

Introduction to Big Data

Graded Assignment 3

Name - Avijeet Palit

Roll - 21f1005675

Date -20/10/2024

Problem Statement:

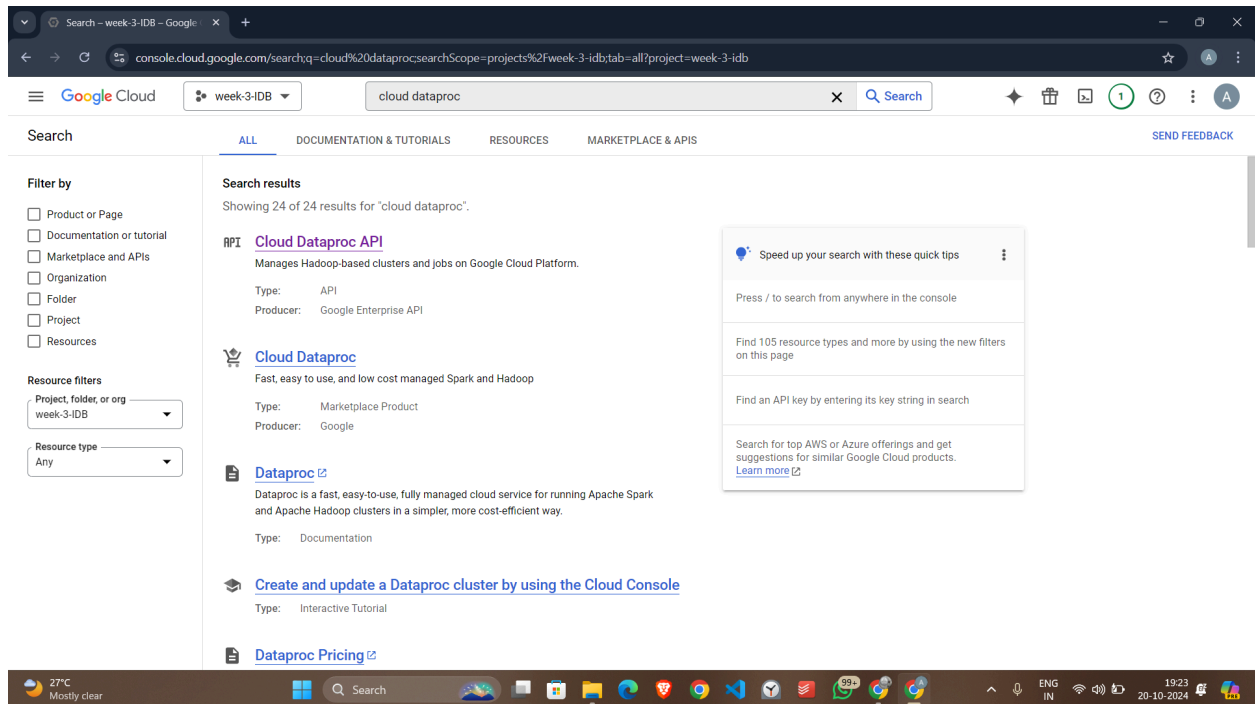
Implement a Spark code to analyze user clicks based on the hashing example discussed in the lecture video. The task involves creating a text file as shown in the video and determining the number of user clicks within specified time intervals: 0-6, 6-12, 12-18, and 18-24.

Approach:

