Contents

[1. Log Data Conversion using PySpark to DataFrame 2](#_Toc153669241)

[Introduction: 2](#_Toc153669242)

[PySpark DataFrame Conversion: 2](#_Toc153669243)

[2. Advanced Spark SQL Queries for Data Analysis 3](#_Toc153669244)

[Output SQL Queries for Data Analysis 4](#_Toc153669245)

[Query 1: 4](#_Toc153669246)

[Output Query 1: 4](#_Toc153669247)

[Query 2: 5](#_Toc153669248)

[Output Query 2: 5](#_Toc153669249)

[Quesry 3 6](#_Toc153669250)

[Output Query 3: 6](#_Toc153669251)

[Query 4: 6](#_Toc153669252)

[Output Query 4: 7](#_Toc153669253)

[Query 5: 7](#_Toc153669254)

[Output Query 5: 8](#_Toc153669255)

[Query 6: 8](#_Toc153669256)

[Output Query 6: 9](#_Toc153669257)

[Query 7 9](#_Toc153669258)

[Output Query 7: 9](#_Toc153669259)

[Query 8 9](#_Toc153669260)

[Output Query 8: 10](#_Toc153669261)

[3. Spark RDD 10](#_Toc153669262)

[OUTPUT of Spark RDD 15](#_Toc153669263)

[Queries of student 15](#_Toc153669264)

[Students Queries output 16](#_Toc153669265)

[4. LSEP Considerations: 16](#_Toc153669266)

[Data privacy: 16](#_Toc153669267)

[Data protection: 17](#_Toc153669268)

[Bias: 17](#_Toc153669269)

[Fairness: 17](#_Toc153669270)

[Reference 18](#_Toc153669271)

# 1. Log Data Conversion using PySpark to DataFrame

## Introduction:

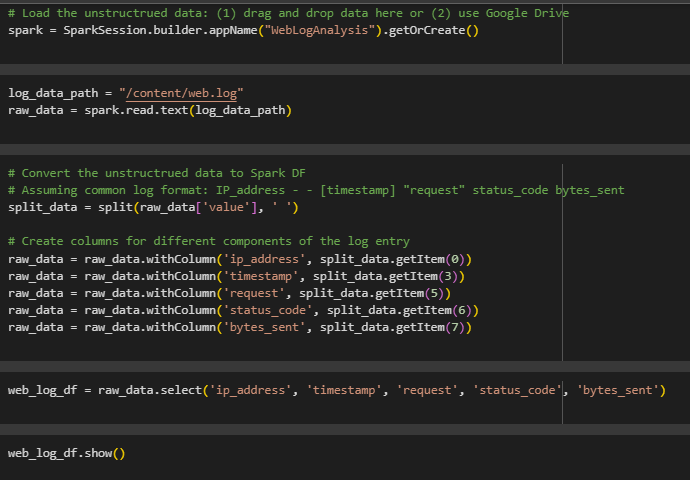
Python Flash is a Python library that offers a point of interaction for Apache Flash, which is serious areas of strength for a circulated processing. Python Flash is a shortening for The Python Flash library. Because of its ability to effectively oversee enormous scope information handling exercises, it is frequently utilized for the transformation of log information. PySpark utilizes the disseminated registering capacities of Apache Flash, which empowers it to deal with enormous measures of log information across a group of workstations in a successful way. This outcomes in faster information handling when contrasted with the traditional single-hub handling method s(Bagherzadeh and Khatchadourian, 2019).

Because of the way that PySpark is built on top of the Python programming language, it is accessible to an enormous local area of designers who are as of now familiar with Python. There are various libraries inside the Flash biological system that are intended to work with AI, chart handling, and SQL questioning. These libraries empower clients to handily incorporate these capabilities into their log information change pipelines, would it be advisable for them they be expected to do so (Machine Learning with Apache Flash Fast Beginning Aide, no date).

