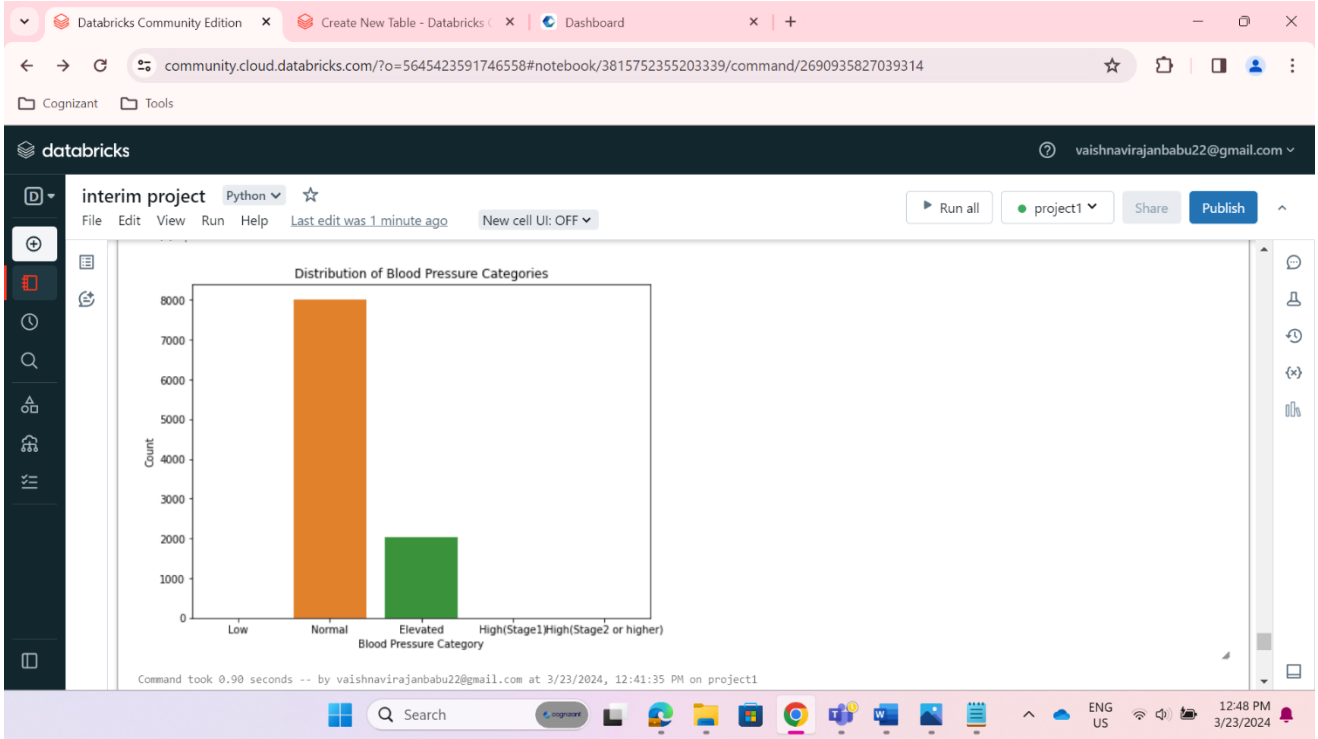
VISUALIZATION:

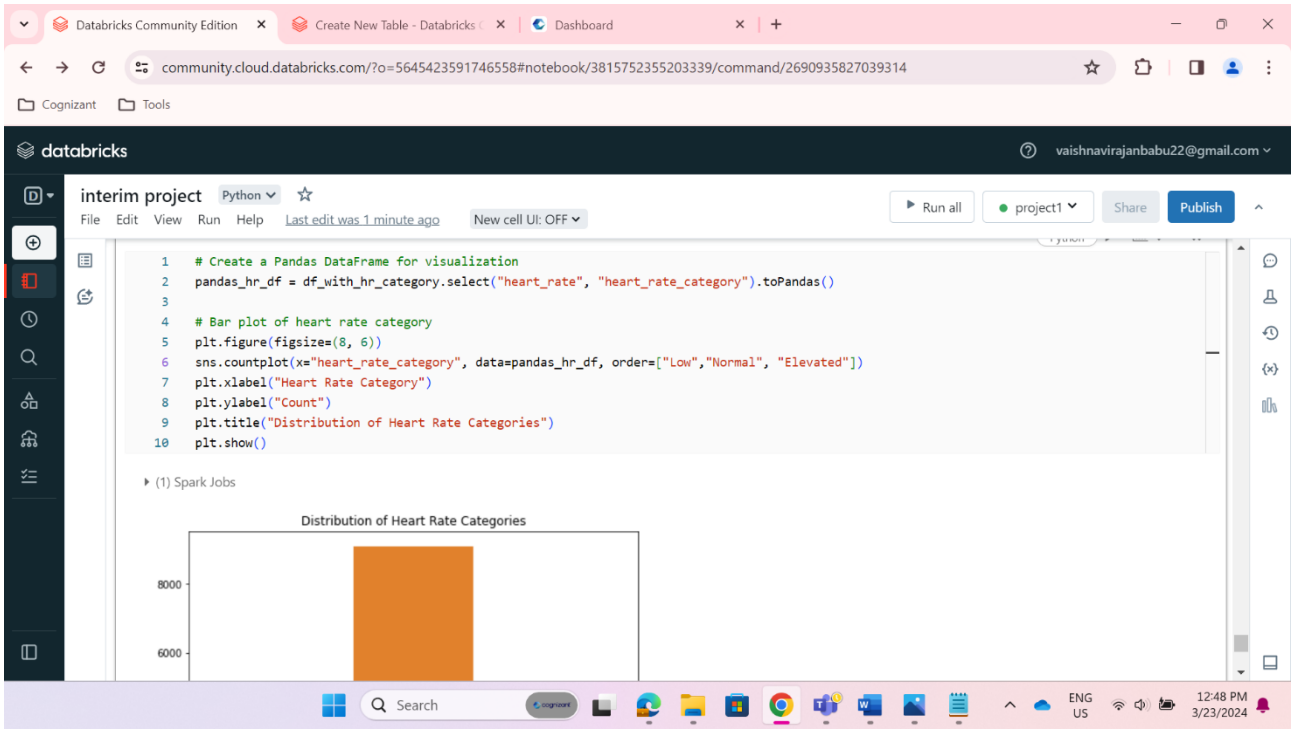





Databricks Community Edition
Create New Table - Databricks
Dashboard
community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2690935827039314
Cognizant
Tools
databricks
vaishnavirajanbabu22@gmail.com
interim project
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: OFF
Run all
project1
Share
Publish
Distribution of Weight Status
Count
1750
1500
1250
1000
750
500
250
0
Underweight
Normal
Overweight
Obese
Weight Status
Command took 1.20 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 12:41:35 PM on project1
Search
ENG
US
12:48 PM
3/23/2024