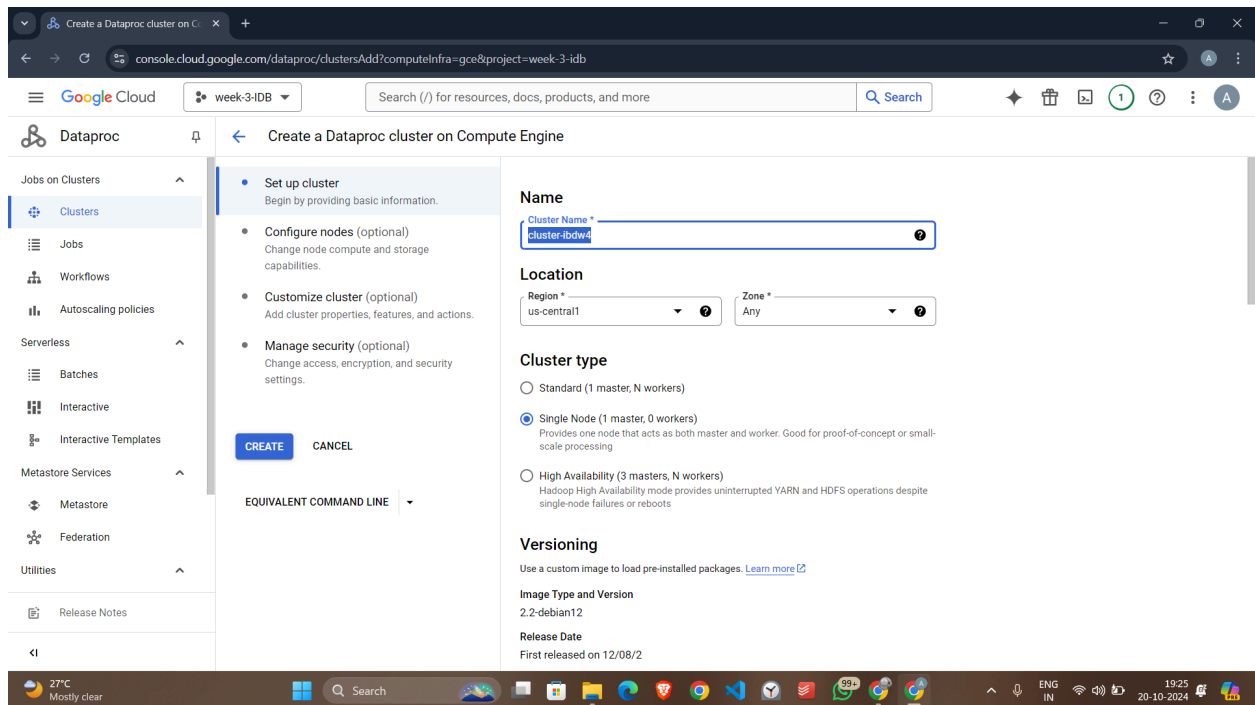
To address the requirements outlined in the problem statement, I developed a Python program to execute Spark code within a Google Cloud Dataproc Cluster. Below are the detailed steps taken:

