RV UNIVERSITY, BENGALURU

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A Project Report On

Analysis of Clash of Clans player communities

B.SC(Honors)

In

School of Computer Science and Engineering

Submitted By

Team Member 01: 1RVU22BSC069 _ PRABHAS BHAT
Team Member 02: 1RVU22BSC072 _ PRASANNA G
Team Member 03: 1RVU22BSC052 _ MERSHIKA U
Team Member 04: 1RVU22BSC021 _ JYOSTHNA

Big Data Analytics
Under the Guidance of
Vinod Kumar Raju
Assistant Professor
School of CSE
RV University, Bengaluru-
560059 2024-2025

RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work titled "Analysis of Clash of Clans player communities" is carried out by **PRABHAS BHAT** (1RVU22BSC069), **PRASANNA G** (1RVU22BSC072), **MERSHIKA U** (1RVU22BSC052), **JYOSTHNA** (1RVU22BSC021) RV University, Bengaluru, **B.sc (Hons) in the School of Computer Science and Engineering** of the RV University, Bengaluru during the year 2025-2026. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the project report. The Project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution.

Signature of Guide

External Viva:

Name of Examiners	**Signature with Date**
1	
2	

RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Prabhas Bhat, Prasanna G, Mershika U, Jyosthna students of second semester BSc(Hons), SoCSE, RV University, Bengaluru, hereby declare that the project titled '**Analysis of Clash of Clans player communities**' has been carried out by us and submitted in partial fulfillment of **Bachelor of science(Hons)** in **School of Computer Science and Engineering** during the year 2025-26.

Further, we declare that the content of the report has not been submitted previously by anybody or to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RV University will be the property of RV University, Bengaluru, and we will be one of the authors of the same.

Place: Bengaluru
Date:

Name	**Signature**
1. **PRABHAS BHAT** (1RVU22BSC069)	
2. **PRASANNA G** (1RVU22BSC072)	
3. **MERSHIKA U** (1RVU22BSC052)	
4. **JYOSTHNA** (1RVU22BSC021)	

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our bachelor's degree in this institution.

A special thanks to our Program Director, **Dr. Sudhakar, and** Dean - **Dr.Shobha G,** for their continuous support and providing the necessary facilities with guidance to carry out mini project work.

We would like to thank our guide, Prof**,** Vinod Kumar Raju **Assistant Professor**, School of Computer Science and Engineering, RV University, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project

work.

Signature of Student

USN:

Name:

Abstract

This project is a case study on a Clash of Clans dataset to analyse the distribution of clan types, and behaviour of each ranked league. We compared execution times for a single-node versus multi-node system, using a multi-node Apache Spark configuration on Google Cloud Platform with a single-node Apache Hive setup on a virtual machine. After running the same aggregation tasks on both systems for a considerable amount of clan data of 3.56 million rows, we came to the conclusion that a multi-node Spark is far more efficient than a single-node Hive setup, at least for such gaming big data analytics.

This study also highlights the importance of distributed computing frameworks in handling large-scale gaming datasets efficiently. By leveraging parallel processing and in-memory computation, Apache Spark demonstrates significant performance improvements over traditional single-node systems. The findings of this case study can serve as a reference for selecting suitable big data technologies for analytics-intensive applications in the gaming domain.

Table of Contents:

List of Tables

List of Figures

1. Introduction

1.1 General Introduction

The recent increase in online multiplayer games has led to the generation of vast amounts of data concerning user behaviour, engagement, and interactions. Clash of Clans is one such globally recognized mobile game with millions of user bases divided into clans. The clans are diverse in terms of structure, accessibility, and level of competition, providing a superb platform to analyze large-scale interactions within a community via big data solutions.

In this particular project, big data analytics is used to analyze the data involving clans in Clash of Clans. The use of distributed computing platforms helps in gaining insights into types, leagues, and overall engagement.

1.2 Literature survey

The relevance of big data platforms such as Apache Spark and Hive has been illustrated in a couple of research papers to efficiently process large chunks of data. Though Spark is known for its in-memory computation and parallelization properties, which make it apt for handling complex tasks, Hive is known for supporting SQL-type query processing on large chunks of data, but it is mostly limited by the disk execution environment when used on a single-node environment. The support research on gaming analytics focuses on developing scalable infrastructure that supports real-time analysis of game-related metrics such as player engagement, matchmaking, and competition balance.

1.3 Problem Statement & Objectives

Problem Statement:

The traditional single node data processing systems pose some challenges in terms of performance when dealing with massive gaming data. It is essential to assess whether the use of distributed computing platforms gives a significant advantage in such analytics tasks.

Objectives:

- The analysis of the distribution of types in the Clash of Clans community.
- To analyze clan-related information in a league, including points, levels, and membership.
- The aim is to compare the performance of PySpark (Multi-node cluster using Google Cloud) with Hive (Single-node Cloudera VM).

2.System Design

2.1 Architectural Diagram

The architecture of the system comprises two parallel environments for processing:

- Spark on GCP: A multi-node cluster supporting Google Cloud Storage for storing the data with Spark for handling.

  Multi-node cluster configuration created on GCP:

  Cluster specifications:

  Master node : n1 standard-4 (4 vCPUs, 15 GB memory), 100 GB harddisk

  Worker Node: n1 standard 2 (2 vCPUs, 7.5 GB memory)

  Supports Jupyter Lab with PySpark

- Hive on CloudEra VM: Single-node virtual machine with Apache Hive for SQL-based analytics.

  CloudEra VM specifications:

  Processor 1, Memory 5GB ,Hard Disk 64 GB

In summary, both environments ingest the same CSV file and run the same aggregation tasks.

2.2 Data Model / Schema Description

The dataset consists of 27 attributes, including clan tag, clan name, clan type, clan war league, clan points, clan level, number of members, and war statistics. Aggregations are primarily performed on clan_type and clan_war_league fields.

3.Software Requirements

3.1 Functional Requirements

- Load large CSV datasets into Spark and Hive environments.
- Perform aggregation and grouping operations on clan data.
- Measure and record execution time for each operation.
- Display and store analytical results.

3.2 Non-Functional Requirements

- High performance and scalability.
- Fault tolerance in distributed execution.
- Accuracy and consistency of analytical results.

3.3 Hardware Requirements

- Multi-node cluster on Google Cloud Platform.
- Virtual Machine with sufficient CPU, RAM, and storage.

3.4 Software Requirements

- Apache Spark
- Apache Hive
- Python (PySpark)
- Google Cloud Platform (GCP)
- Cloudera VM with HIVE, Hadoop installed

3.5 Summary

This chapter outlined the functional and non-functional requirements necessary to support large-scale data analytics using distributed and single-node systems.

4.Implementation and Testing

4.1 Modules

4.1.1 Data Ingestion Module

Loads the Clash of Clans CSV dataset from cloud storage into Spark DataFrames and Hive tables.

4.1.2 Aggregation Module

Performs:

- Simple aggregation on clan_type.
- Complex aggregation on clan_war_league including averages of points, levels, and members.