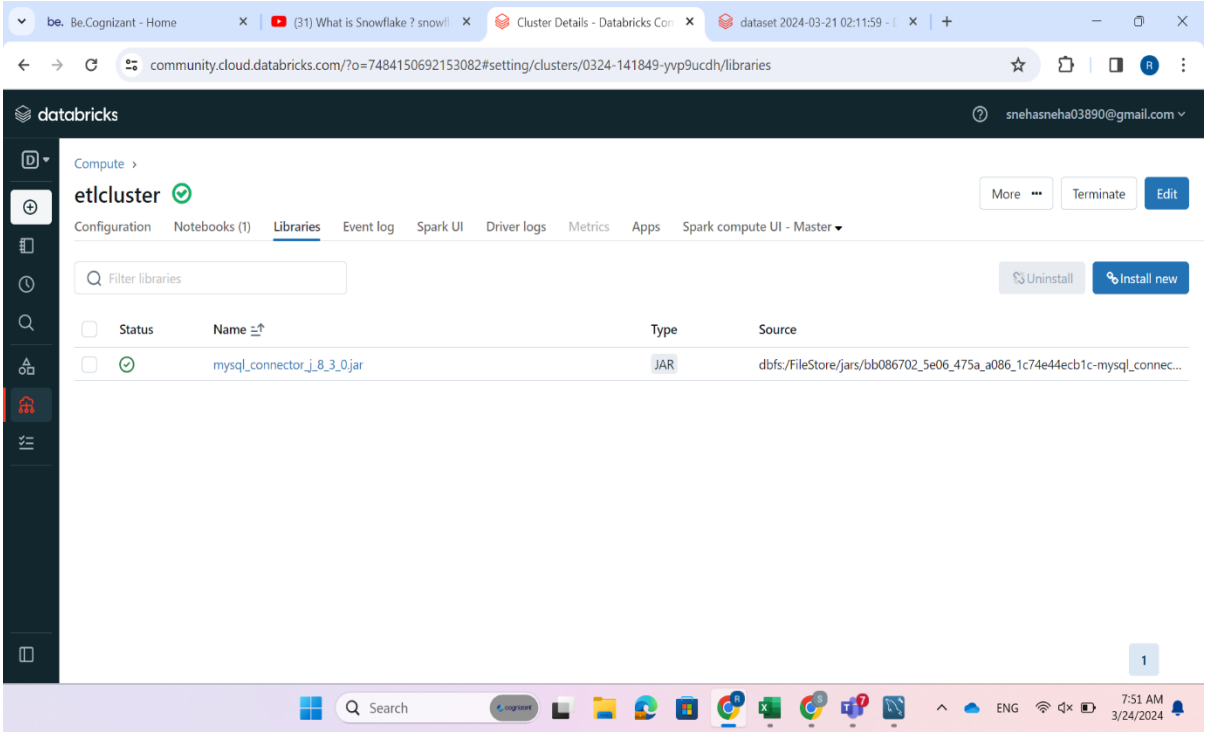

Databricks Community Edition
Create New Table - Databricks
Dashboard
community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2690935827039314
Cognizant
Tools
databricks
vaishnavirajanbabu22@gmail.com
interim project
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: OFF
Run all
project1
Share
Publish
# Create a Pandas DataFrame for visualization
pandas_bp_df = df_with_bp_category.select("bp_category").toPandas()
# Bar plot of blood pressure category
plt.figure(figsize=(8, 6))
sns.countplot(x="bp_category", data=pandas_bp_df, order=["Low", "Normal", "Elevated", "High(Stage1)", "High(Stage2 or higher)"])
plt.xlabel("Blood Pressure Category")
plt.ylabel("Count")
plt.title("Distribution of Blood Pressure Categories")
plt.show()
(1) Spark Jobs
Distribution of Blood Pressure Categories
8000
7000
6000
5000
Search
ENG
US
12:48 PM
3/23/2024


Databricks Community Edition
Create New Table - Databricks
Dashboard
community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2690935827039314
Cognizant
Tools
databricks
vaishnavirajanbabu22@gmail.com
interim project
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: OFF
Run all
project1
Share
Publish
Distribution of Blood Pressure Categories
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
Low
Normal
Elevated
High(Stage1)
High(Stage2 or higher)
Blood Pressure Category
Command took 0.90 seconds -- by vaishnavirajanbabu22@gmail.com at 3/23/2024, 12:41:35 PM on project1
Search
ENG
US
12:48 PM
3/23/2024

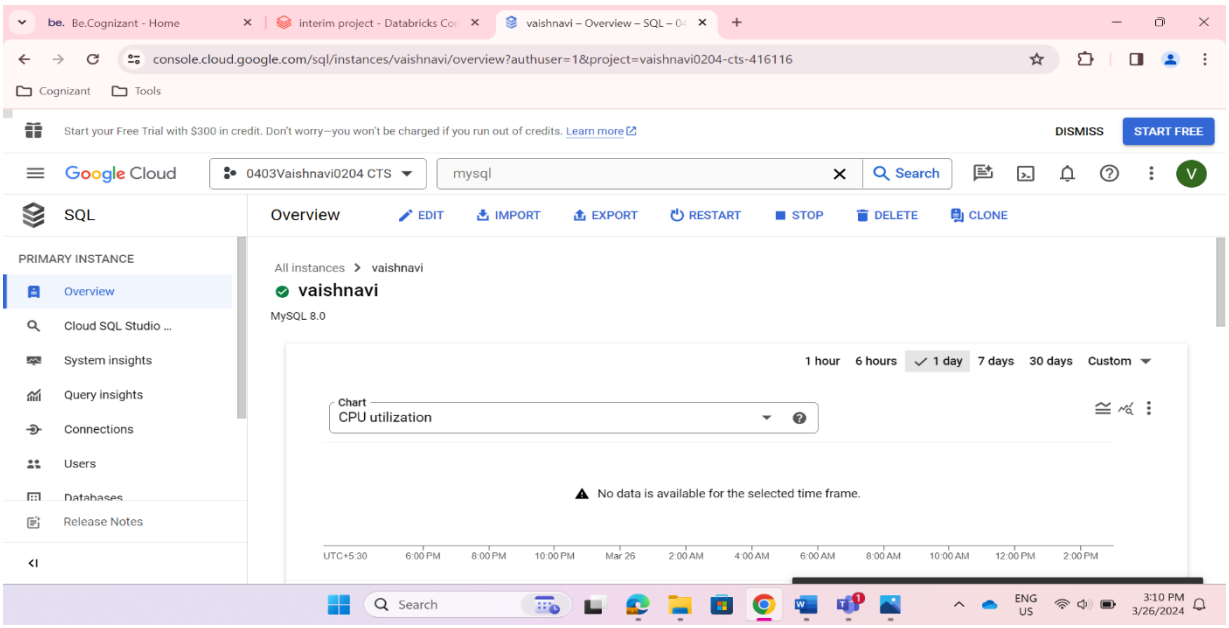

Databricks Community Edition
Create New Table - Databricks
Dashboard
community.cloud.databricks.com/?o=5645423591746558#notebook/3815752355203339/command/2690935827039314
Cognizant
Tools
databricks
vaishnavirajanbabu22@gmail.com
interim project
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: OFF
Run all
project1
Share
Publish
# Create a Pandas DataFrame for visualization
pandas_hr_df = df_with_hr_category.select("heart_rate", "heart_rate_category").toPandas()
# Bar plot of heart rate category
plt.figure(figsize=(8, 6))
sns.countplot(x="heart_rate_category", data=pandas_hr_df, order=["Low","Normal", "Elevated"])
plt.xlabel("Heart Rate Category")
plt.ylabel("Count")
plt.title("Distribution of Heart Rate Categories")
plt.show()
(1) Spark Jobs
Distribution of Heart Rate Categories
8000
6000
Search
ENG
US
12:48 PM
3/23/2024