1. Setting Up the Google Cloud Dataproc Cluster:

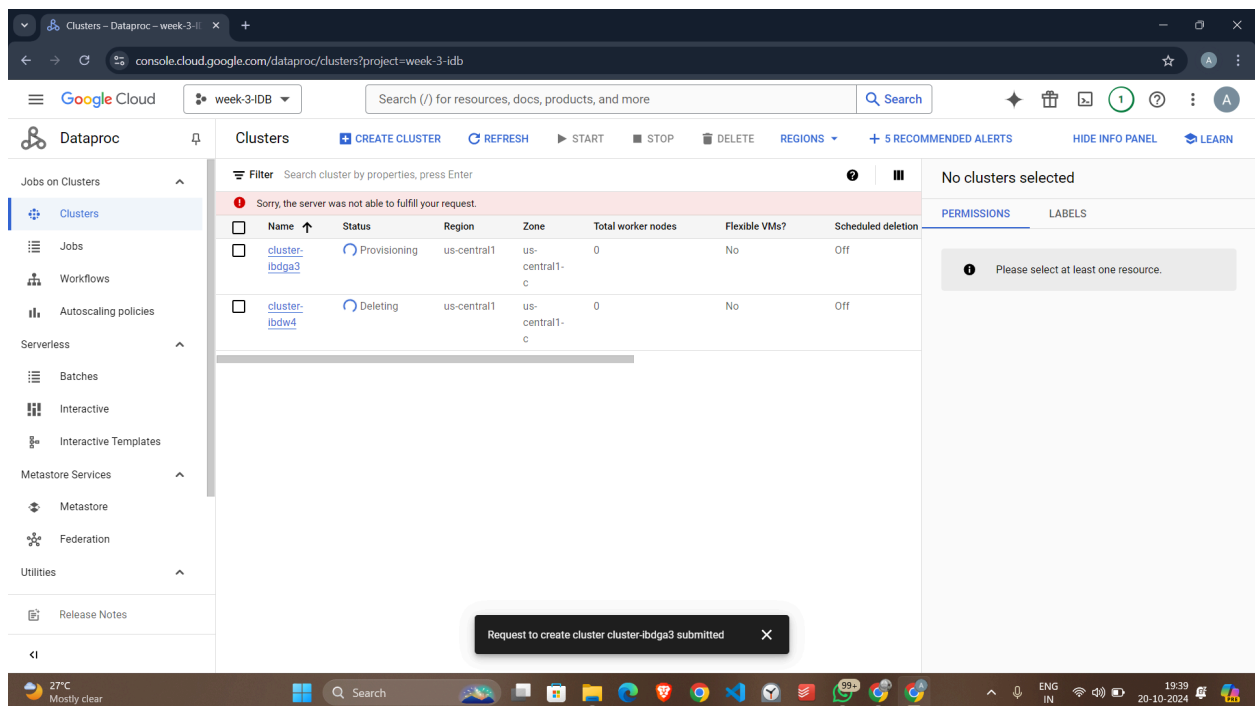
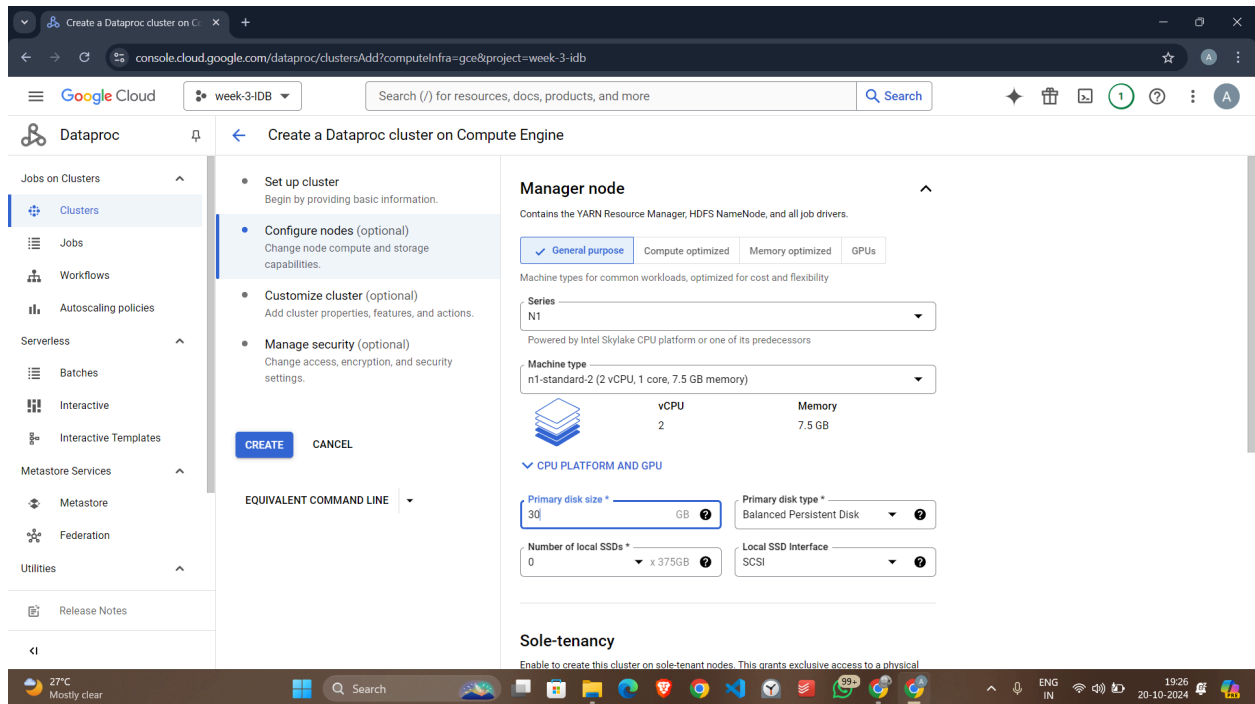
- Enabled the necessary APIs.