4.1.3 Performance Measurement Module

Measures execution time using Python's time library to compare Spark and Hive performance.

4.2 Testing

- Validation of row counts across both systems.
- Cross-verification of aggregation results.
- Performance testing using execution time comparisons.

5.Results and Discussion

5.1 Clan Type Distribution Results:

Clan Type	Count
open	2,569,771
inviteOnly	820,839
closed	169,133

Table-1: Clan type aggregation

Most clans are open, indicating a casual and inclusive community structure.

5.2 League-wise Aggregation Results

Unranked clans dominate the dataset, while higher leagues such as Champion and Master show fewer but more competitive clans with higher points, levels, and member counts.

League	Num Clans	Avg Clan Points	Avg Clan Level	Avg Num Members
Bronze League I	29,300	4124.03	3.38	11.50
Bronze League II	9,965	3912.77	3.13	12.52
Bronze League III	3,998	3749.62	2.94	13.54
Champion League I	556	22192.79	20.41	20.19
Champion League II	676	27267.77	20.01	24.20
Champion League III	1,480	30826.17	20.49	27.68
Crystal League I	21,187	31010.17	17.56	35.71
Crystal League II	22,230	25932.40	15.10	32.28
Crystal League III	26,156	21514.61	12.81	29.08
Gold League I	34,396	16801.08	10.60	24.33
Gold League II	49,293	12626.05	8.25	19.62
Gold League III	70,027	9165.39	6.17	15.29
Master League I	2,963	35091.31	21.34	33.04
Master League II	5,863	34971.71	20.60	34.95
Master League III	11,288	33625.49	19.39	35.84
Silver League I	81,990	6925.99	4.93	12.72
Silver League II	77,200	5530.49	4.25	11.45
Silver League III	75,017	4539.81	3.70	10.93
Unranked	3,036,158	1072.62	1.49	2.57

Table-2: Clan league aggregation

5.3 Performance Comparison

The Spark cluster consistently outperformed the single-node Hive environment:

- Up to 5×–10× faster execution times.
- Better scalability for large aggregations.

Execution times comparison:

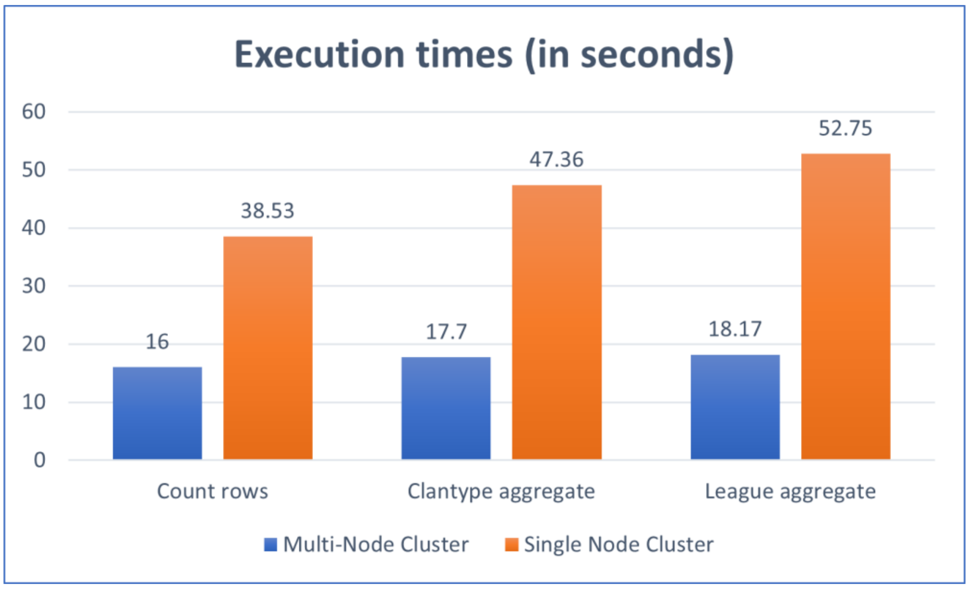


Figure-1: Bar chart of execution times

5.4 Summary

Business Problem perspective: Open clans dominate the ecosystem, highlighting a preference for accessible and casual gameplay communities. In contrast, invite-only and closed clans represent more structured and controlled participation, indicating varied engagement strategies within the player base.

Performance perspective: The row count operation executed **5.5x faster** on the multi-node Spark cluster, while clan type aggregation, which involves a GROUP BY operation, showed a **4x** performance improvement on Spark. The league-wise aggregation, which includes multiple aggregate functions (COUNT, AVG) ran almost **3x faster** on multi-node Spark.

6.Conclusion and Future Work

Conclusion

The results prove that Apache Spark running on a multi-node GCP cluster provided substantial performance benefits over a single-node Hive setup when dealing with big datasets. The Spark cluster benefits from **parallel execution, in-memory computation, and efficient task scheduling**, while the single-node Hive setup is limited by disk I/O and sequential processing.

Future Work

- Including real-time streaming data.
- Applying machine learning models for clan behaviour prediction.
- Extending analysis to player-level engagement metrics.