LOADING INTO MYSQL DATABASE:

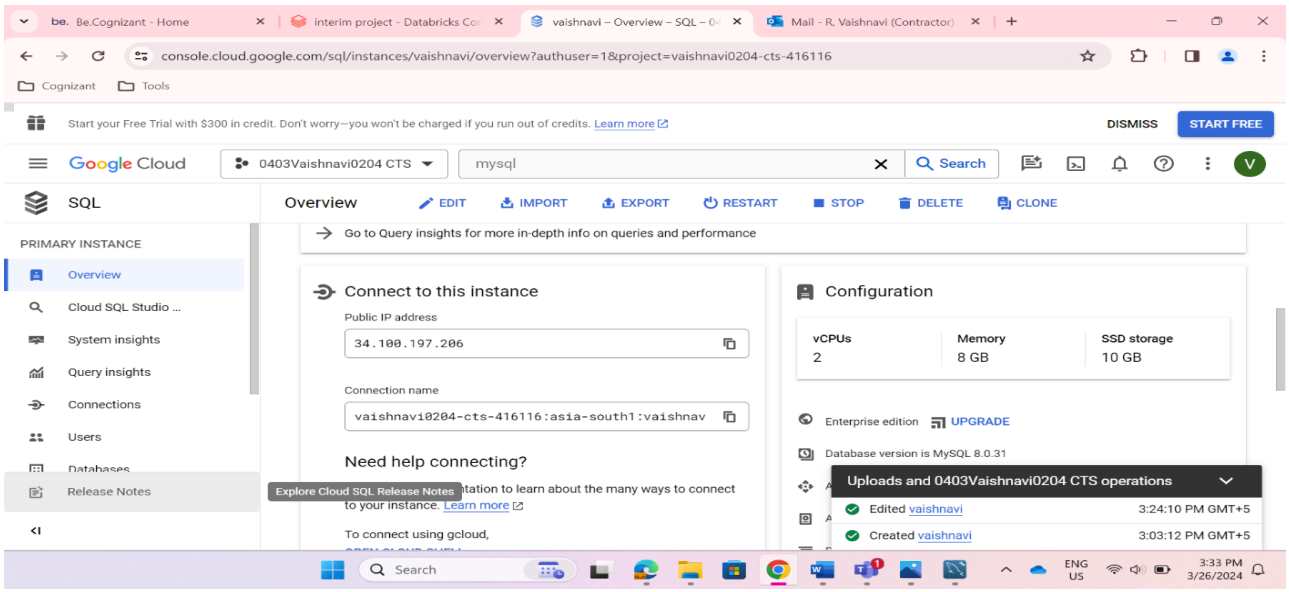
- Download mySQL connector jar file from Google and upload the file in Databricks libraries.



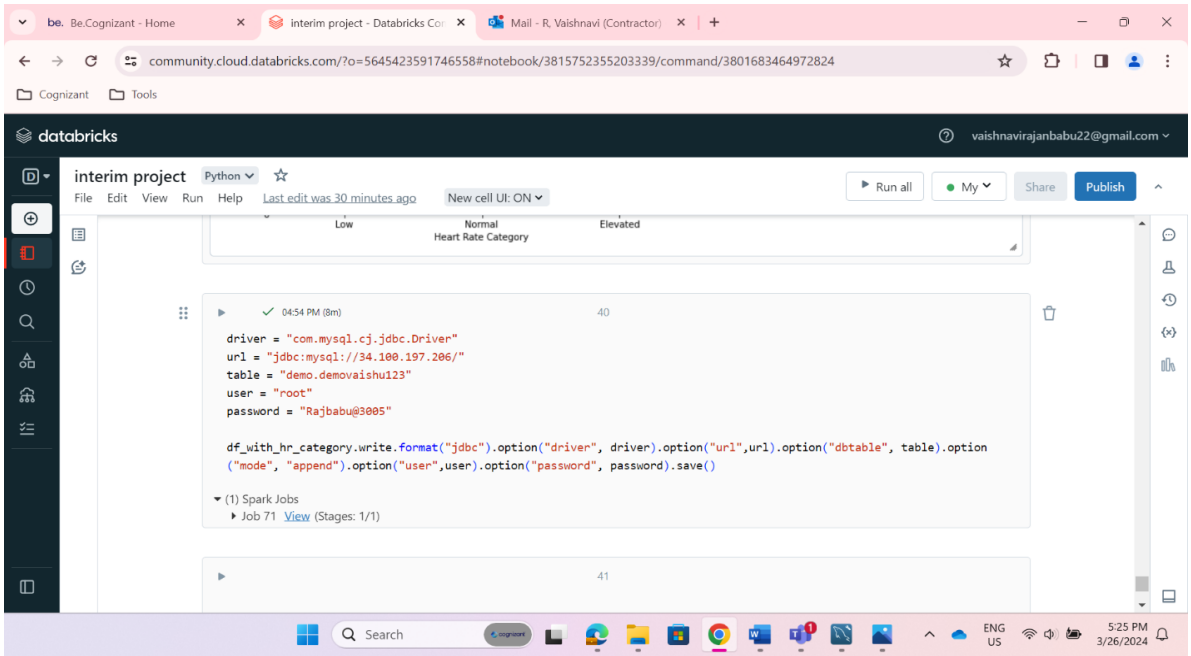
- Open the Google cloud console and create CloudSQL instance.