- Created a new Google Cloud Dataproc Cluster by navigating to "Dataproc" under the "Cluster" section in the "Analytics" sidebar.
- Named the cluster `cluster-ibdga3`, selected `us-central1` as the region, and chose "Single Node" for the cluster type.



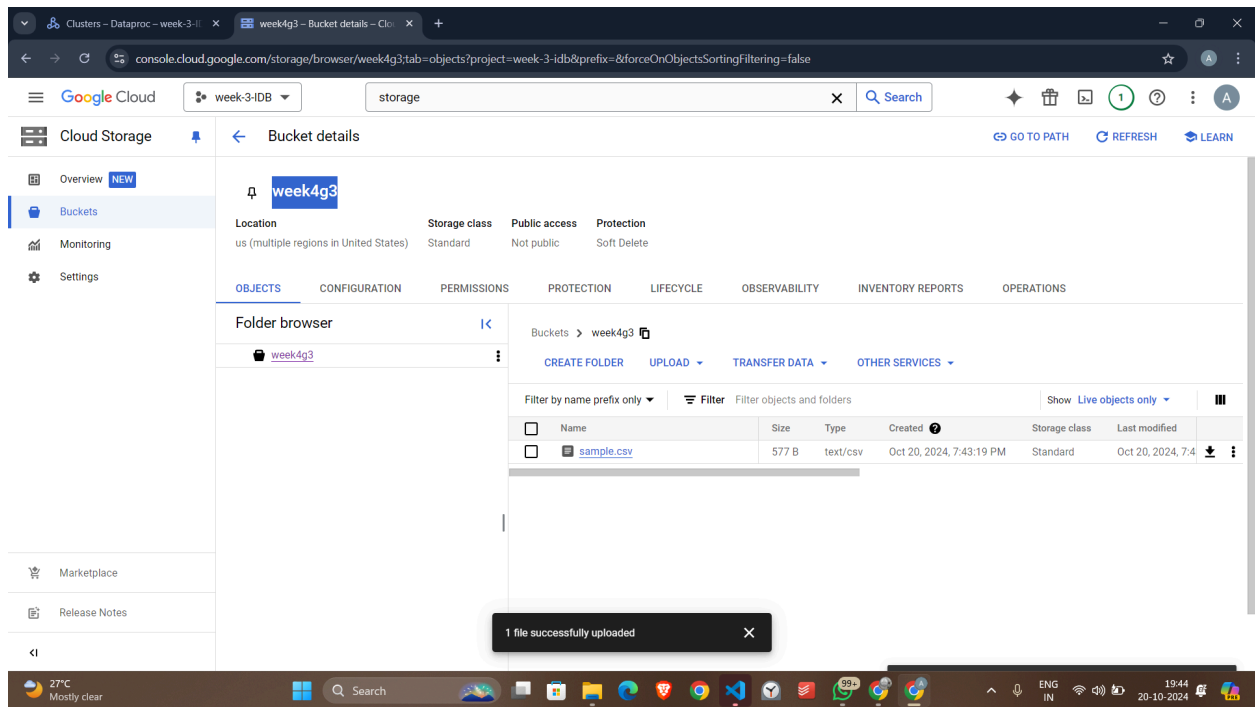
- Configured the Manager Node series as N1 along with the required settings.