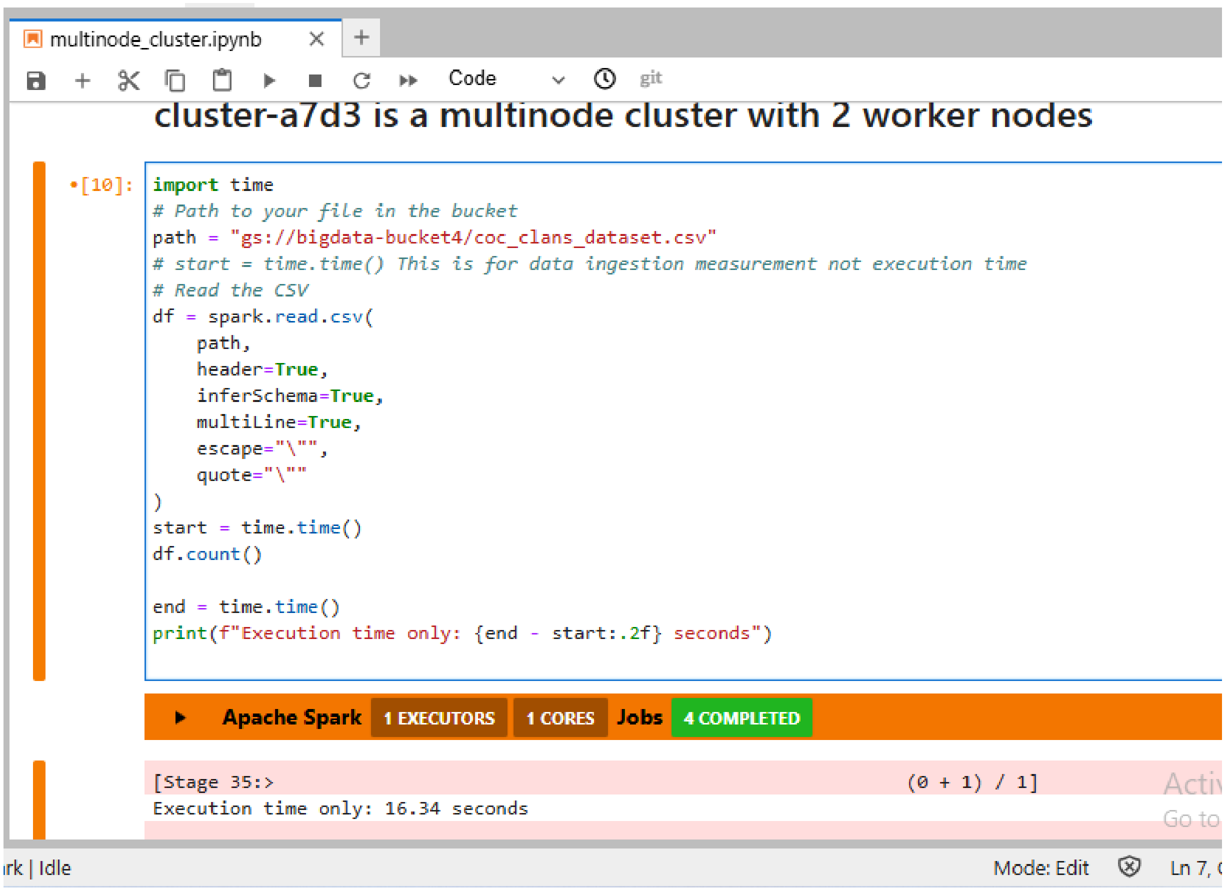
References

- Zaharia, M. et al. (2016). *Apache Spark: A Unified Engine for Big Data Processing*. Communications of the ACM.
- Kaggle. (2023). *Clash of Clans Clans Dataset.*

Appendices

Appendix 1: Screenshots of Multi-node GCP and CloudEra VM operations.

1)Multi-node row count operation

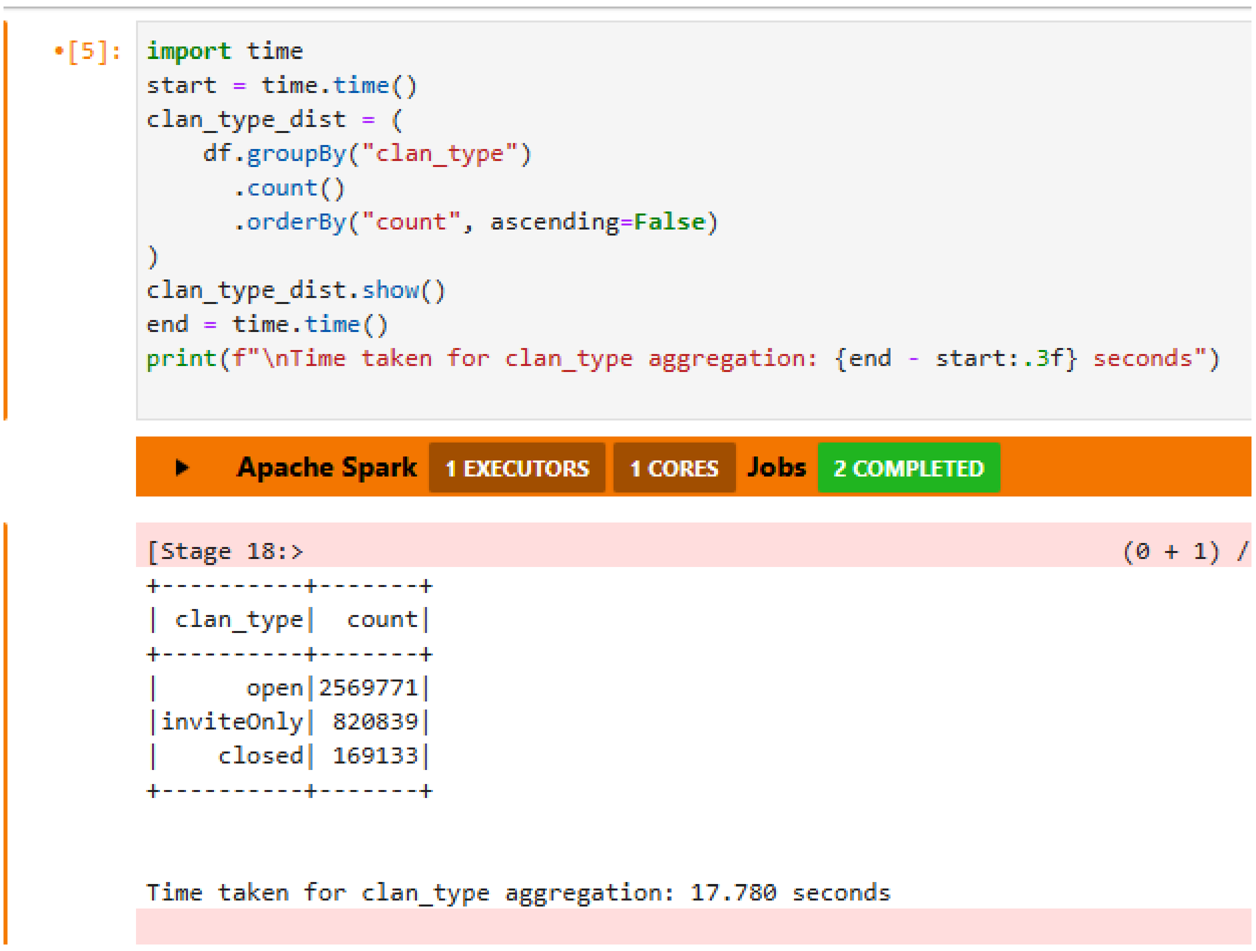


cluster-a7d3 is a multinode cluster with 2 worker nodes

```
[10]:
import time
# Path to your file in the bucket
path = "gs://bigdata-bucket4/coc_clans_dataset.csv"
# start = time.time() This is for data ingestion measurement not execution time
# Read the CSV
df = spark.read.csv(
    path,
    header=True,
    inferSchema=True,
    multiLine=True,
    escape="\"",
    quote="\""
)
start = time.time()
df.count()

end = time.time()
print(f"Execution time only: {end - start:.2f} seconds")
```

```
[Stage 35:>                                                          (0 + 1) / 1]
Execution time only: 16.34 seconds
```