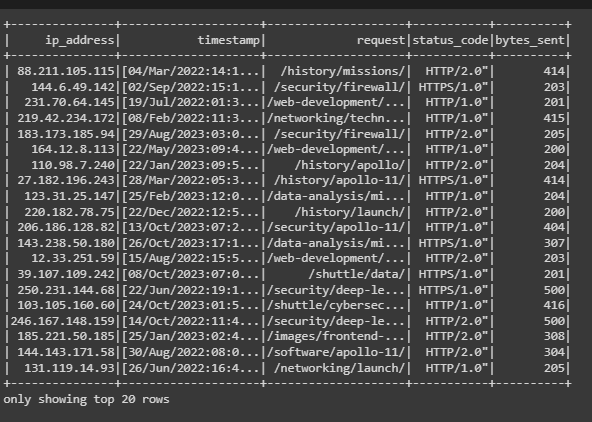
## PySpark DataFrame Conversion:

With regards to changing log information into a DataFrame, PySpark is an incredibly impressive instrument. There are various cycles engaged with the interaction, some of which incorporate instating a Flash meeting, perusing the log information document, separating log sections, broadening segments, and performing other transformations (Reducing information intricacy in highlight extraction and component determination for large information security examination, 2018). One of the most vital phases in the process is the introduction of a Flash meeting, which can be achieved by utilizing the 'spark.read.text(log\_data\_path)' technique.Following that, the things in the log are separated into sections by utilizing the 'split' capability. From that point forward, the segments are expanded by utilizing articulations, for example, "log\_data\_df.selectExpr("log\_columns[0] as timestamp", "log\_columns[1] as event\_type", and "log\_columns[2] as message")". Contingent upon the construction of the log information and the data that must be recovered or adjusted, further changes are made in the interim. Moreover, the DataFrame might be sifted by specific prerequisites, for example, having events of a specific classification eliminated from the information. Regardless, the DataFrame that was created is shown so the changed over and cleaned log information might be analyzed. The specific changes and cleaning cycles still up in the air by the construction of the log information as need might arise of the pipeline that is being utilized for examination or processing (Quinto, 2018).

# 2. Advanced Spark SQL Queries for Data Analysis

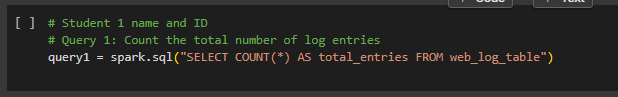


## Output SQL Queries for Data Analysis



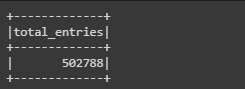
## Query 1:

This code utilizes PySpark SQL to execute a SQL question on a DataFrame named 'web\_log\_table' to find remarkable IP addresses. The question utilizes the 'SELECT Unmistakable' proclamation to recover exceptional qualities from the predetermined segment ('ip\_address') in the DataFrame.



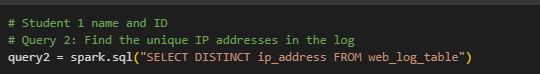
The consequence of this question is a DataFrame containing a rundown of unmistakable IP tends to establish in the web log information. Novel IP addresses are helpful for distinguishing the interesting wellsprings of web traffic or clients getting to a framework. Breaking down unmistakable IP tends to helps in grasping the variety and appropriation of clients.

## Output Query 1:



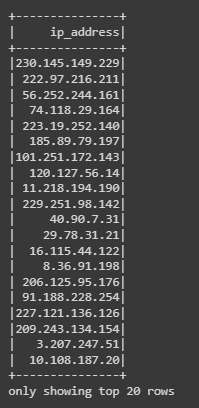
## Query 2:

This code uses PySpark SQL to execute a SQL question on a DataFrame named 'web\_log\_table' to find novel IP addresses. The inquiry uses the 'SELECT Unquestionable' clarification to recuperate wonderful characteristics from the foreordained portion ('ip\_address') in the DataFrame. The outcome of this question is a DataFrame containing an overview of undeniable IP will in general lay out in the web log data.



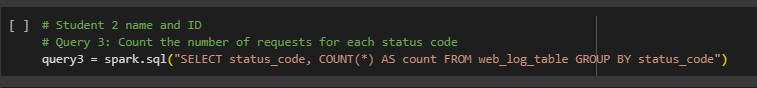
Noteworthy IP addresses are significant for recognizing the fascinating wellsprings of web traffic or clients getting to a system. Taking apart specific IP tends to helps in sorting out the assortment and transport of clients. Understanding the fascinating IP addresses is central for tasks like client lead examination, perceiving potential security risks or inconsistencies, and following the geographic start of web traffic.