2. Creating and Using a Cloud Storage Bucket:

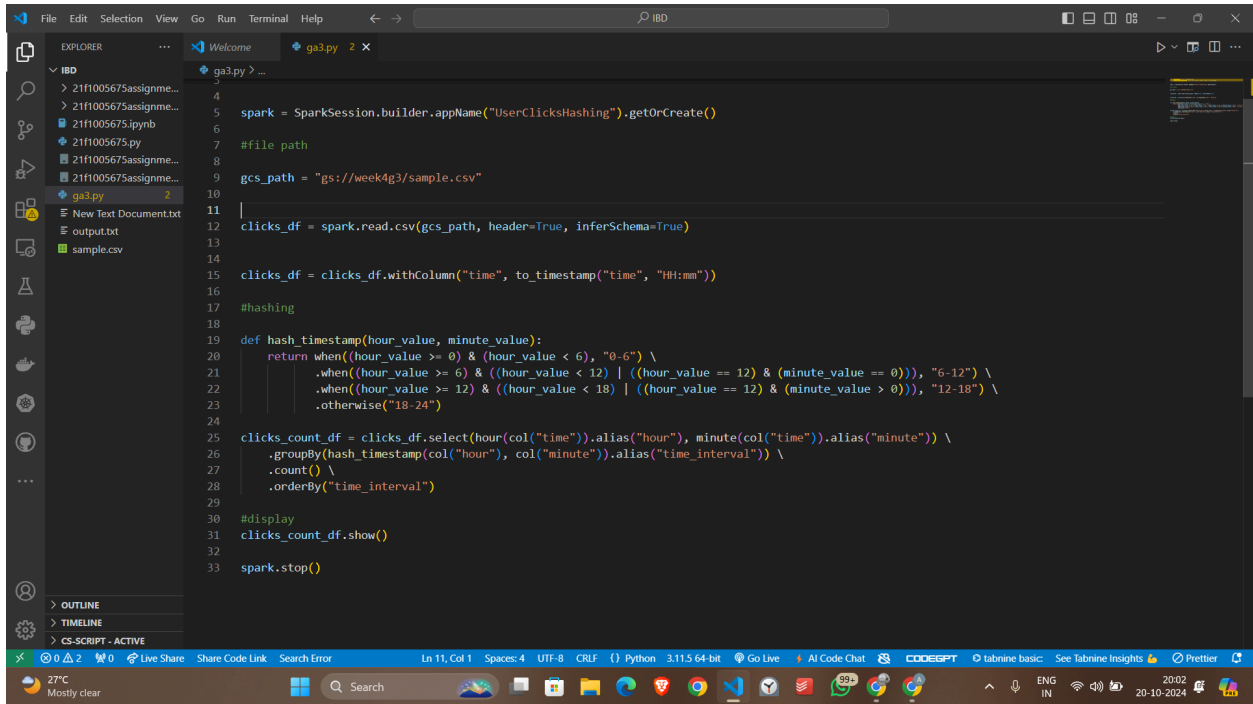
- Created a bucket named **week4g3** on Google Cloud Storage for data management and organization.

- Uploaded the Excel file `sample.csv` to the bucket.