2) Multi-Node clantype aggregation

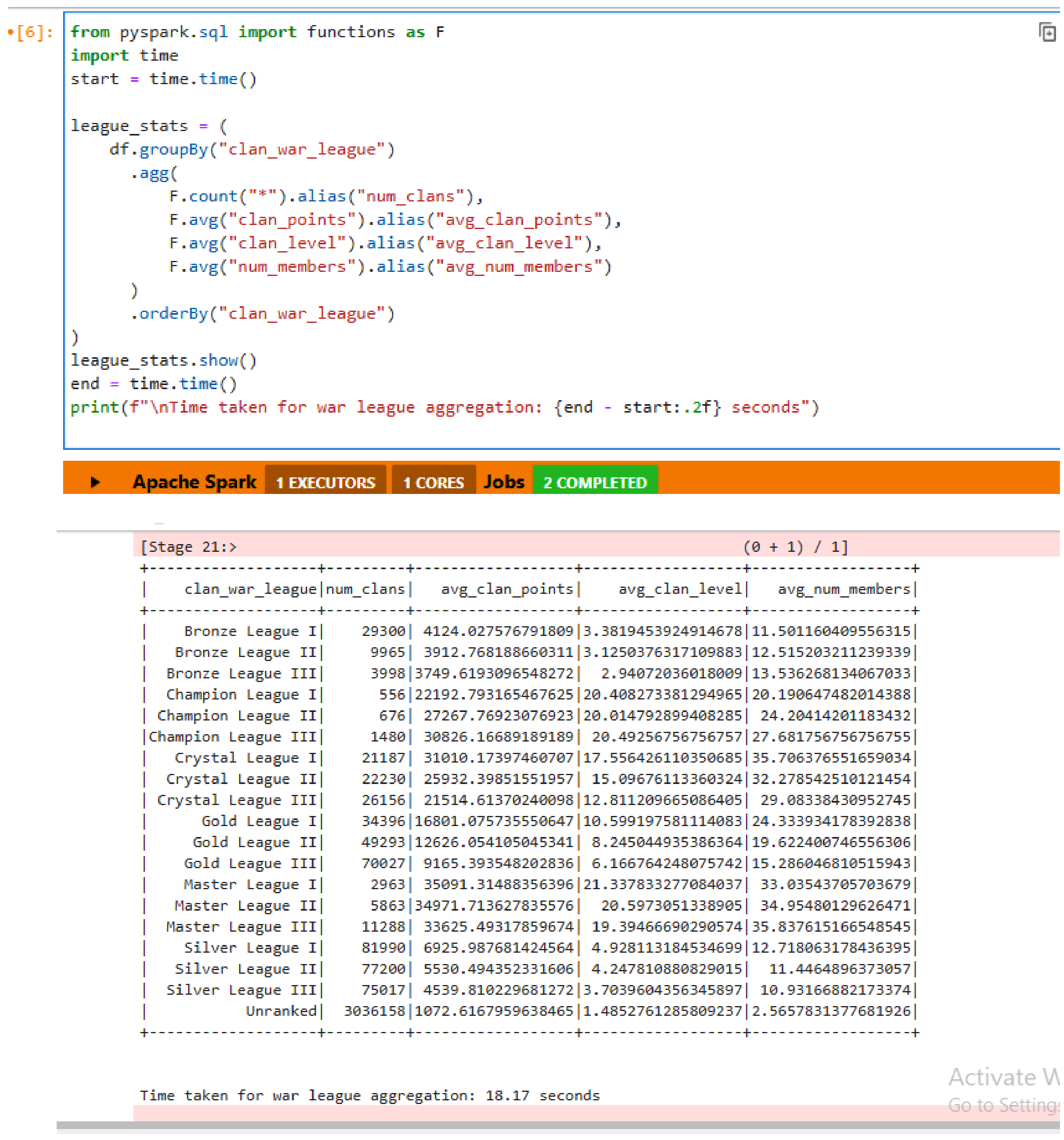


```
•[5]: import time
      start = time.time()
      clan_type_dist = (
          df.groupBy("clan_type")
            .count()
            .orderBy("count", ascending=False)
      )
      clan_type_dist.show()
      end = time.time()
      print(f"\nTime taken for clan_type aggregation: {end - start:.3f} seconds")
```

▶ Apache Spark 1 EXECUTORS 1 CORES Jobs 2 COMPLETED

```
[Stage 18:>                                                         (0 + 1) /
+----------+-------+
| clan_type|  count|
+----------+-------+
|      open|2569771|
|inviteOnly| 820839|
|    closed| 169133|
+----------+-------+


Time taken for clan_type aggregation: 17.780 seconds
```

3) Multi-node Clan League Aggregation



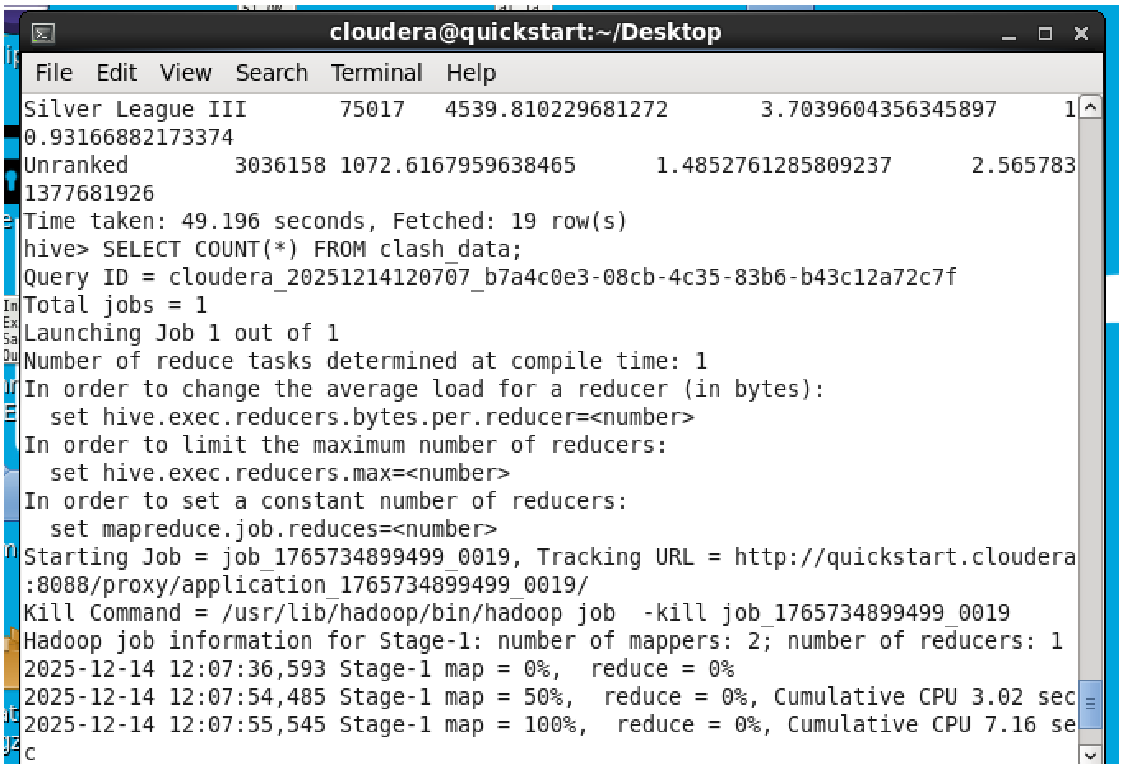
•[6]:

```
from pyspark.sql import functions as F
import time
start = time.time()

league_stats = (
    df.groupBy("clan_war_league")
      .agg(
          F.count("*").alias("num_clans"),
          F.avg("clan_points").alias("avg_clan_points"),
          F.avg("clan_level").alias("avg_clan_level"),
          F.avg("num_members").alias("avg_num_members")
      )
      .orderBy("clan_war_league")
)
league_stats.show()
end = time.time()
print(f"\nTime taken for war league aggregation: {end - start:.2f} seconds")
```