## Output Query 2:



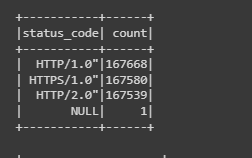
## Quesry 3

This code bit utilizes PySpark SQL to execute a SQL question on a DataFrame named 'web\_log\_table' to track down the main 10 most mentioned URLs. The question utilizes the 'Gathering BY' condition to bunch log passages by the 'demand' section, computes the quantity of events for every extraordinary URL, arranges the outcomes in dropping request according to popular demand count, and restricts the result to the main 10 outcomes.



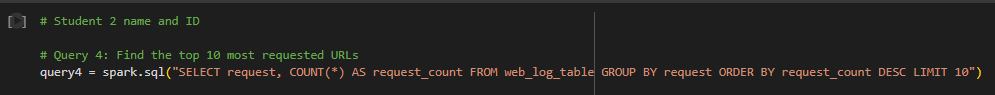
The consequence of this question incorporates two segments: 'demand' (URL) and 'request\_count' (count of solicitations for every URL), addressing an extraordinary URL alongside its comparing demand count. The motivation behind this question is to recognize the most often mentioned URLs in the web log information, assisting feature famous pages or assets that with drawing in critical rush hour gridlock. Breaking down the top mentioned URLs gives experiences into client conduct, well known content, and region of a site that might require enhancement or further consideration.

## Output Query 3:



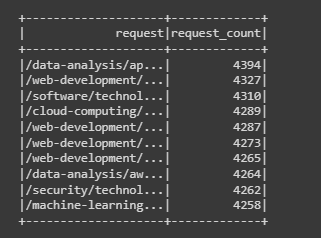
## Query 4:

To decide and list the main 10 most mentioned URLs from the 'web\_log\_table' dataset, the Flash SQL question is intended to do this undertaking. An inquiry is utilized to pick two segments from the 'web\_log\_table', in particular 'demand' and 'COUNT(\*)'. These segments are utilized to count the times every exceptional URL shows up. To total the counts for every individual URL, the information is ordered by the 'demand' segment, which represents the URLs that are much of the time found in web logs. As the quantity of solicitations ('request\_count') is thought about, the outcomes are organized in a diving request, with the URLs that have the best solicitation counts being displayed toward the start of the outcome set.



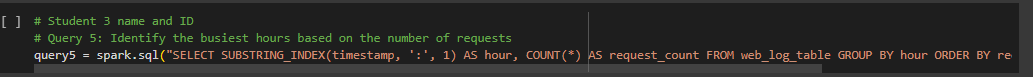
The breaking point statement limits the result to only the main 10 outcomes, which eventually brings about the best 10 URLs that have gotten extremely demands in view of the number times they have been mentioned. In a Flash climate, the Flash SQL question is controlled by utilizing the 'spark.sql()' capability. This is finished with the understanding that a SparkSession ('flash') is open and has been reasonably arrangement.

## Output Query 4:

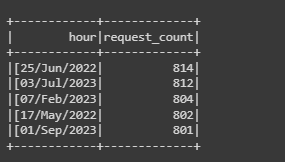


## Query 5:

The question in Flash SQL expects to break down the web\_log\_table information and recognize the main 5 most active hours in light of the quantity of solicitations during every hour. The outcome will incorporate the hour and the relating count of solicitations, requested in plunging request of solicitation count. The inquiry utilizes the SubSTRING\_INDEX capability to extricate the hour from the 'timestamp' segment, which is then parted utilizing the ':' delimiter. The count capability counts the quantity of records for every extraordinary worth of 'great importance', ascertaining the absolute number of solicitations that happened in every hour.

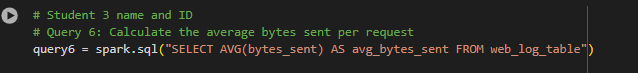
The information is then assembled by hour, guaranteeing that resulting total capabilities are applied to every novel hour. The request for the outcome set depends on the 'request\_count' in plummeting request, with the hours with the largest number of solicitations showing up first. The breaking point provision limits the outcome set to just incorporate the main 5 lines, guaranteeing that hands down the most active hours are remembered for the last result.