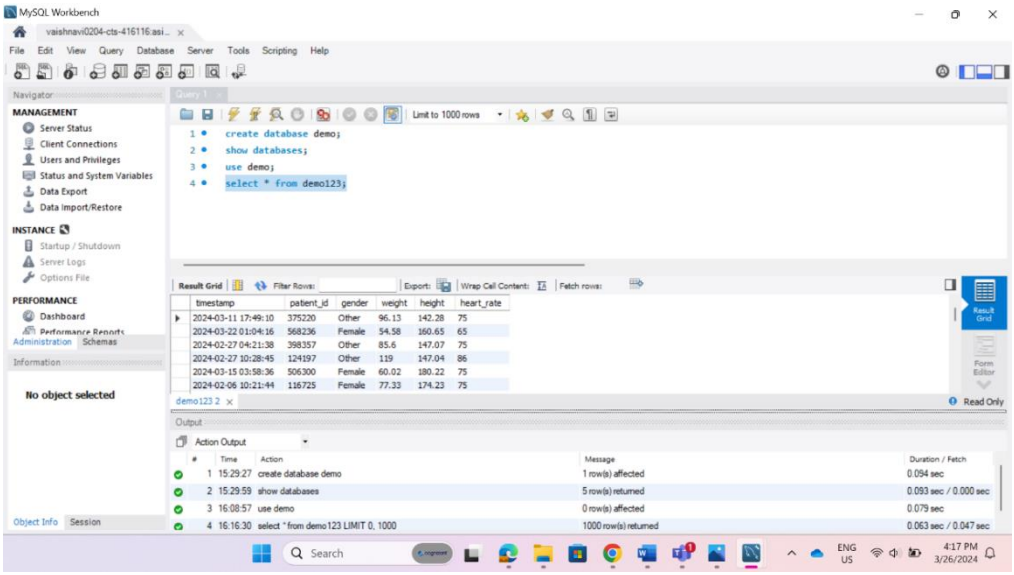
- After creating the CloudSQL instance, copy the public IP address.



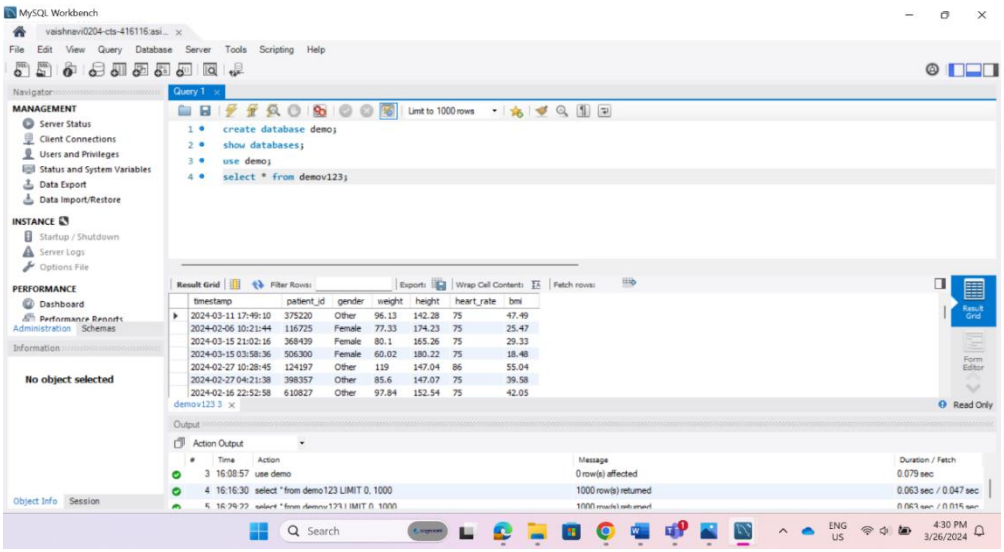
- The code aims to save data from a DataFrame into a MySQL database table. It uses the JDBC driver to establish a connection between Python and the MySQL database.