▶ Apache Spark 1 EXECUTORS 1 CORES Jobs 2 COMPLETED

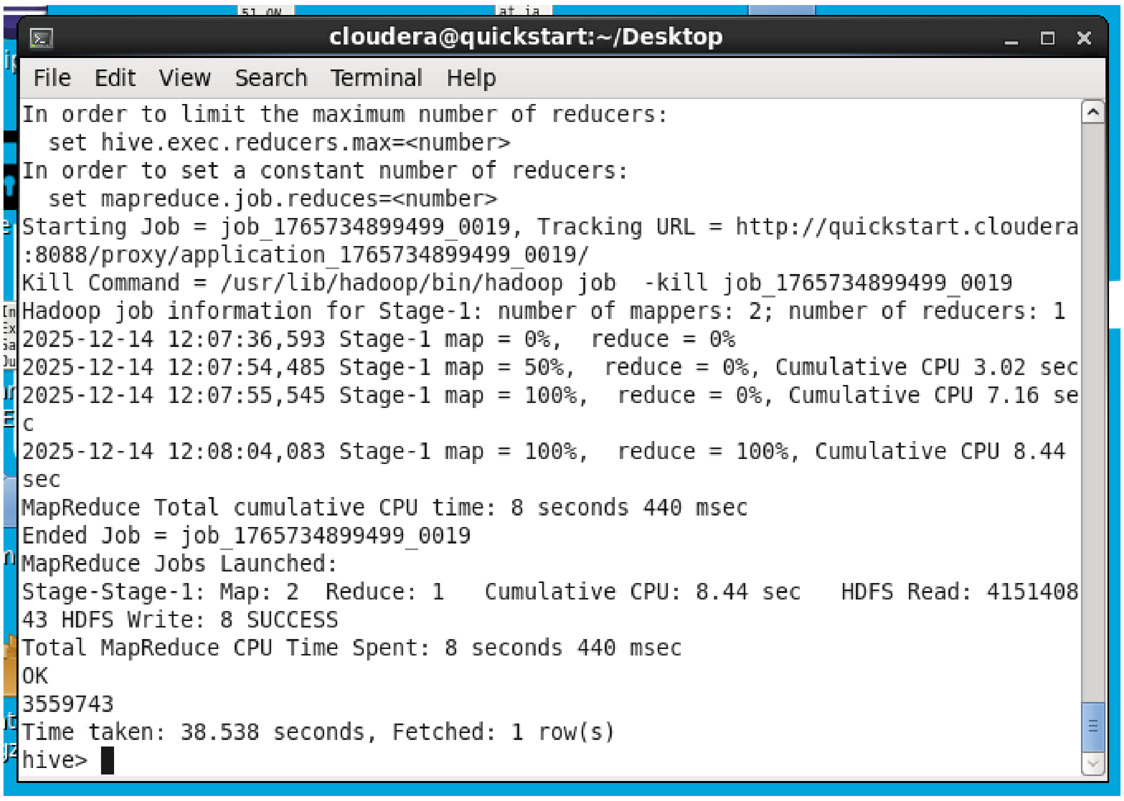
```
[Stage 21:>                                                         (0 + 1) / 1]
+-------------------+---------+------------------+------------------+------------------+
|    clan_war_league|num_clans|   avg_clan_points|    avg_clan_level|   avg_num_members|
+-------------------+---------+------------------+------------------+------------------+
|    Bronze League I|    29300| 4124.027576791809|3.3819453924914678|11.501160409556315|
|   Bronze League II|     9965| 3912.768188660311|3.1250376317109883|12.515203211239339|
|  Bronze League III|     3998|3749.6193096548272|  2.94072036018009|13.536268134067033|
|  Champion League I|      556|22192.793165467625|20.408273381294965|20.190647482014388|
| Champion League II|      676| 27267.76923076923|20.014792899408285| 24.20414201183432|
|Champion League III|     1480| 30826.16689189189| 20.49256756756757|27.681756756756755|
|   Crystal League I|    21187| 31010.17397460707|17.556426110350685|35.706376551659034|
|  Crystal League II|    22230| 25932.39851551957| 15.09676113360324|32.278542510121454|
| Crystal League III|    26156| 21514.61370240098|12.811209665086405| 29.08338430952745|
|      Gold League I|    34396|16801.075735550647|10.599197581114083|24.333934178392838|
|     Gold League II|    49293|12626.054105045341| 8.245044935386364|19.622400746556306|
|    Gold League III|    70027| 9165.393548202836| 6.166764248075742|15.286046810515943|
|    Master League I|     2963| 35091.31488356396|21.337833277084037| 33.03543705703679|
|   Master League II|     5863|34971.713627835576|  20.5973051338905| 34.95480129626471|
|  Master League III|    11288| 33625.49317859674| 19.39466690290574|35.837615166548545|
|    Silver League I|    81990| 6925.987681424564| 4.928113184534699|12.718063178436395|
|   Silver League II|    77200| 5530.494352331606| 4.247810880829015|  11.4464896373057|
|  Silver League III|    75017| 4539.810229681272|3.7039604356345897| 10.93166882173374|
|           Unranked|  3036158|1072.6167959638465|1.4852761285809237|2.5657831377681926|
+-------------------+---------+------------------+------------------+------------------+


Time taken for war league aggregation: 18.17 seconds
```