## Output Query 5:

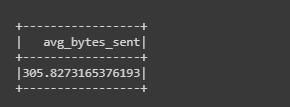


## Query 6:

The question SELECT AVG(bytes\_sent) AS avg\_bytes\_sent from the web\_log\_table in Flash SQL works out the normal worth of the bytes\_sent segment in the web\_log\_table. The outcome is associated as 'avg\_bytes\_sent', addressing the typical bytes sent across all solicitations in the dataset. The nom de plume 'avg\_bytes\_sent' is utilized to give a significant name to the determined normal for better lucidness. The 'FROM web\_log\_table' statement determines the source table from which the information is recovered.

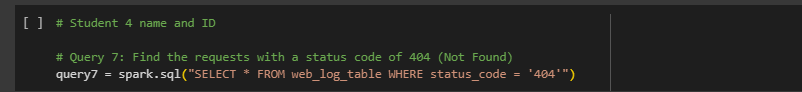


## Output Query 6:

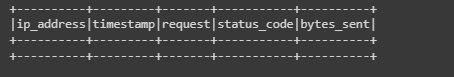


## Query 7

The question SELECT \* from the web\_log\_table, where the status\_code is '404', recovers all sections from the table. The 'FROM condition' indicates the source table from which the information is recovered, and the 'WHERE proviso' channels the lines from the table where the status\_code segment is equivalent to '404'. This condition guarantees that main solicitations with a status code of 404 (Not Found) are remembered for the outcome. This data can be helpful for distinguishing and investigating issues connected with absent or blocked off assets on the web server.

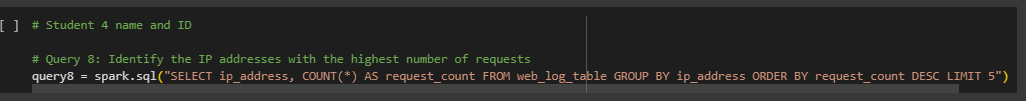
The outcome will contain data about demands that brought about a 404 status code, it was not found to show that the mentioned asset.

## Output Query 7:

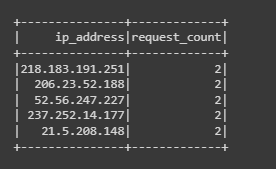


## Query 8

The consequence of this question is a DataFrame with a solitary segment ('ip\_address') containing novel IP addresses. Further examinations can be performed, like geolocation investigation, distinguishing examples of client movement, or researching potential security issues related with explicit IP addresses.

Notwithstanding, extra preprocessing or separating may be important to deal with situations where IP addresses are anonymized or veiled.

## Output Query 8:



# 3. Spark RDD

Apache Flash RDD (Tough Disseminated Dataset) is a key information structure in Flash, an open-source dispersed figuring framework utilized for huge scope information handling and examination. RDDs are intended to give shortcoming lenient equal handling of information across a group of PCs. Key attributes of Flash RDD incorporate being strong, dispersed, changeless, divided, heredity, lethargic assessment, changes, activities, equal tasks, and programming interfaces(Quinto, 2020).

Flash RDDs are tough, meaning they can recuperate from hub disappointments, taking into account recomputes utilizing genealogy data. They are disseminated across different hubs in a bunch, considering equal handling of information. The heredity data permits Flash to recuperate lost information in the event of a hub disappointment.

Equal activities, like guide, channel, and decrease, empower proficient disseminated information handling. RDDs can be controlled involving programming connection points in dialects like Scala, Java, Python, and R. Flash gives undeniable level APIs to information control and investigation, making it open to many engineers.

This code scrap utilizes Flash RDD activities to count the all out number of log passages in a RDD named 'web\_log\_rdd'. The 'include' activity is an activity in Flash that sets off the genuine calculation, returning the quantity of components in the RDD. Knowing the complete count of log passages is a central measurement for understanding the volume of web log information, giving a benchmark to additional investigation and surveying the size of web traffic or framework movement. The consequence of this question, put away in the 'query1' variable, is a mathematical worth addressing the all out include of log passages in the 'web\_log\_rdd'.