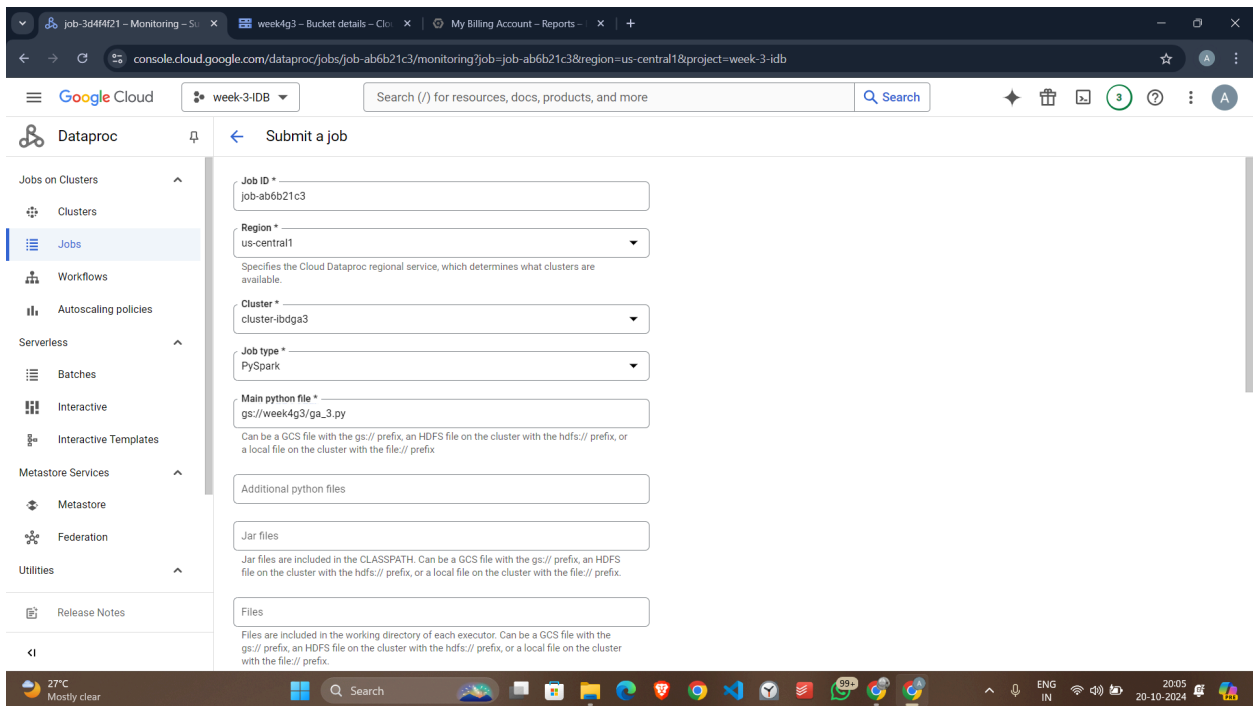


3. Writing and Submitting the PySpark Code:

- Developed the PySpark script `ga.py` using VS Code and saved the file in the bucket.
- Set up a job with dataprop



```
1 spark = SparkSession.builder.appName("UserClicksHashing").getOrCreate()
2
3 #file_path
4
5 gcs_path = "gs://week4g3/sample.csv"
6
7
8
9
10
11
12 clicks_df = spark.read.csv(gcs_path, header=True, inferSchema=True)
13
14
15 clicks_df = clicks_df.withColumn("time", to_timestamp("time", "HH:mm"))
16
17 #hashing
18
19 def hash_timestamp(hour_value, minute_value):
20     return when((hour_value >= 0) & (hour_value < 6), "0-6" \
21               .when((hour_value >= 6) & ((hour_value < 12) | ((hour_value == 12) & (minute_value == 0))), "6-12" \
22               .when((hour_value >= 12) & ((hour_value < 18) | ((hour_value == 12) & (minute_value > 0))), "12-18" \
23               .otherwise("18-24")
24
25 clicks_count_df = clicks_df.select(hour(col("time")).alias("hour"), minute(col("time")).alias("minute")) \
26     .groupBy(hash_timestamp(col("hour"), col("minute")).alias("time_interval")) \
27     .count() \
28     .orderBy("time_interval")
29
30 #display
31 clicks_count_df.show()
32
33 spark.stop()
```



Google Cloud | week-3-IDB | Search (/) for resources, docs, products, and more

Dataproc | Submit a job

Jobs on Clusters

- Clusters
- Jobs**
- Workflows
- Autoscaling policies

Serverless

- Batches
- Interactive
- Interactive Templates

Metastore Services

- Metastore
- Federation

Utilities

- Release Notes

Job ID *
job-ab6b21c3

Region *
us-central1
Specifies the Cloud Dataproc regional service, which determines what clusters are available.

Cluster *
cluster-ibdga3

Job type *
PySpark

Main python file *
gs://week4g3/ga_3.py
Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Additional python files

Jar files
Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Files
Files are included in the working directory of each executor. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Explanation of the Code:

1. Importing Libraries:

- Imported `SparkSession` and necessary functions from `pyspark.sql.functions` and `pyspark.sql`.

2. Initializing Spark Session:

- Created a Spark session named "UserClicksHashing".