4)Hive Simple row Count



cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help

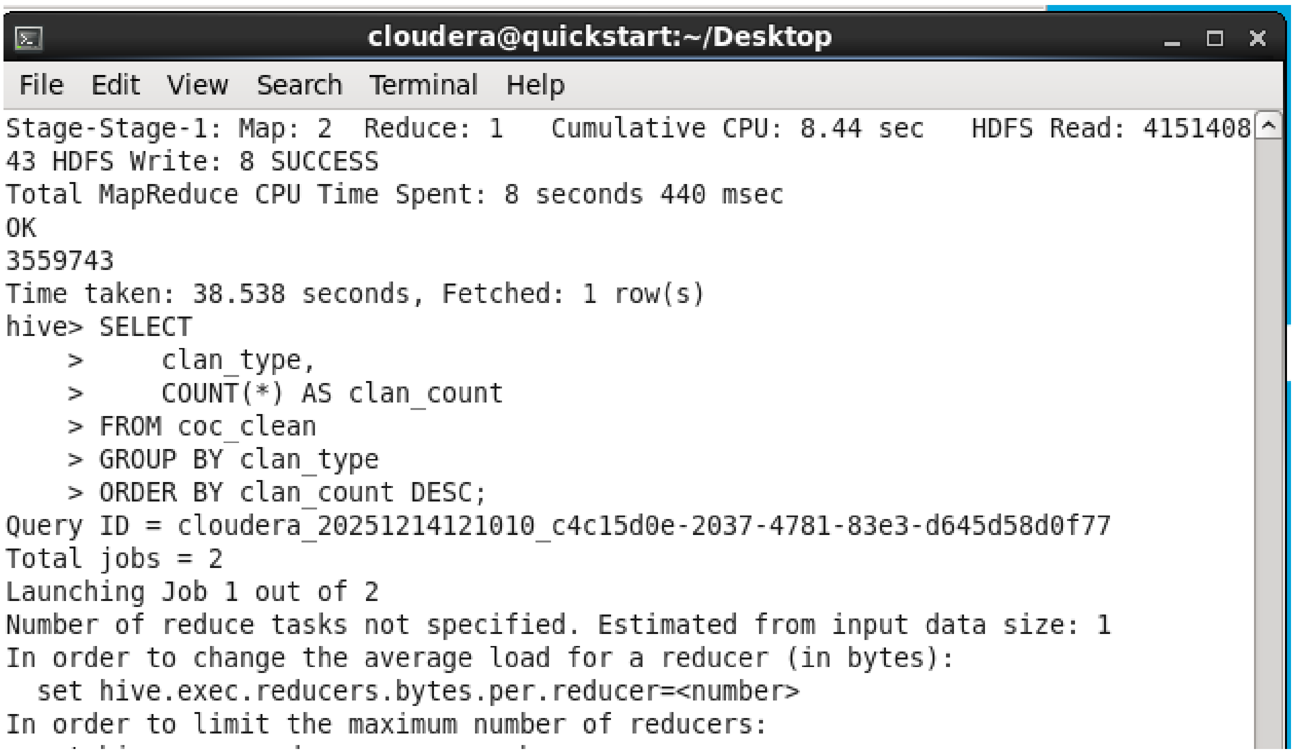
```
Silver League III       75017   4539.810229681272       3.7039604356345897      1
0.93166882173374
Unranked        3036158 1072.6167959638465      1.4852761285809237      2.565783
1377681926
Time taken: 49.196 seconds, Fetched: 19 row(s)
hive> SELECT COUNT(*) FROM clash_data;
Query ID = cloudera_20251214120707_b7a4c0e3-08cb-4c35-83b6-b43c12a72c7f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1765734899499_0019, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1765734899499_0019/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1765734899499_0019
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2025-12-14 12:07:36,593 Stage-1 map = 0%,  reduce = 0%
2025-12-14 12:07:54,485 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 3.02 sec
2025-12-14 12:07:55,545 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 7.16 se
c
```



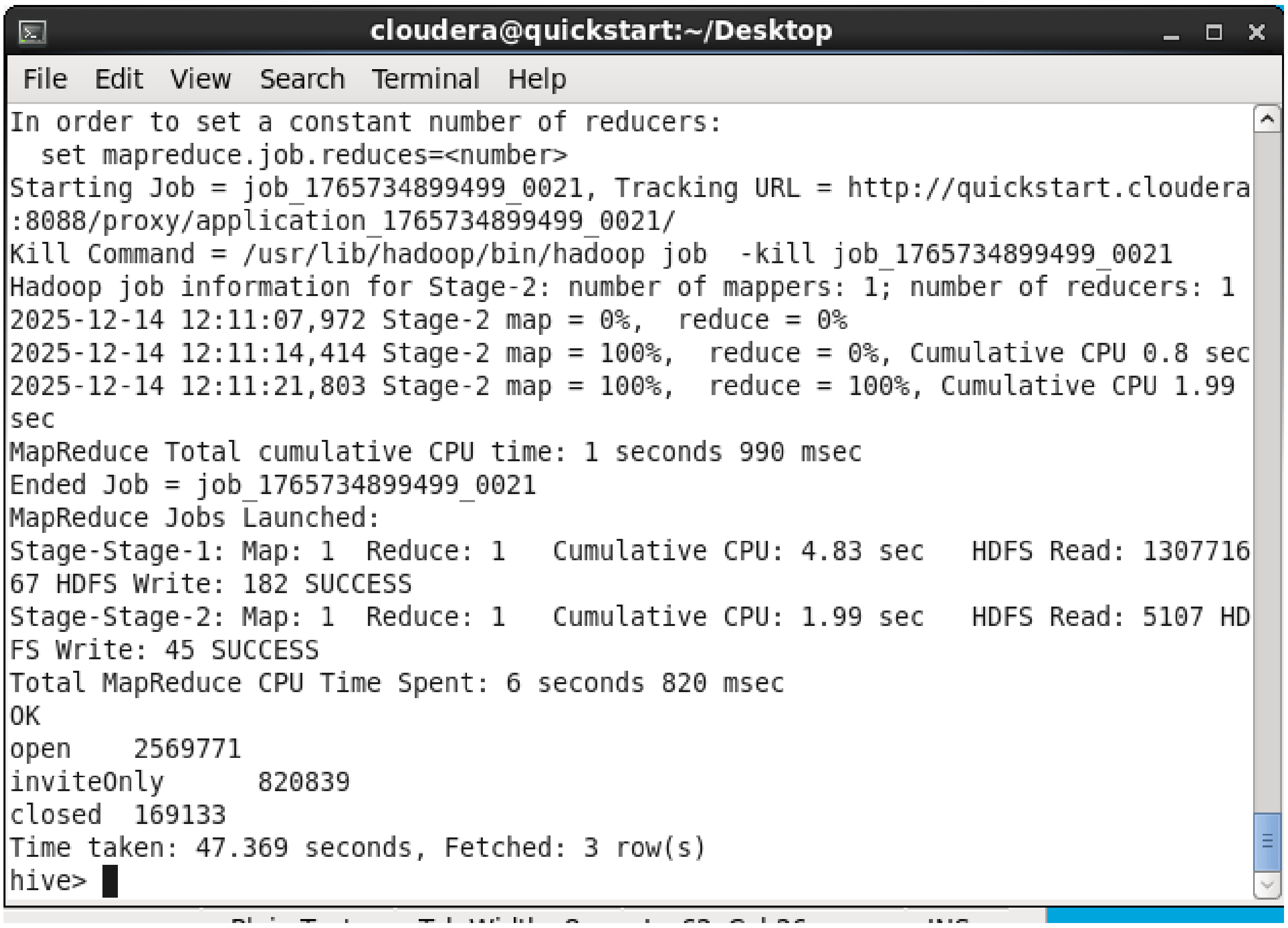
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help

```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1765734899499_0019, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1765734899499_0019/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1765734899499_0019
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2025-12-14 12:07:36,593 Stage-1 map = 0%,  reduce = 0%
2025-12-14 12:07:54,485 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 3.02 sec
2025-12-14 12:07:55,545 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 7.16 se
c
2025-12-14 12:08:04,083 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.44
sec
MapReduce Total cumulative CPU time: 8 seconds 440 msec
Ended Job = job_1765734899499_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 8.44 sec   HDFS Read: 4151408
43 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 440 msec
OK
3559743
Time taken: 38.538 seconds, Fetched: 1 row(s)
hive> 
```