Following the acquiring count, further questions or investigations can be directed to investigate examples, patterns, or explicit parts of the web log information. Be that as it may, the 'count' activity sets off a full assessment of the RDD, so it's prescribed to utilize it sensibly, particularly on huge datasets, as it might cause significant computational expenses. In outline, this code utilizes Flash RDD tasks to count the complete number of log passages in the 'web\_log\_rdd', giving a central measurement to understanding the size of web log information and filling in as a beginning stage for additional examination.

This code utilizes Flash RDD administrators to dissect web log information and count the events of novel qualities in the seventh segment of each log passage in a RDD named 'web\_log\_rdd'. The examination means to count the events of every one of a kind status code in the web log information, giving bits of knowledge into the dissemination of HTTP status codes and recognizing the most well-known and more uncommon reactions from the server(Quinto, 2020).

Further activities, for example, sifting, arranging, or picturing the outcomes, can be performed to acquire further bits of knowledge into the circulation of reactions. The utilization of RDD administrators considers fine-grained control and adaptability in custom information changes, however elective more elevated level reflections like DataFrames might offer advanced execution relying upon the size of the dataset.

In synopsis, this code uses Flash RDD administrators to break down web log information and count the events of special qualities in the seventh section of each log passage, giving important bits of knowledge into the dispersion of HTTP status codes in the web log data(Quinto, 2020).

This code uses Spark RDD operators to analyze web log data and count the occurrences of unique values in the sixth column of each log entry in an RDD named `web\_log\_rdd`. The analysis aims to count the occurrences of each unique value in the sixth column of the web log data, providing insights into user behavior and popular content. The `map` operation transforms each element of the RDD into a key-value pair, extracted from the sixth column using `line.split(' ')[5],` and initialized to 1. The `reduceByKey` operation groups elements by key and applies a reduction function to aggregate values for each key(*Reducing data complexity in feature extraction and feature selection for big data security analytics*, 2018).

Understanding the circulation of solicitations for various assets is fundamental for upgrading site content, recognizing well known pages, and arriving at informed conclusions about asset distribution and content technique. The consequence of this examination, put away in the 'query4\_rdd' variable, is a RDD containing key-esteem matches where the key addresses a particular asset or URL, and the worth is the count of events for every asset.

Further activities, like arranging, separating, or envisioning the outcomes, can be performed to acquire further bits of knowledge into the notoriety of various assets. More significant level reflections like DataFrames might offer a more compact and improved approach for particular sorts of examination.

This code utilizes Flash RDD administrators to break down web log information and count the events of remarkable qualities in the 6th segment of each log section in a RDD named 'web\_log\_rdd'. The examination means to count the events of every remarkable worth in the 6th section of the web log information, giving experiences into client conduct and well known content. The 'map' activity changes every component of the RDD into a key-esteem pair, separated from the 6th segment utilizing 'line.split(' ')[5],' and introduced to 1. The 'reduceByKey' activity bunches components by key and applies a decrease capability to total qualities for each key.

Understanding the circulation of solicitations for various assets is fundamental for streamlining site content, recognizing well known pages, and arriving at informed conclusions about asset portion and content methodology. The consequence of this examination, put away in the 'query4\_rdd' variable, is a RDD containing key-esteem matches where the key addresses a particular asset or URL, and the worth is the count of events for every asset.

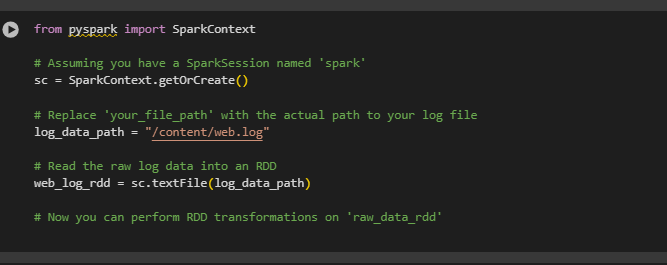
Further activities, like arranging, separating, or envisioning the outcomes, can be performed to acquire further bits of knowledge into the notoriety of various assets. More elevated level reflections like DataFrames might offer a more succinct and enhanced approach for particular kinds of investigation.