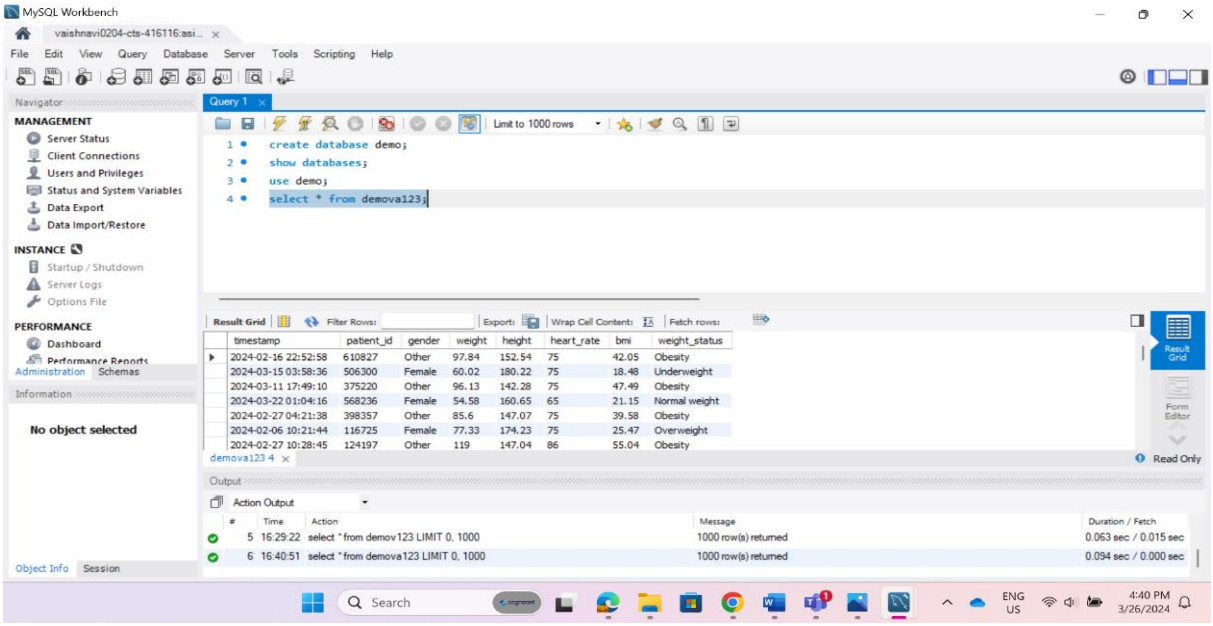


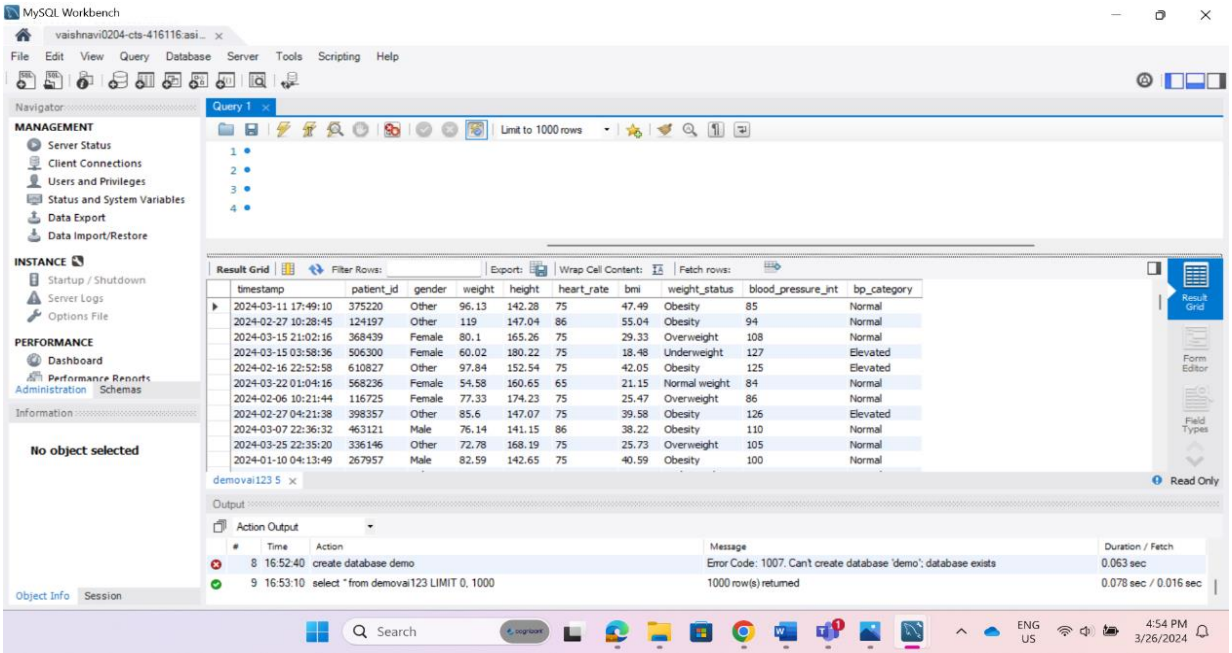
- Once connection is done, open MySQL workbench and create a database demo, use show database query to check the available databases. Use the database demo that we created and select all the columns from table demo123.