5)Hive clantype aggregation

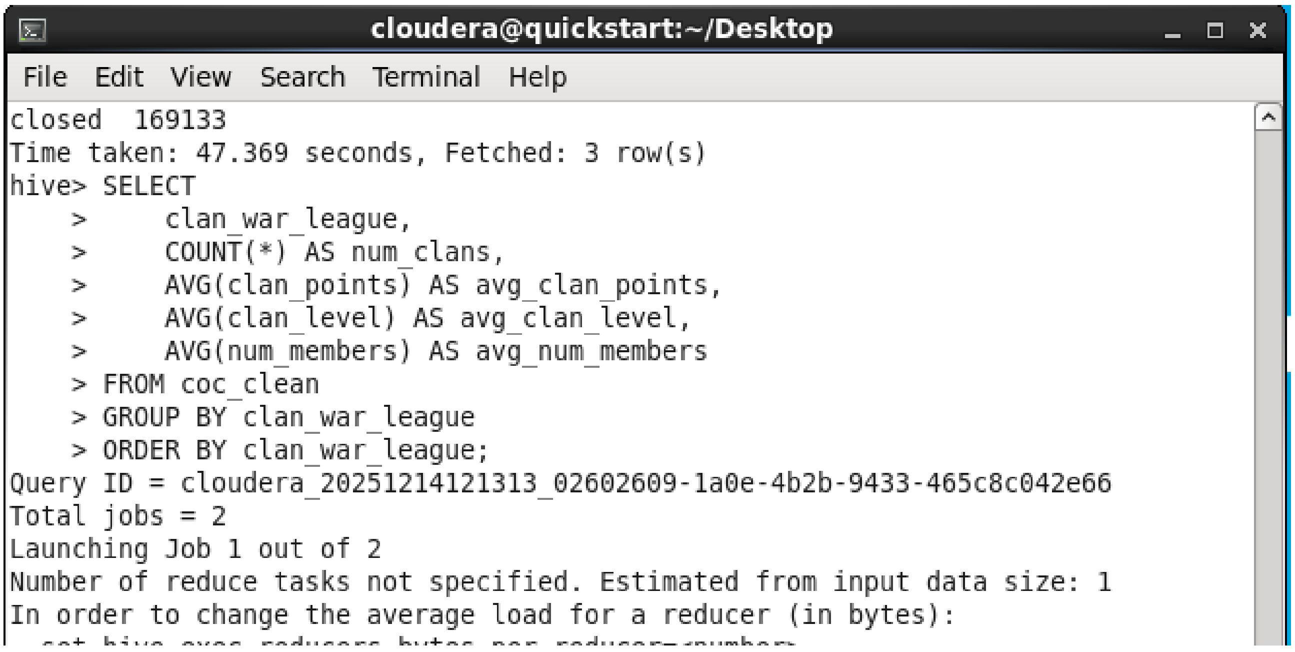


```
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 8.44 sec   HDFS Read: 4151408
43 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 440 msec
OK
3559743
Time taken: 38.538 seconds, Fetched: 1 row(s)
hive> SELECT
    >     clan_type,
    >     COUNT(*) AS clan_count
    > FROM coc_clean
    > GROUP BY clan_type
    > ORDER BY clan_count DESC;
Query ID = cloudera_20251214121010_c4c15d0e-2037-4781-83e3-d645d58d0f77
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
```

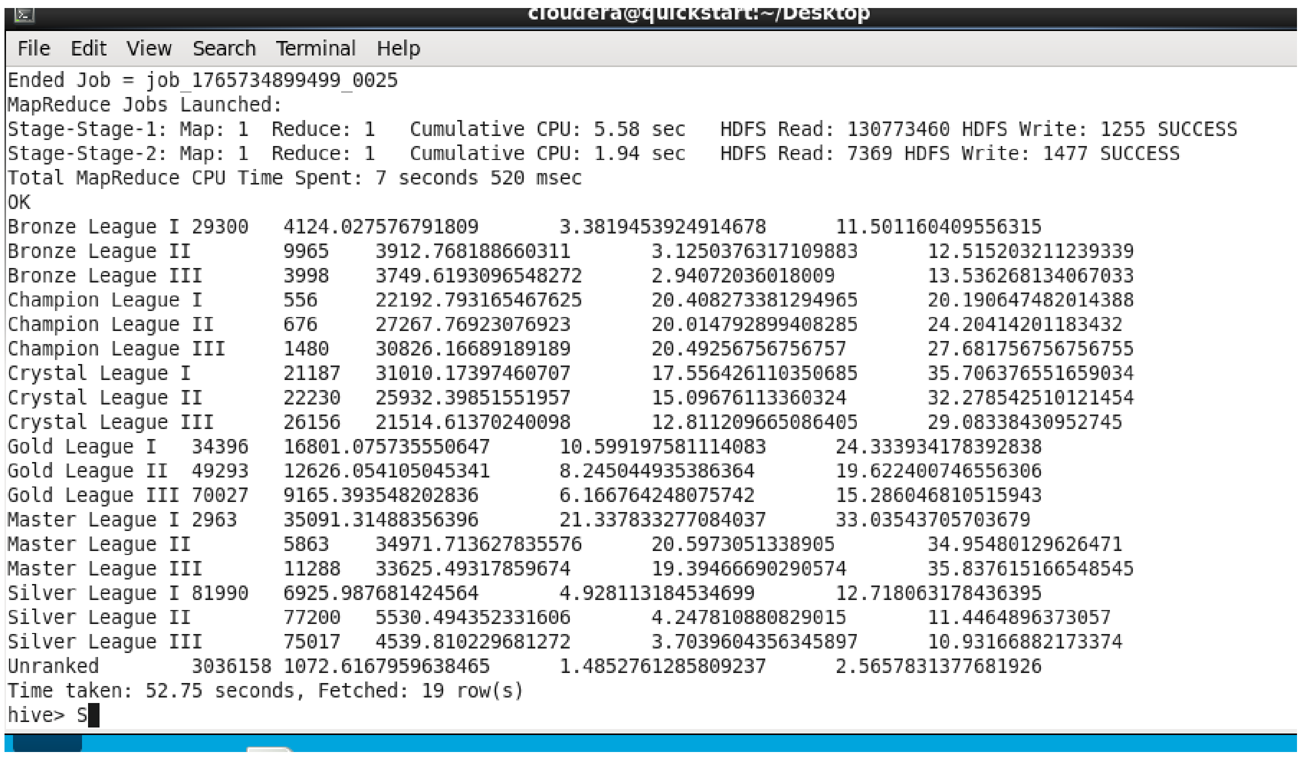


```
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1765734899499_0021, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1765734899499_0021/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1765734899499_0021
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-12-14 12:11:07,972 Stage-2 map = 0%,  reduce = 0%
2025-12-14 12:11:14,414 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.8 sec
2025-12-14 12:11:21,803 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.99
sec
MapReduce Total cumulative CPU time: 1 seconds 990 msec
Ended Job = job_1765734899499_0021
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.83 sec   HDFS Read: 1307716
67 HDFS Write: 182 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.99 sec   HDFS Read: 5107 HD
FS Write: 45 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 820 msec
OK
open    2569771
inviteOnly      820839
closed  169133
Time taken: 47.369 seconds, Fetched: 3 row(s)
hive> 
```