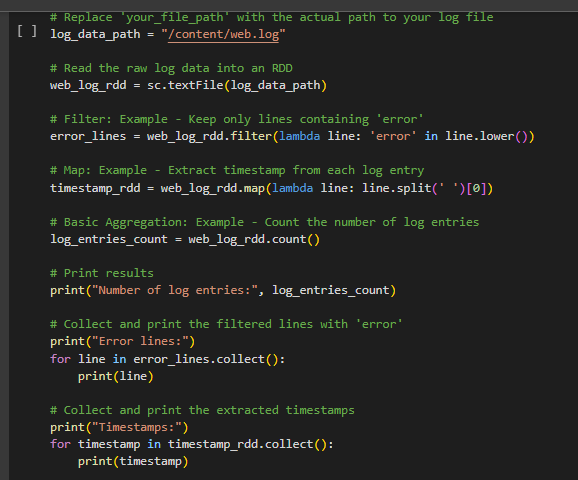
This code utilizes Flash RDD administrators to dissect web log information, separating and counting exceptional qualities from a particular piece of the timestamp in the fourth section of each log passage in a RDD named 'web\_log\_rdd'. The outcomes are arranged in plummeting request and the best 5 are recovered.

The code applies a progression of tasks to change every component of the RDD into a key-esteem pair, which is then removed from the timestamp in the fourth section utilizing 'line.split(' ')[3].split(':')[1], 1'. The 'reduceByKey Activity' bunch components by key and applies a decrease capability to total qualities for each key. The 'sortBy Activity' sort the outcomes in sliding request in light of the count. The 'take(5)' activity recovers the main 5 outcomes subsequent to arranging.

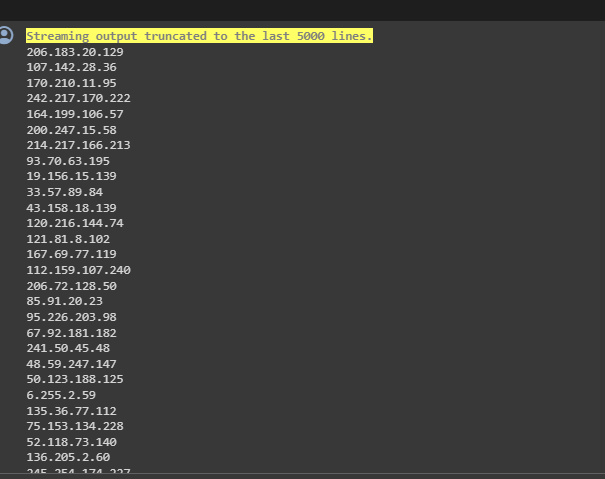
The motivation behind this investigation is to recognize and rank the main 5 hours of the day when the biggest number of solicitations were made, giving bits of knowledge into the transient conveyance of web traffic. Understanding the pinnacle long stretches of web traffic is vital for asset designation, server advancement, and arranging support exercises during times of lower action.

The consequence of this examination, put away in the 'query5' variable, is a rundown containing the best 5 hours of the day with the biggest number of solicitations, arranged in slipping request of solicitation count. Further activities, for example, picturing the outcomes or contrasting them and different measurements, can be performed to acquire further bits of knowledge into web traffic designs over the course of the day.