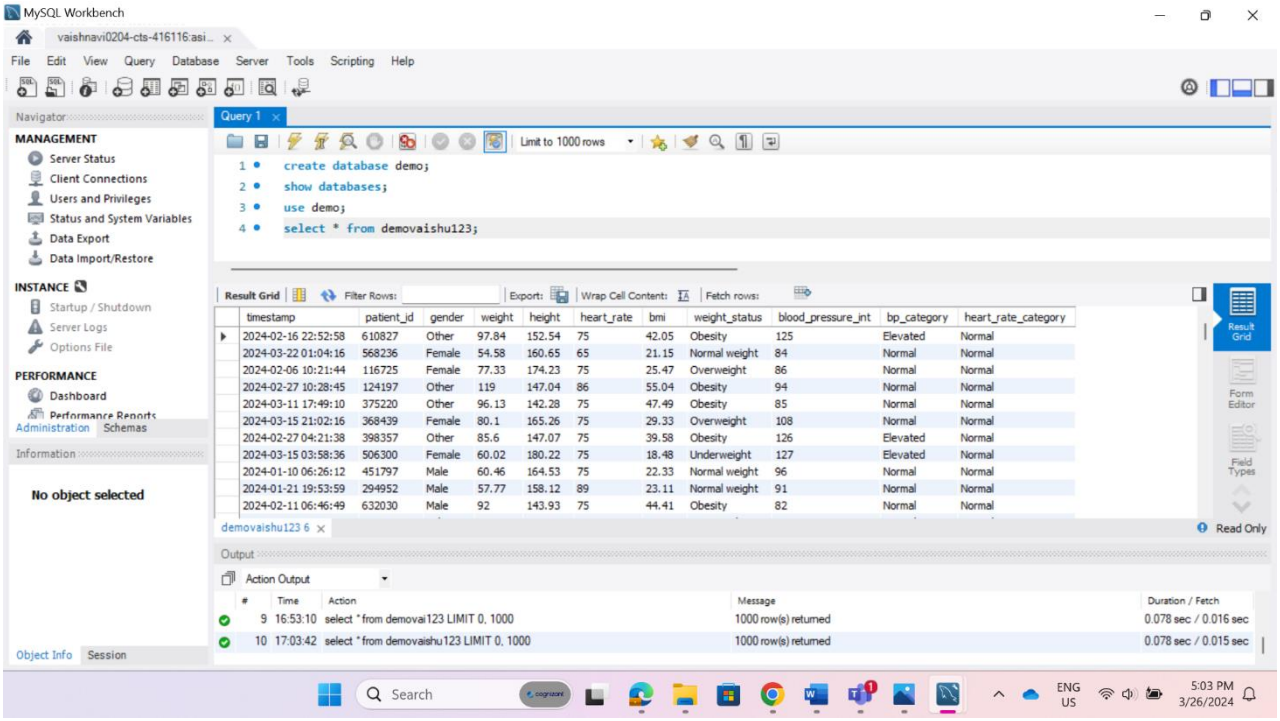


- Continue the same process for other tables demov123, demova123, demovai123, demovaishu123