3. Reading Data:

- Loaded the `sample.csv` file from the Google Cloud Storage bucket (`gs://week4g3/sample.csv`) into a DataFrame called `clicks_df`.

4. Parsing the Time Column and Creating a Hash Function:

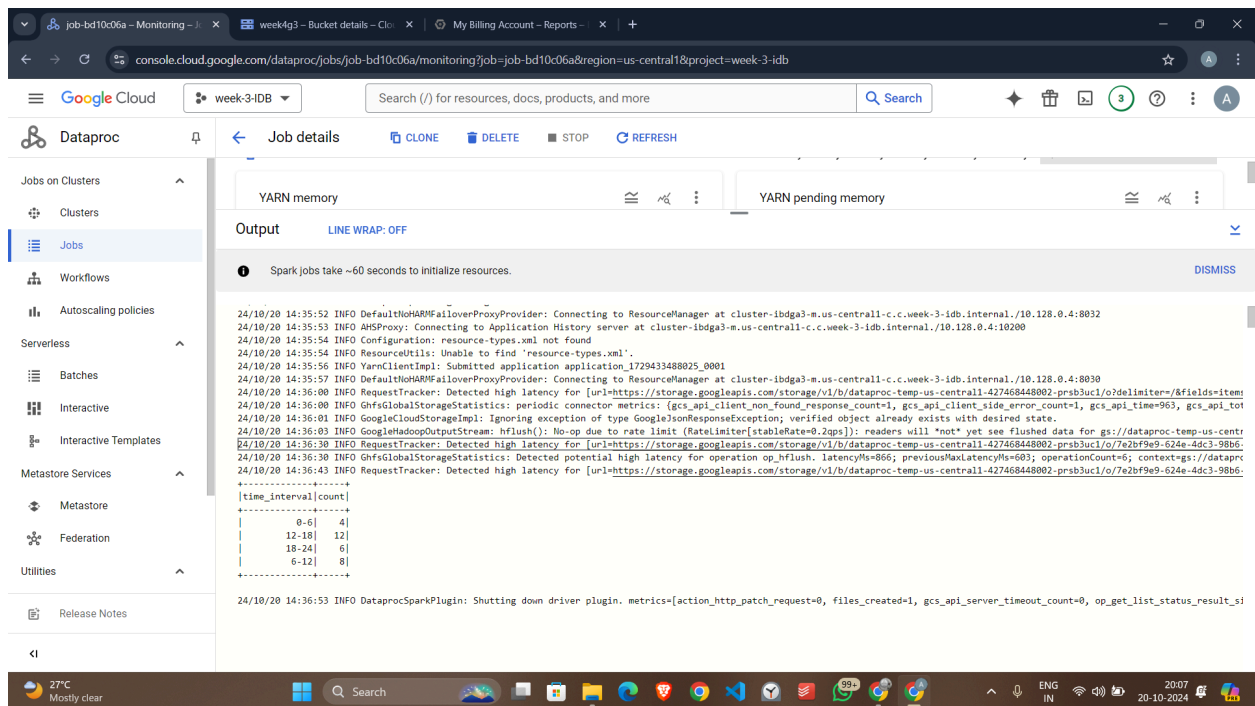
- Parsed the time column from the `sample.csv` file into a `time` timestamp and created a `hash_timestamp` function to hash the time values.

5. Counting Clicks:

- Applied the hashing function to the DataFrame.
- Grouped the data by specified time intervals and counted the number of clicks within each interval.

Results:

The output displayed the number of user clicks for each specified time interval, confirming the correct implementation of the logic. The results were as follows:



The screenshot shows the Google Cloud Dataproc console interface. The left sidebar contains navigation options like Clusters, Jobs, Workflows, and Autoscaling policies. The main area displays 'Job details' for 'week-3-1db'. The 'Output' tab is active, showing a log of Spark job execution. A table summarizes the click counts for different time intervals.

time_interval	count
0-6	4
12-18	12
18-24	6
6-12	8

The counts accurately reflect the number of clicks that occurred within each defined interval.