6) Hive Clan League aggregate



```
closed  169133
Time taken: 47.369 seconds, Fetched: 3 row(s)
hive> SELECT
    >     clan_war_league,
    >     COUNT(*) AS num_clans,
    >     AVG(clan_points) AS avg_clan_points,
    >     AVG(clan_level) AS avg_clan_level,
    >     AVG(num_members) AS avg_num_members
    > FROM coc_clean
    > GROUP BY clan_war_league
    > ORDER BY clan_war_league;
Query ID = cloudera_20251214121313_02602609-1a0e-4b2b-9433-465c8c042e66
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
```



```
Ended Job = job_1765734899499_0025
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.58 sec   HDFS Read: 130773460 HDFS Write: 1255 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 1.94 sec   HDFS Read: 7369 HDFS Write: 1477 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 520 msec
OK
Bronze League I 29300   4124.027576791809       3.3819453924914678      11.501160409556315
Bronze League II        9965    3912.768188660311       3.1250376317109883      12.515203211239339
Bronze League III       3998    3749.6193096548272      2.94072036018009        13.536268134067033
Champion League I       556     22192.793165467625      20.408273381294965      20.190647482014388
Champion League II      676     27267.76923076923       20.014792899408285      24.20414201183432
Champion League III     1480    30826.16689189189       20.49256756756757       27.681756756756755
Crystal League I        21187   31010.17397460707       17.556426110350685      35.706376551659034
Crystal League II       22230   25932.39851551957       15.09676113360324       32.278542510121454
Crystal League III      26156   21514.61370240098       12.811209665086405      29.08338430952745
Gold League I   34396   16801.075735550647      10.599197581114083      24.333934178392838
Gold League II  49293   12626.054105045341      8.245044935386364       19.622400746556306
Gold League III 70027   9165.393548202836       6.166764248075742       15.286046810515943
Master League I 2963    35091.31488356396       21.337833277084037      33.03543705703679
Master League II        5863    34971.713627835576      20.5973051338905        34.95480129626471
Master League III       11288   33625.49317859674       19.39466690290574       35.837615166548545
Silver League I 81990   6925.987681424564       4.928113184534699       12.718063178436395
Silver League II        77200   5530.494352331606       4.247810880829015       11.4464896373057
Silver League III       75017   4539.810229681272       3.7039604356345897      10.93166882173374
Unranked        3036158 1072.6167959638465      1.4852761285809237      2.5657831377681926
Time taken: 52.75 seconds, Fetched: 19 row(s)
hive> S
```

Appendix 2:
GCP Source Code:
#Count rows
```
import time
# Path to your file in the bucket
path = "gs://bigdata-bucket4/coc_clans_dataset.csv"
# start = time.time() This is for data ingestion measurement not execution time
# Read the CSV
df = spark.read.csv(
    path,
    header=True,
    inferSchema=True,
    multiLine=True,
    escape="\"",
    quote="\""
)
start = time.time()
df.count()

end = time.time()
print(f"Execution time only: {end - start:.2f} seconds")
```

#Clan type aggregation
```
from pyspark.sql import functions as F

import time
start = time.time()
clan_type_dist = (
    df.groupBy("clan_type")
      .count()
      .orderBy("count", ascending=False)
)
clan_type_dist.show()
end = time.time()
print(f"\nTime taken for clan_type aggregation: {end - start:.3f} seconds")
```

#Clan League aggregation
```
from pyspark.sql import functions as F
import time
start = time.time()

league_stats = (
    df.groupBy("clan_war_league")
      .agg(
          F.count("*").alias("num_clans"),
          F.avg("clan_points").alias("avg_clan_points"),
          F.avg("clan_level").alias("avg_clan_level"),
          F.avg("num_members").alias("avg_num_members")
      )
      .orderBy("clan_war_league")
)
league_stats.show()
```

```
end = time.time()
print(f"\nTime  taken for war league aggregation: {end - start:.2f} seconds")
```

Hive Source Code:
Steps for Hive analysis:
Hadoop Filesystem: hadoop fs -put 'clash_clean.csv'
Initialised hive in terminal
Step-1: Creating table and inserting data:

```
CREATE TABLE clash_data ( clan_tag STRING, clan_name STRING, clan_type STRING,
clan_location STRING, isFamilyFriendly STRING, clan_level INT, clan_points INT,
clan_builder_base_points INT, clan_versus_points INT, required_trophies INT, war_frequency
STRING, war_win_streak INT, war_wins INT, war_ties INT, war_losses INT, clan_war_league
STRING, num_members INT, required_builder_base_trophies INT, required_versus_trophies INT,
required_townhall_level INT, clan_capital_hall_level INT, clan_capital_points INT, capital_league
STRING, mean_member_level DOUBLE, mean_member_trophies DOUBLE ) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

STEP-2: (Load csv data from hadoop into the table) :

```
LOAD DATA INPATH 'clash_clean_25.csv' INTO TABLE clash_data;
```

STEP-3: (Verify data insertion)

```
SELECT COUNT(*) FROM clash_data;
```

STEP-4: Clantype aggregation:

```
SELECT clan_type, COUNT(*) AS clan_count FROM coc_clean GROUP BY clan_type ORDER
BY clan_count DESC;
```

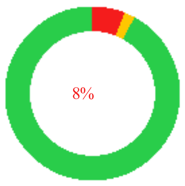
STEP-5: Clan league aggregation:

```
SELECT clan_war_league, COUNT(*) AS num_clans, AVG(clan_points) AS avg_clan_points,
AVG(clan_level) AS avg_clan_level, AVG(num_members) AS avg_num_members FROM
coc_clean GROUP BY clan_war_league ORDER BY clan_war_league;
```

SmallSEOTools

Appendix 3: Plagiarism Details

OLS



● Plagiarism	8%	● Partial Match	2%
● Exact Match	6%	● Unique	92%

Scan details

Total Words	Total Characters	Plagiarized Sentences	Unique Sentences
1025	7023	3.28	37.72 (92%)

Plagiarism Results: (4)

#1 2% Similar https://massedcompute.com/faq-answers/?questio...

* Virtual Machine with sufficient CPU, RAM, and storage.

#2 2% Similar https://massedcompute.com/faq-answers/?questio...

* Virtual Machine with sufficient CPU, RAM, and storage.

#3 2% Similar https://security.cbidigital.com/uploaded-files/e7671...

* Virtual Machine with sufficient CPU, RAM, and storage.

#4 2% Similar https://www.databricks.com/research/apache-spar...

Apache Spark: A Unified Engine for Big Data Processing.