MySQL Workbench
vaishnavi0204-cts-416116:asi...
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
create database demo;
show databases;
use demo;
select * from demova123;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
timestamp patient_id gender weight height heart_rate bmi weight_status
2024-02-16 22:52:58 610827 Other 97.84 152.54 75 42.05 Obesity
2024-03-15 03:58:36 506300 Female 60.02 180.22 75 18.48 Underweight
2024-03-11 17:49:10 375220 Other 96.13 142.28 75 47.49 Obesity
2024-03-22 01:04:16 568236 Female 54.58 160.65 65 21.15 Normal weight
2024-02-27 04:21:38 398357 Other 85.6 147.07 75 39.58 Obesity
2024-02-06 10:21:44 116725 Female 77.33 174.23 75 25.47 Overweight
2024-02-27 10:28:45 124197 Other 119 147.04 86 55.04 Obesity
demova123 4
Result Grid
Form Editor
Read Only
Output
Action Output
Time Action Message Duration / Fetch
5 16:29:22 select * from demov123 LIMIT 0, 1000 1000 row(s) returned 0.063 sec / 0.015 sec
6 16:40:51 select * from demova123 LIMIT 0, 1000 1000 row(s) returned 0.094 sec / 0.000 sec
Search
ENG US
4:40 PM 3/26/2024


MySQL Workbench
vaishnavi0204-cts-416116:asi...
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
timestamp patient_id gender weight height heart_rate bmi weight_status blood_pressure_int bp_category
2024-03-11 17:49:10 375220 Other 96.13 142.28 75 47.49 Obesity 85 Normal
2024-02-27 10:28:45 124197 Other 119 147.04 86 55.04 Obesity 94 Normal
2024-03-15 21:02:16 368439 Female 80.1 165.26 75 29.33 Overweight 108 Normal
2024-03-15 03:58:36 506300 Female 60.02 180.22 75 18.48 Underweight 127 Elevated
2024-02-16 22:52:58 610827 Other 97.84 152.54 75 42.05 Obesity 125 Elevated
2024-03-22 01:04:16 568236 Female 54.58 160.65 65 21.15 Normal weight 84 Normal
2024-02-06 10:21:44 116725 Female 77.33 174.23 75 25.47 Overweight 86 Normal
2024-02-27 04:21:38 398357 Other 85.6 147.07 75 39.58 Obesity 126 Elevated
2024-03-07 22:36:32 463121 Male 76.14 141.15 86 38.22 Obesity 110 Normal
2024-03-25 22:35:20 336146 Other 72.78 168.19 75 25.73 Overweight 105 Normal
2024-01-10 04:13:49 267957 Male 82.59 142.65 75 40.59 Obesity 100 Normal
demovai123 5
Result Grid
Form Editor
Field Types
Read Only
Output
Action Output
Time Action Message Duration / Fetch
8 16:52:40 create database demo Error Code: 1007. Can't create database 'demo'; database exists 0.063 sec
9 16:53:10 select * from demovai123 LIMIT 0, 1000 1000 row(s) returned 0.078 sec / 0.016 sec
Search
ENG US
4:54 PM 3/26/2024



- While doing this process make sure to change the table name and name of dataframe in the jdbc connectivity code for each table that are loaded.

9. CONCLUSION :

In this project, we have successfully implemented a data cleansing pipeline using Apache Spark and DataFrames, focusing on removing noise, handling missing values, and correcting errors. This real-time conclusion highlights the key findings, challenges, and implications of our data cleansing process.

KEY FINDINGS:

1. Improved Data Quality: By leveraging Spark and DataFrames, we effectively cleansed the raw data, enhancing its quality and reliability for downstream analyses and applications.

2. Scalability: Spark's distributed computing capabilities allowed us to process large volumes of data efficiently, making the data cleansing pipeline scalable to handle big data scenarios.

3. Effective Noise Removal: Through data exploration and transformation techniques, we successfully identified and removed noise from the dataset, ensuring that only relevant and accurate information is retained.

4. Robust Error Correction: By implementing error handling mechanisms, such as data validation and correction, we minimized the impact of errors on the integrity of the dataset, leading to more reliable insights and decision-making.

CHALLENGES FACED:

1. Data Variety: Dealing with diverse data sources and formats posed challenges in standardizing and harmonizing the data for cleansing purposes. However, Spark's versatility and flexibility helped in addressing these challenges effectively.

2. Computational Complexity: Processing large-scale datasets required careful optimization and resource management to ensure efficient execution and minimize computational overhead.

3. Data Imbalance: Handling missing values and outliers while maintaining the balance and representativeness of the dataset required thoughtful consideration and domain expertise.

IMPLICATIONS:

1. Enhanced Decision-Making: The cleansed dataset serves as a reliable foundation for generating insights, building models, and making data-driven decisions across various domains, including finance, healthcare, and e-commerce.

2. Operational Efficiency: By automating the data cleansing process using Spark and DataFrames, organizations can streamline their data preparation workflows, reducing manual efforts and time-to-insight.

3. Future Opportunities: Continuously improving and refining the data cleansing pipeline will enable organizations to adapt to evolving data challenges and capitalize on emerging opportunities in data analytics and machine learning.

In conclusion, the successful implementation of a data cleansing pipeline using Spark and DataFrames underscores the importance of data quality in driving actionable insights and informed decision-making in real-time data analytics environments.