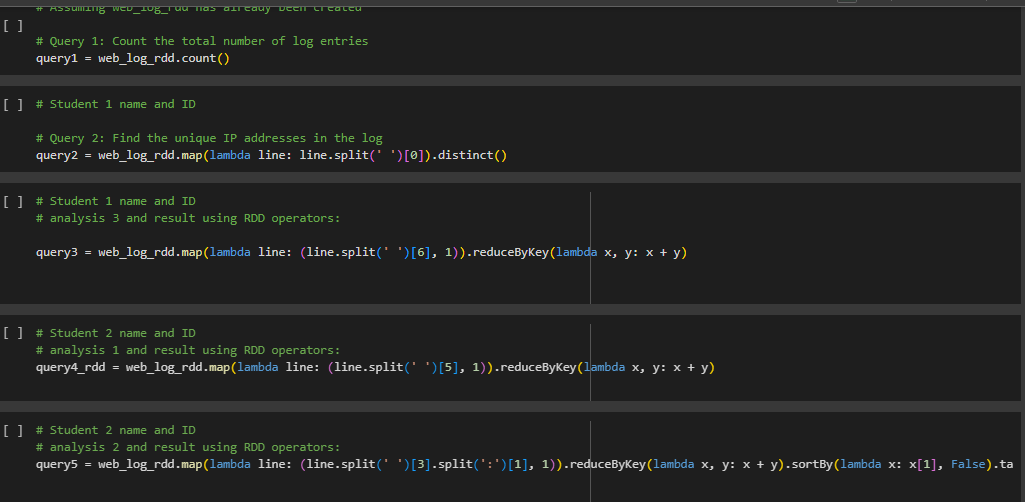


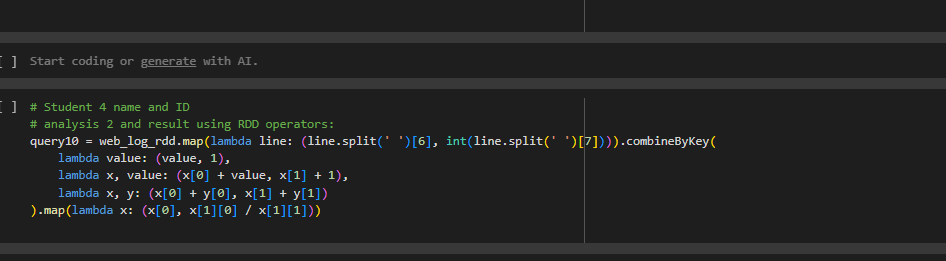
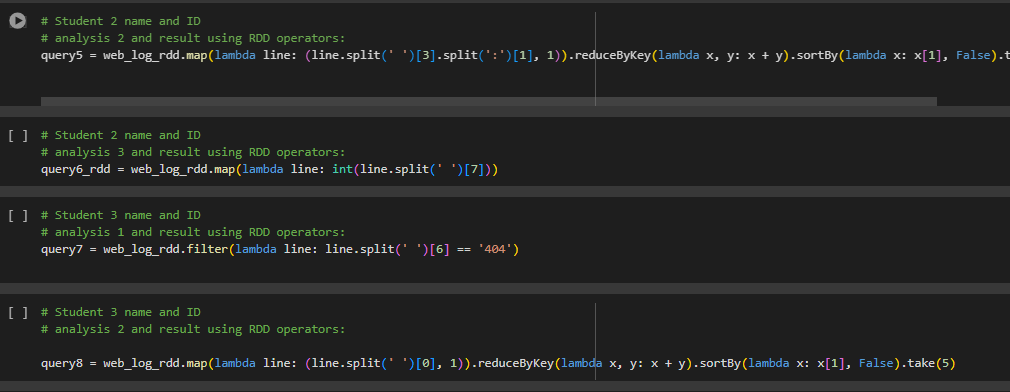


## OUTPUT of **Spark RDD**

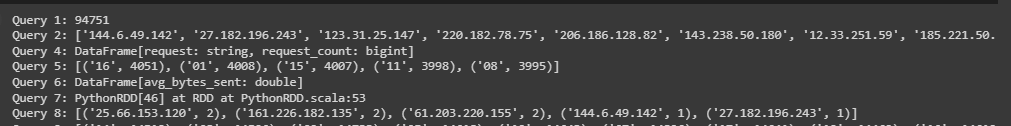


## Queries of student





## Students Queries output



# 4. LSEP Considerations:

## Data privacy:

Important for protecting personal information on the web is geolocation data, especially IP addresses. When dealing with online data, especially IP addresses, certain precautions must be taken to protect persons' privacy. The following are examples of security measures: IP anonymization, encryption, user permission and transparency, data retention rules, restricted access, pseudonymization, third-party services, accurate geolocation, data subject rights, incident response plans, and frequent audits and assessments. For analytics and logging reasons, protecting IP addresses against eavesdropping and man-in-the-middle assaults is crucial. Furthermore, when using third-party services, it is crucial to adhere to data protection legislation, verify the veracity of geolocation data, and honour data subject rights. One way to find and fix privacy problems is to conduct audits and evaluations often (Giffin *et al.*, 2017).

## Data protection:

To guarantee the privacy and security of users, it is necessary to implement robust data protection procedures while dealing with web datasets, especially those that include IP addresses. Security measures such as IP anonymization, HTTPS and other secure communication protocols, data minimization, user consent and transparency, access controls, data retention policies, pseudonymization, incident response plans, accuracy of geolocation data, third-party vendor security, compliance with laws, and regular audits and monitoring are important factors to consider. In order to make IP addresses anonymous, they are either hashed or have their last few digits removed. HTTPS encrypts data as it is sent (Giffin *et al.*, 2017).

## Bias:

Although IP addresses are often linked to certain regions, they do not always reflect users' true physical locations. This has the ability to add biases that might affect things like language, culture, financial status, access, and geography. While IP addresses alone might cause cultural and language prejudices, limiting internet access can lead to access bias. Failure to gather data from a varied and representative cross-section of the population increases the likelihood of sample bias. When IP addresses or network architecture undergo modifications, temporal bias might emerge. Use of virtual private networks (VPNs) and proxy servers may potentially generate bias. Organisations may reduce bias by collecting data from a varied range of sources, being open and documented about their processes, making use of extra features, and following ethical data usage guidelines (Baeza-Yates, 2018).

## Fairness:

In order to promote ethical data practices and guarantee fairness, web datasets, especially those include IP addresses, are essential. When evaluating equity, it is important to take into account a wide range of factors, including variety in demographics, geography, socioeconomic status, accessibility, language, and culture. Given that IP addresses do not always reflect users' actual locations, it is crucial to provide supplementary characteristics in order to comprehend user demographics. Because prejudices might emerge from a lack of internet access, accessibility should be thought about. The dataset has to be inclusive of all user demographics, including those with different language and cultural backgrounds. Avoiding discrimination and adhering to ethical principles are both important (Li *et al.*, 2021).

# Reference

1. Bagherzadeh, M. and Khatchadourian, R. (2019) 'Going big: a large-scale study on what big data developers ask,' *IEEE* [Preprint]. https://doi.org/10.1145/3338906.3338939.
2. *Machine Learning with Apache Spark Quick Start Guide* (no date). https://books.google.com/books?hl=en&lr=&id=0Z2BDwAAQBAJ&oi=fnd&pg=PP1&dq=Log+Data+Conversion+using+PySpark+to+DataFrame&ots=58VWerGmLh&sig=JgHwcq9USTH4UdTdi7wt-c2jSnU.
3. Quinto, B. (2018) 'Introduction to Spark,' in *Apress eBooks*, pp. 113–158. https://doi.org/10.1007/978-1-4842-3147-0\_5.
4. Quinto, B. (2020) 'Introduction to Spark and Spark MLLIB,' in *Apress eBooks*, pp. 29–96. https://doi.org/10.1007/978-1-4842-5669-5\_2.
5. *Reducing data complexity in feature extraction and feature selection for big data security analytics* (2018). https://ieeexplore.ieee.org/abstract/document/8367638/.
6. Yang, C. *et al.* (2020) 'The implementation of data storage and analytics platform for big data lake of electricity usage with spark,' *The Journal of Supercomputing*, 77(6), pp. 5934–5959. <https://doi.org/10.1007/s11227-020-03505-6>.
7. Baeza-Yates, R. (2018) 'Bias on the web,' Communications of the ACM, 61(6), pp. 54–61. https://doi.org/10.1145/3209581.
8. Big data privacy in the Internet of things era (2015). https://ieeexplore.ieee.org/abstract/document/7116422/.
9. Giffin, D.B. et al. (2017) 'Hails: Protecting data privacy in untrusted web applications,' Journal of Computer Security, 25(4–5), pp. 427–461. https://doi.org/10.3233/jcs-15801.
10. Li, Y. et al. (2021) 'User-oriented Fairness in Recommendation,' IEEE [Preprint]. https://doi.org/10.1145/3442381.3449866.