

**LARGE SCALE DATA ANALYSIS USING MAPREDUCE**

**ASSIGNMENT 1**

EC7205: CLOUD COMPUTING

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF RUHUNA

EG/2020/3797 - Aberuwan R. M. M. P.

EG/2020/4190 - Sandamal I.M.K.

EG/2020/4219 - Siriwardhana T.D.R.D.

Submitted Date: 7th June 2025

**CONTENTS**

[1 Introduction 1](#_Toc200217670)

[2 Dataset Description 1](#_Toc200217671)

[3 Problem Definition 1](#_Toc200217672)

[4 Environment Setup 1](#_Toc200217673)

[5 Implementation 8](#_Toc200217674)

[6 Execution & Testing 8](#_Toc200217675)

[7 Results & Interpretation 13](#_Toc200217676)

[8 Conclusion 14](#_Toc200217677)

[9 References 14](#_Toc200217678)

**LIST OF FIGURES**

[Figure 1: Evidence for Java Version 2](#_Toc200217681)

[Figure 2: Evidence for Hadoop Directory Structure 3](#_Toc200217682)

[Figure 3: Evidence for SSH Setup 3](#_Toc200217683)

[Figure 4: Evidence for Environment Variables 4](#_Toc200217684)

[Figure 5: Evidence for Configuration Files 5](#_Toc200217685)

[Figure 6: Evidence for Namenode Formatting 5](#_Toc200217686)

[Figure 7: Evidence for Starting Hadoop Service 6](#_Toc200217687)

[Figure 8: Evidence for Running Processes 6](#_Toc200217688)

[Figure 9: Evidence for NameNode UI: http://localhost:9870 7](#_Toc200217689)

[Figure 10: Evidence for ResourceManager UI: http://localhost:8088 7](#_Toc200217690)

[Figure 11: Evidence for HDFS Commands 8](#_Toc200217691)

[Figure 12: Evidence for Uploaded the Spotify CSV Dataset to HDFS 9](#_Toc200217692)

[Figure 13: Evidence for Made Sure both Scripts were Executable and Had the Correct Shebang. 9](#_Toc200217693)

[Figure 14: Evidence for Ran the Hadoop Streaming Job 10](#_Toc200217694)

[Figure 15: Evidence for Retrieved and Viewed the Output 11](#_Toc200217695)

[Figure 16: Evidence for Input 12](#_Toc200217696)

[Figure 17: Evidence for Output 13](#_Toc200217697)

# Introduction

The aim of this assignment is to implement a custom MapReduce job using Hadoop to process the Spotify dataset and extract meaningful insights related to artist popularity, genre trends, and follower distribution.

# Dataset Description

* Dataset: Spotify Datasets (Kaggle) [1]

<https://www.kaggle.com/datasets/lehaknarnauli/spotify-datasets?select=artists.csv>

* Size: 1104349 rows, 5 columns
* Features Used: id, followers, genres, name, popularity

This dataset contains metadata about Spotify artists, including unique IDs, follower counts, associated genres, artist names, and a popularity score. The dataset is suitable for large-scale distributed analysis due to its size and structure.

# Problem Definition

Analyze the Spotify dataset to answer the following questions using MapReduce:

* What is the average number of followers for artists at each popularity score?
* How does follower count correlate with popularity scores?

Chosen MapReduce Task: **Popularity-Follower Aggregation**

* Map: For each artist, emit (popularity\_score, followers)
* Reduce: For each popularity score, compute:
  + Total followers across all artists with that score
  + Average followers per popularity score

Mapper Logic:

* Your mapper.py extracts popularity (as the key) and followers (as the value) from each artist.
* Example output: 0 52.68 (popularity=0, followers=52.68).

Reducer Logic:

* Your reducer.py groups data by popularity and calculates the average followers for each score.
* Example output: 0 52.68 (average followers for popularity=0).

# Environment Setup

Platform:

* Local Linux machine (single-node setup)

Installation Steps Overview:

1. Install Java:  
   sudo apt-get install default-jdk
2. Create Hadoop User:  
   sudo adduser hadoop
3. Set Up SSH:  
   ssh-keygen -t rsa and enable passwordless SSH with cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
4. Download Hadoop:  
   Download and extract Hadoop from [Apache Hadoop](https://hadoop.apache.org/releases.html).
5. Set Environment Variables:  
   Add Hadoop and Java paths to ~/.bashrc.
6. Configure Hadoop:  
   Edit config files in /usr/local/hadoop/etc/hadoop/.
7. Format NameNode:  
   hdfs namenode -format
8. Start Hadoop:  
   start-dfs.sh and start-yarn.sh
9. Verify:  
   Use jps and access web UIs at http://localhost:9870 (NameNode) and http://localhost:8088 (ResourceManager).

Screenshots for Evidence:

JavaVersion:

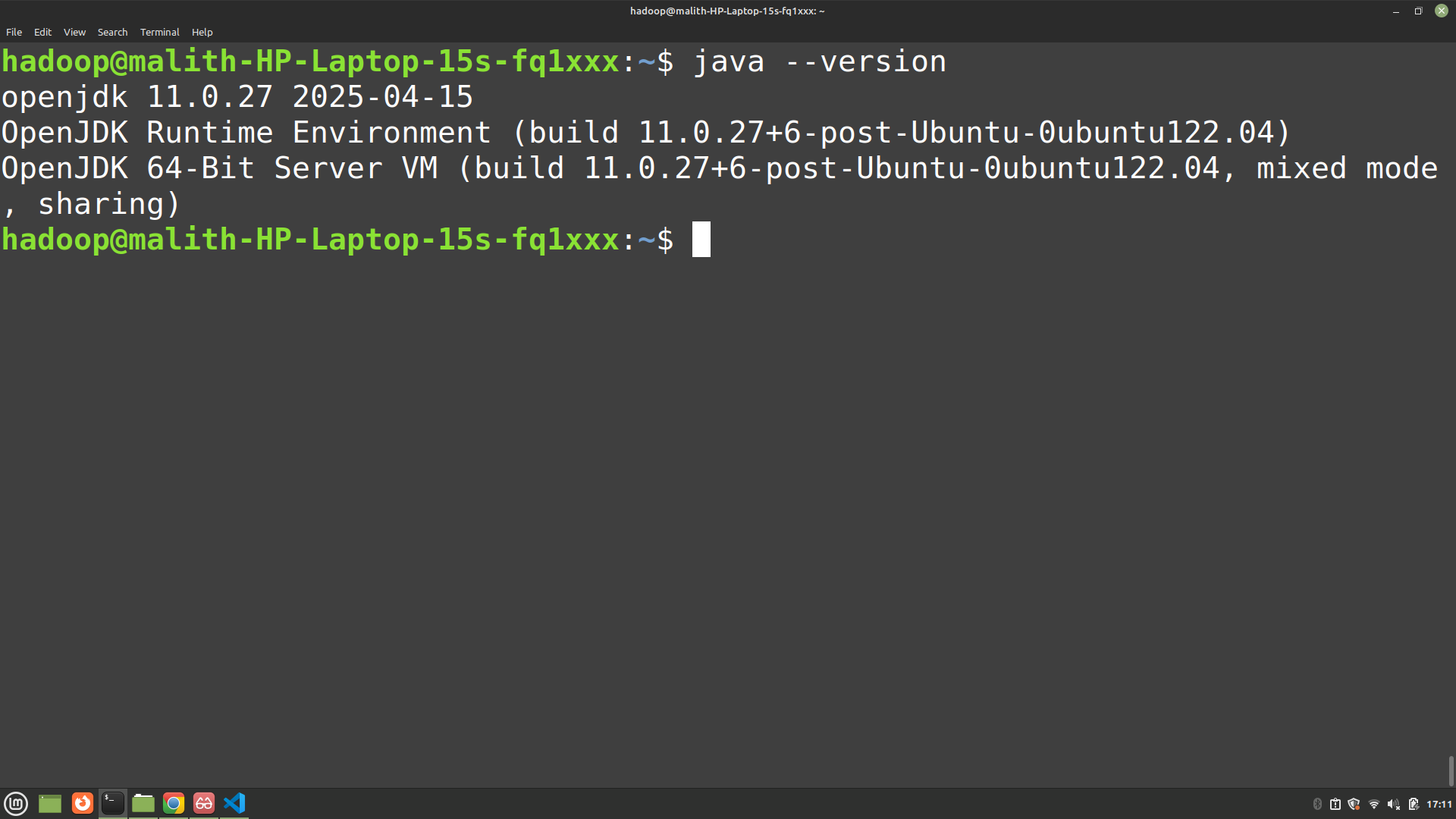


Figure 1: Evidence for Java Version

Hadoop Directory Structure:

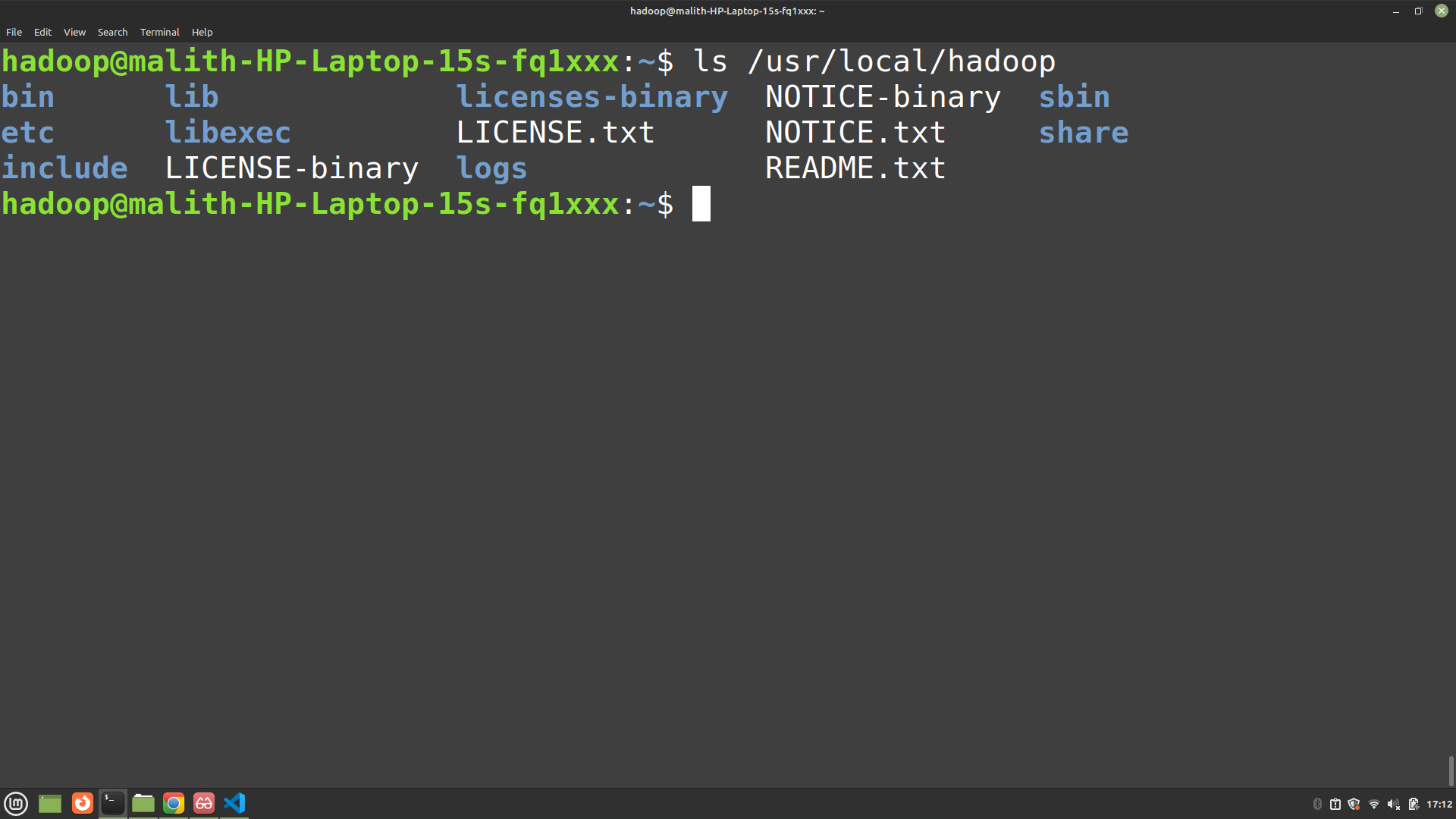


Figure 2: Evidence for Hadoop Directory Structure

SSH Setup:

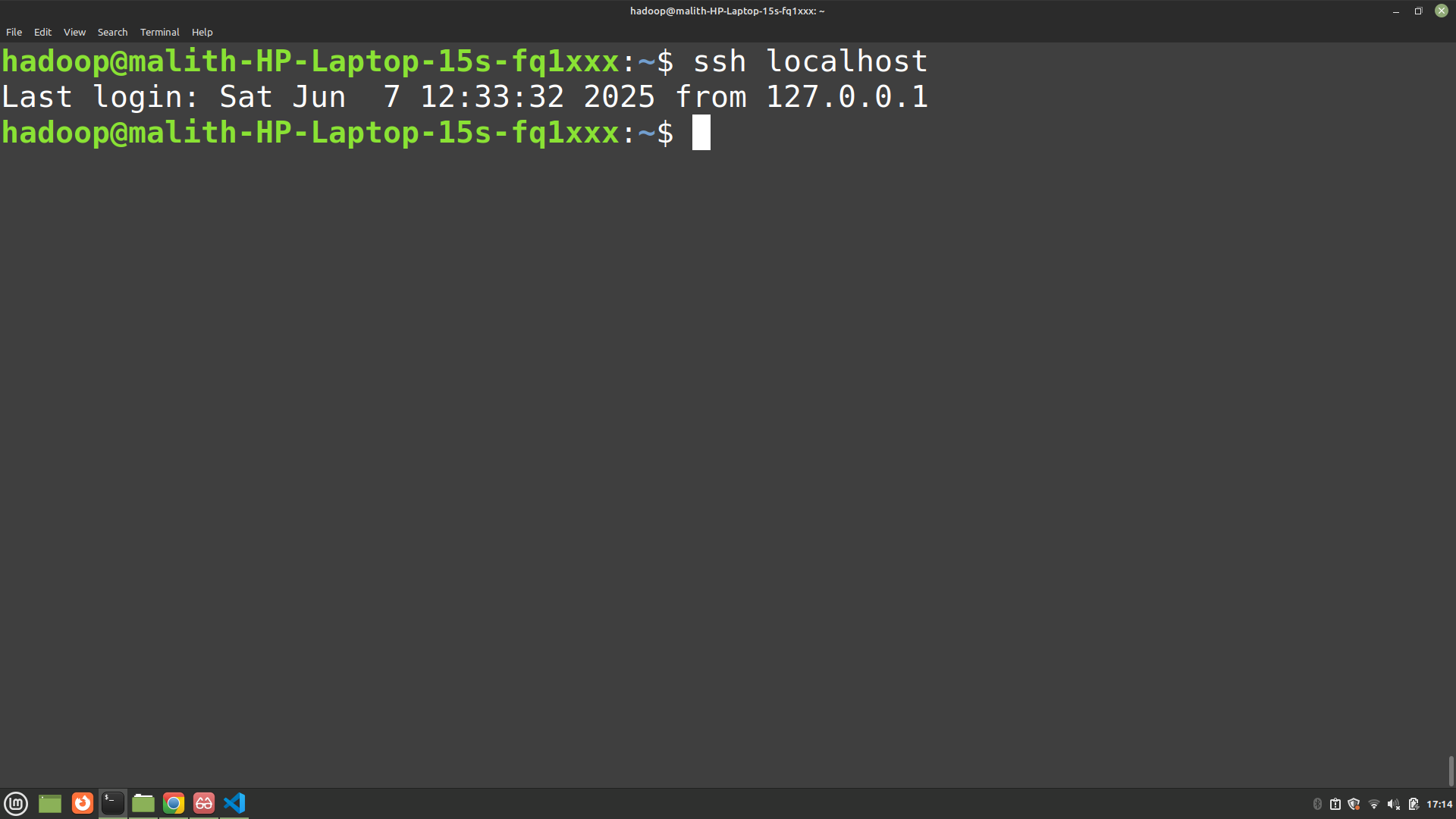


Figure 3: Evidence for SSH Setup

Environment Variables:

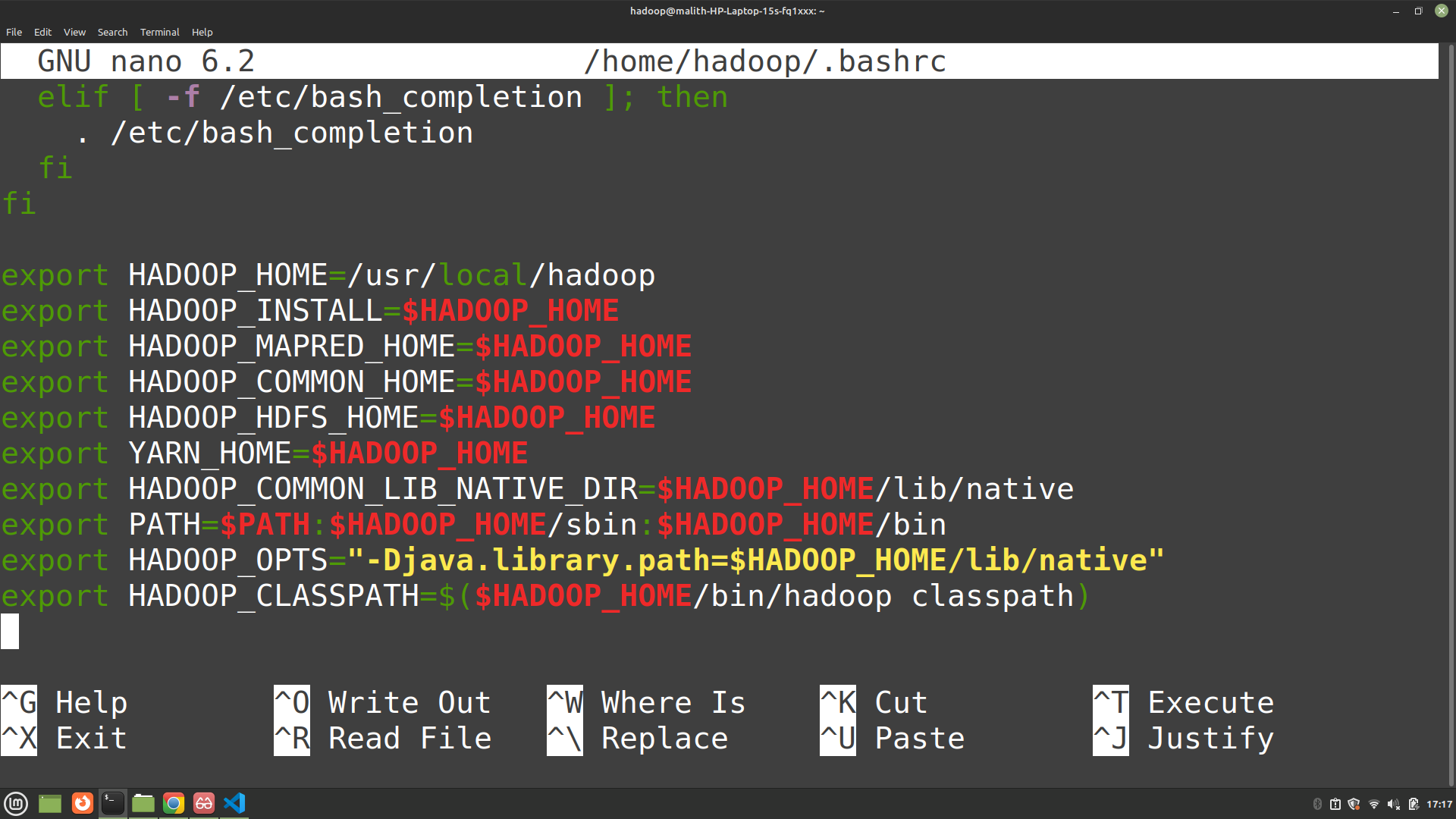
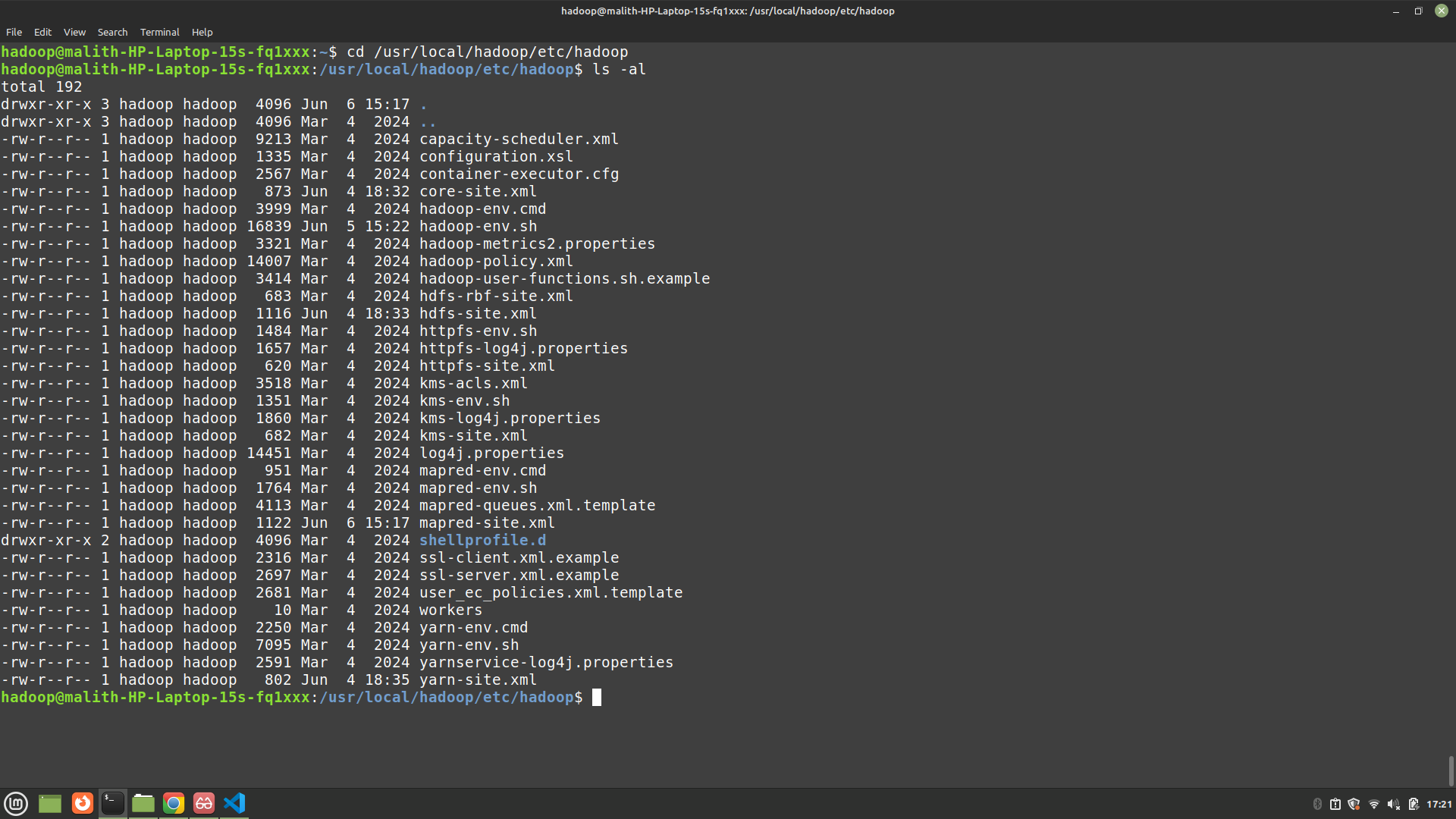


Figure 4: Evidence for Environment Variables

Configuration Files:



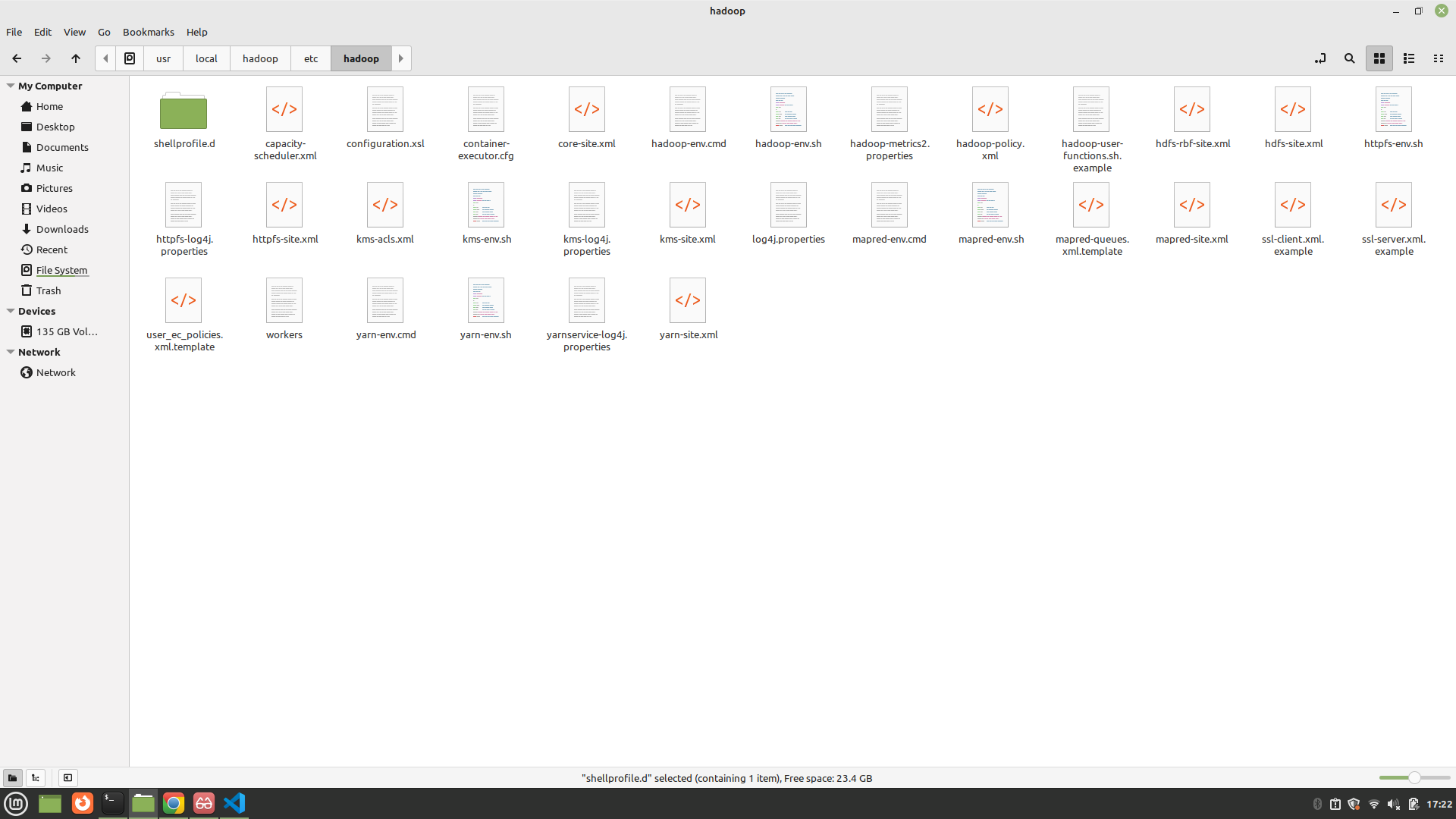


Figure 5: Evidence for Configuration Files

Namenode Formatting:

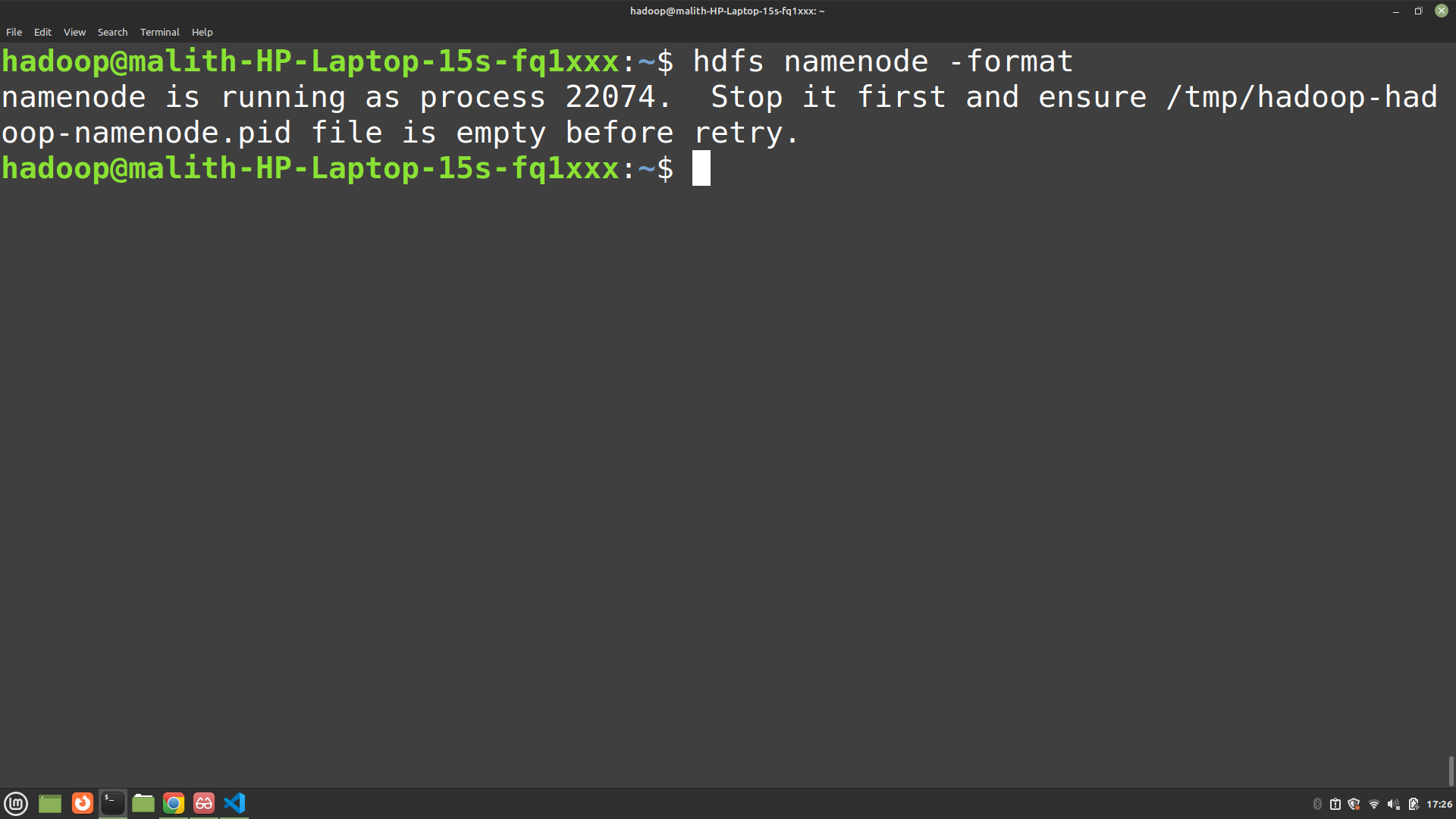


Figure 6: Evidence for Namenode Formatting

Starting Hadoop Service:

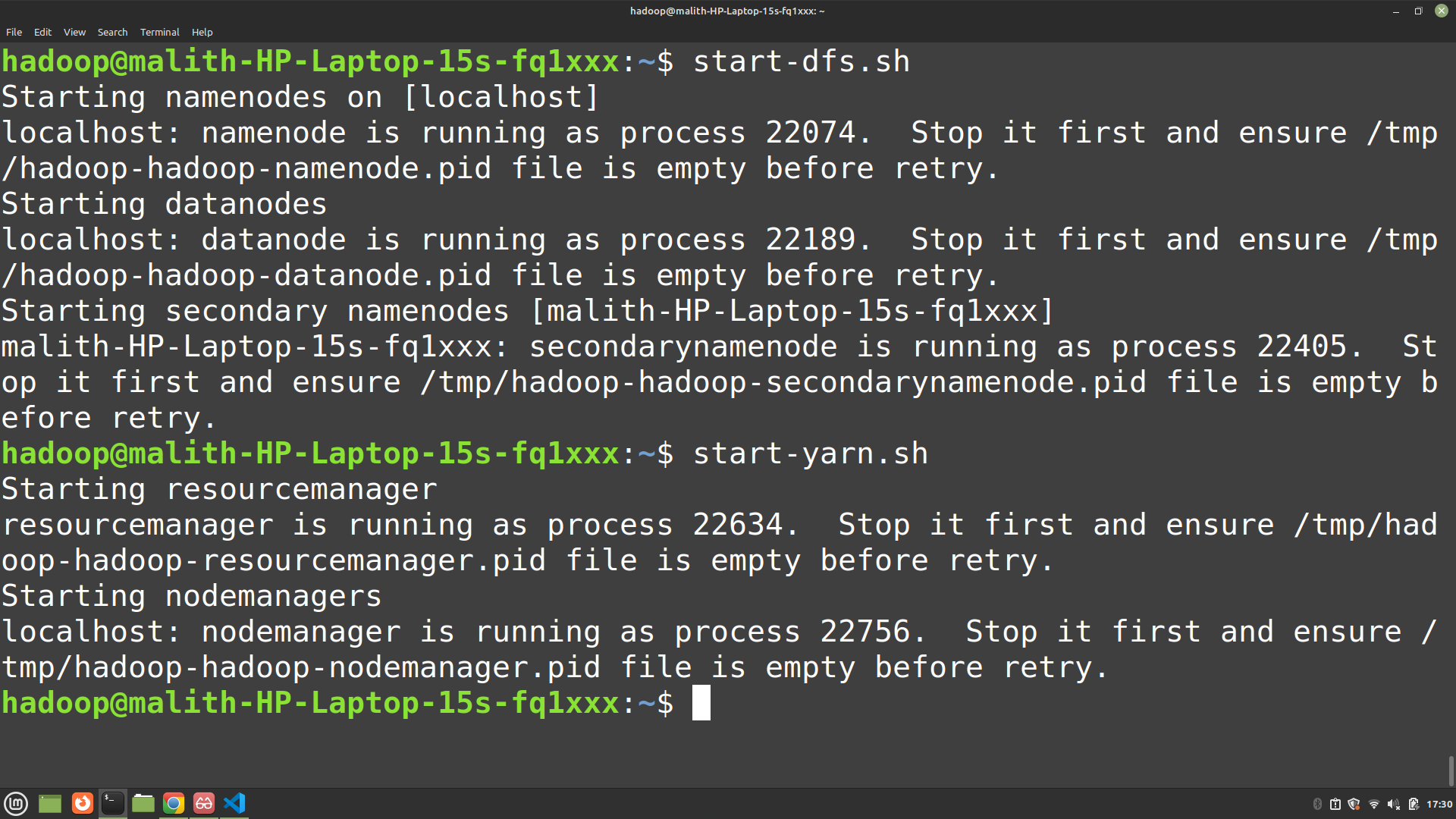


Figure 7: Evidence for Starting Hadoop Service

Running Processes:

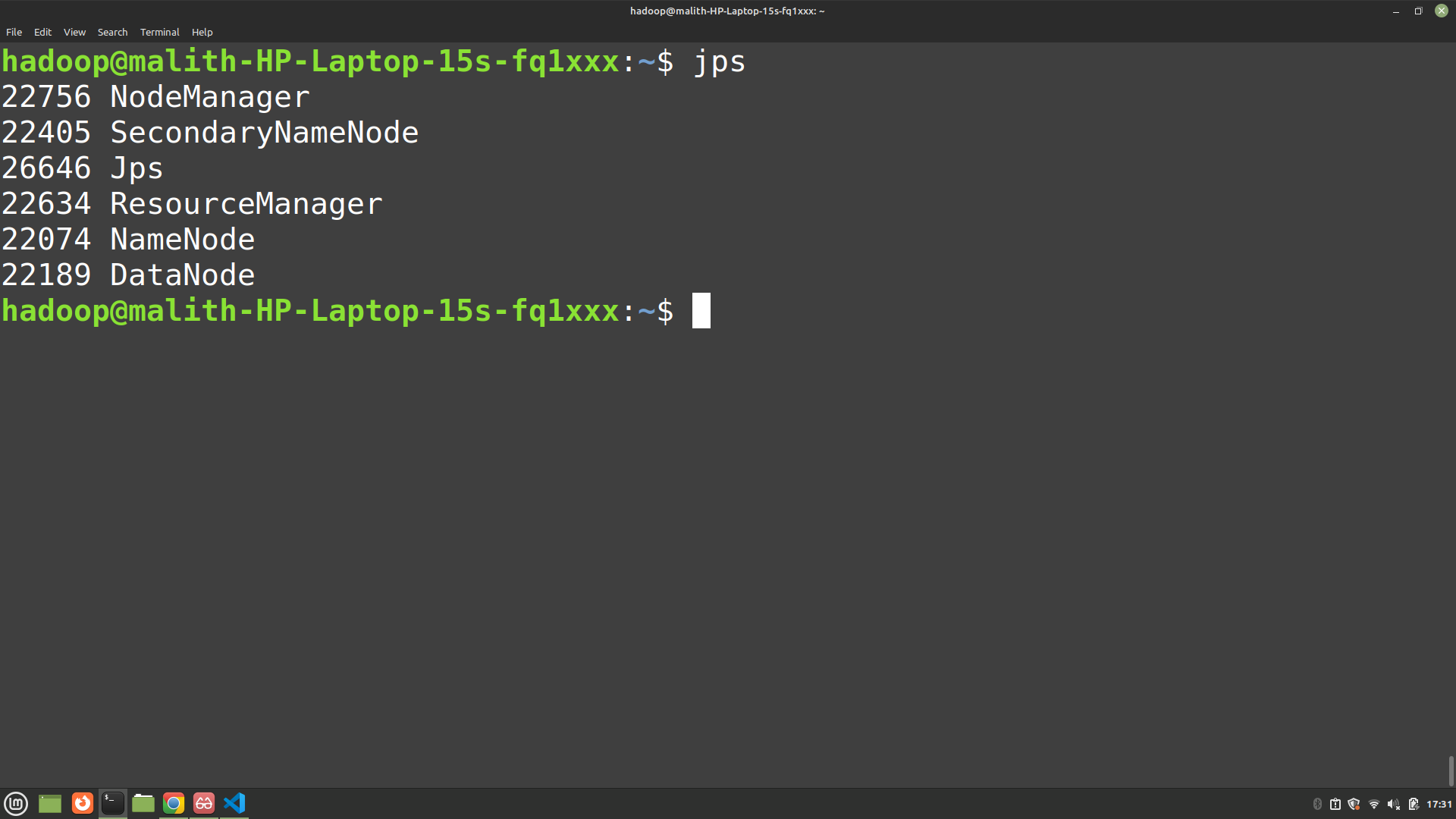


Figure 8: Evidence for Running Processes

Web UI Access:

* NameNode UI: http://localhost:9870

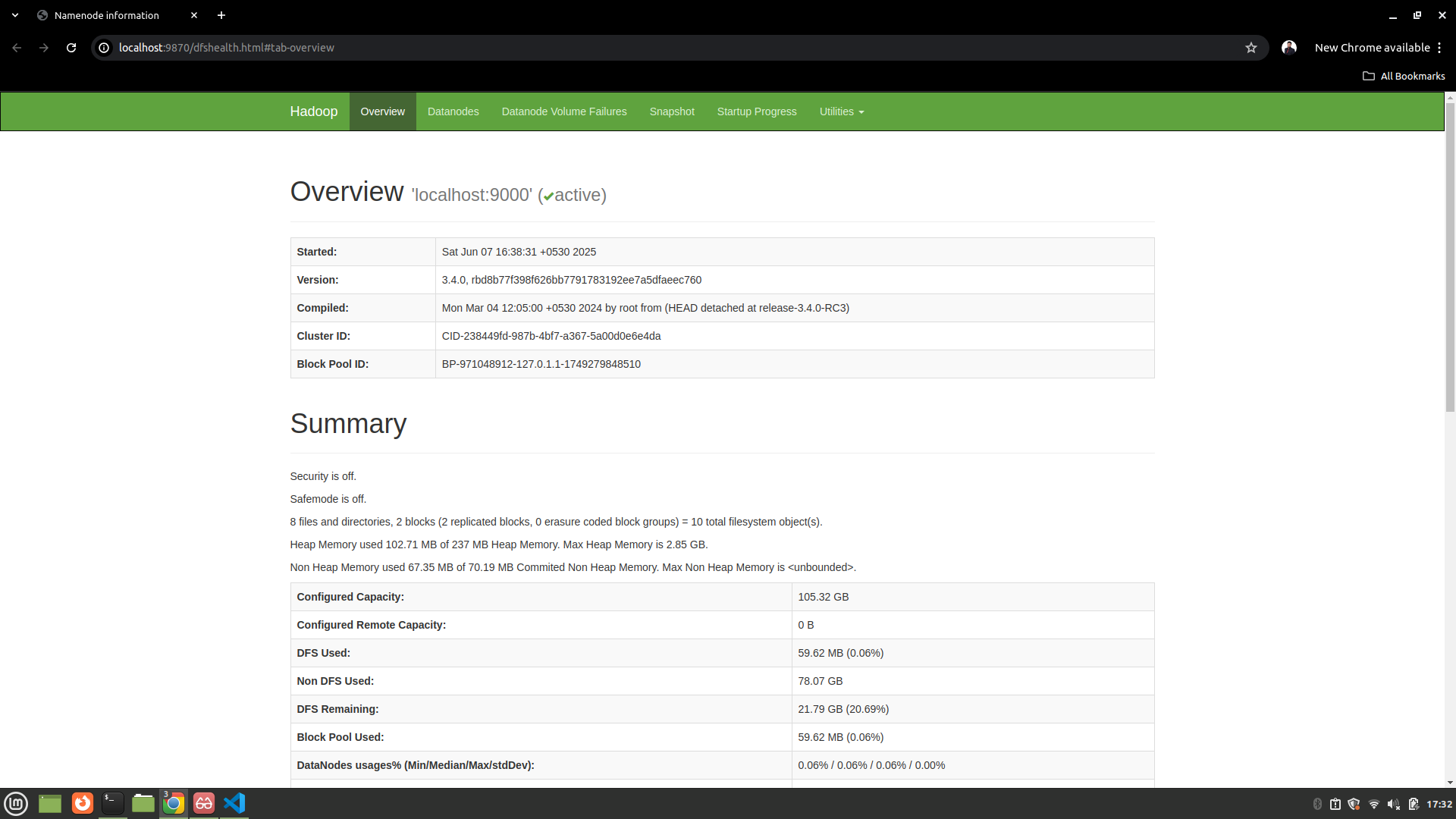


Figure 9: Evidence for NameNode UI: http://localhost:9870

* ResourceManager UI: http://localhost:8088

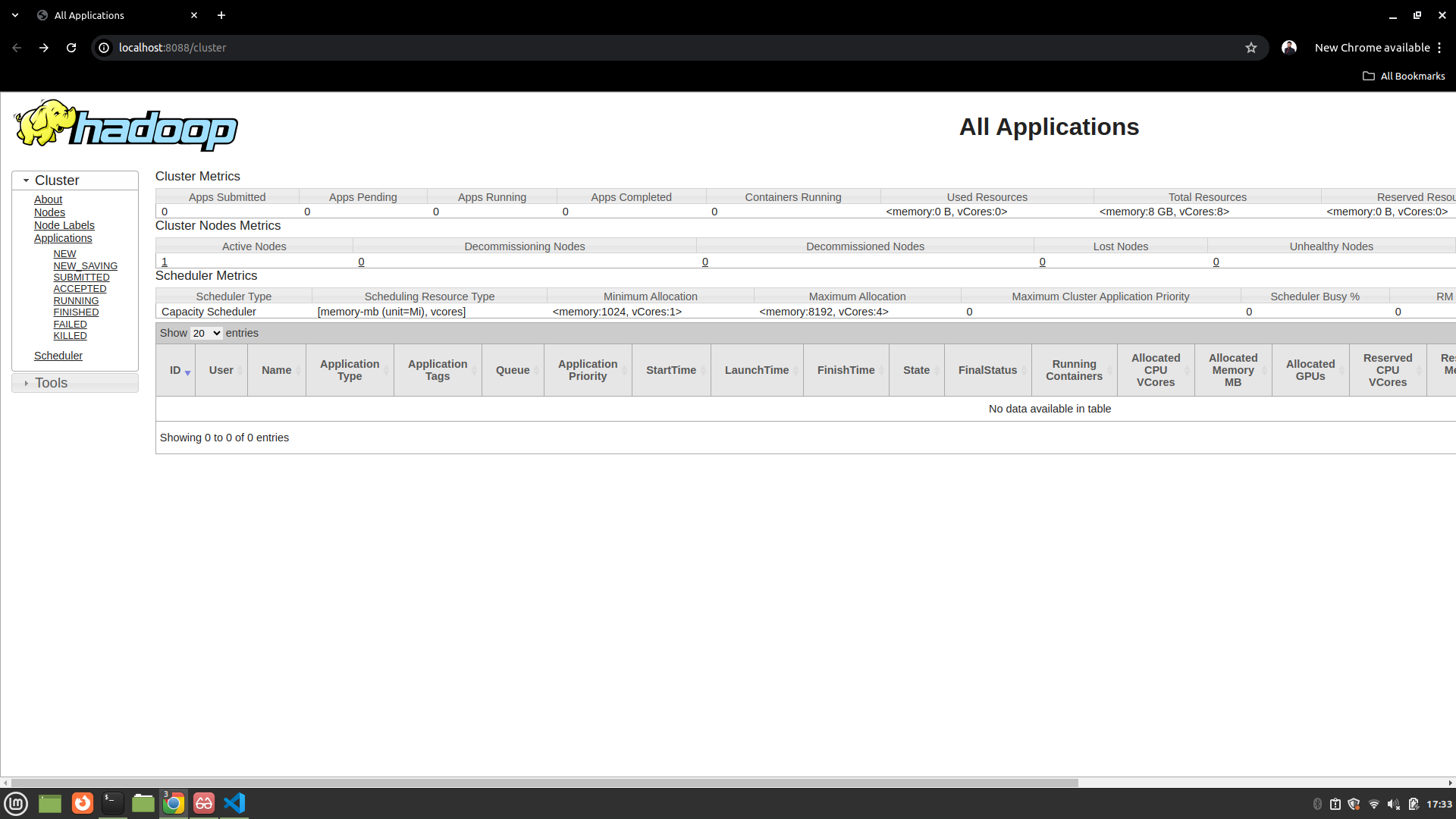


Figure 10: Evidence for ResourceManager UI: http://localhost:8088

HDFS Commands:

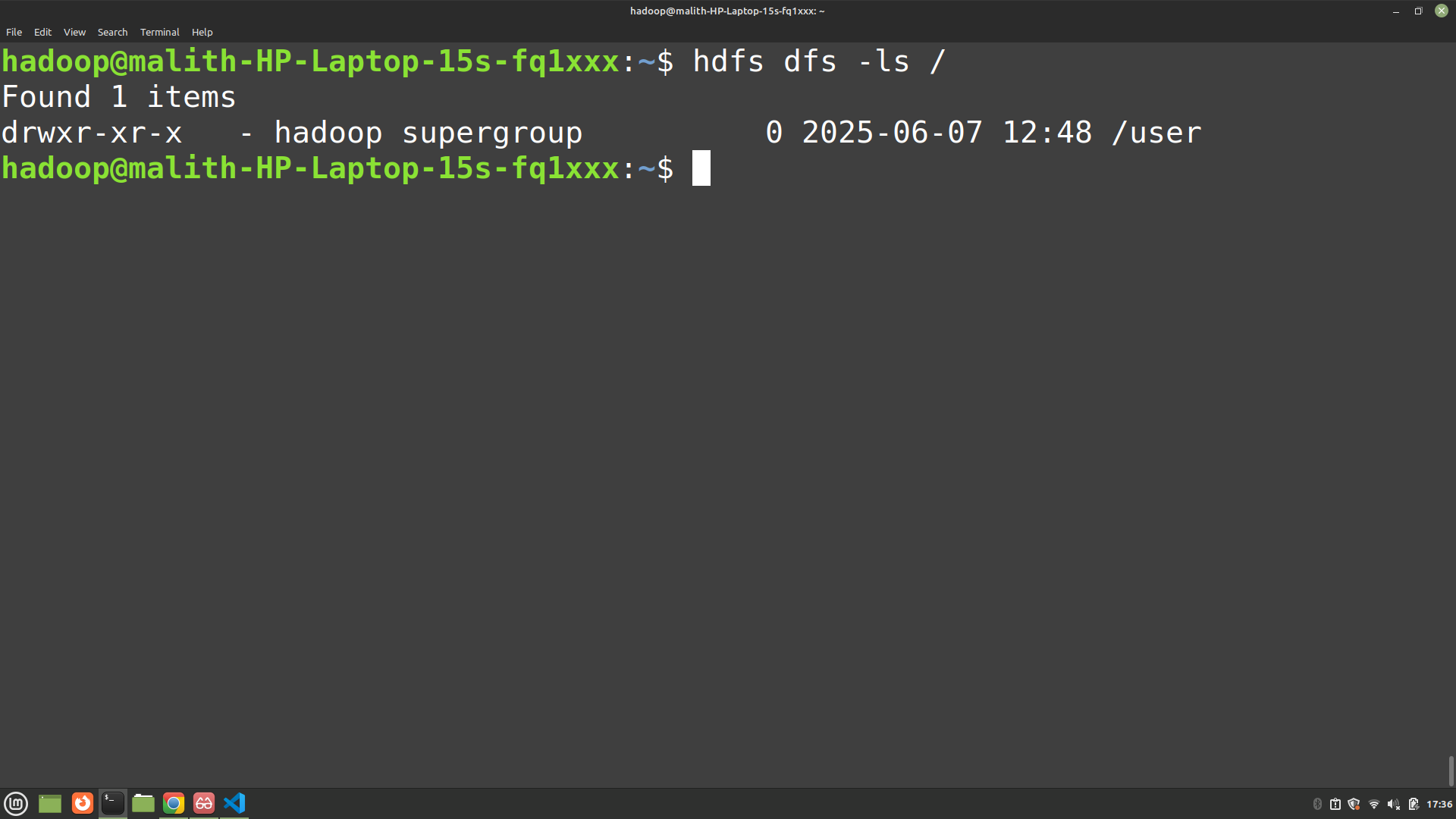


Figure 11: Evidence for HDFS Commands

# Implementation

Map Logic:

* The mapper reads each row of the Spotify dataset, extracts the artist's popularity score and follower count, and emits them as a key-value pair (popularity\_score, followers). It skips rows with missing or invalid values.

Reduce Logic:

* The reducer receives all follower counts grouped by popularity score. For each popularity score, it sums the followers and counts the number of artists, then calculates and emits the average followers for that popularity score.

Source Code:

* Both mapper.py and reducer.py are implemented in Python 3 using the standard csv and sys libraries. [<https://github.com/MalithPramoditha/assignment_1>]

Libraries/Frameworks Used:

* Python 3 standard library (csv, sys)
* Hadoop Streaming for running Python scripts as MapReduce jobs

# Execution & Testing

Steps Taken to Run the Job:

1. Uploaded the Spotify CSV dataset to HDFS:

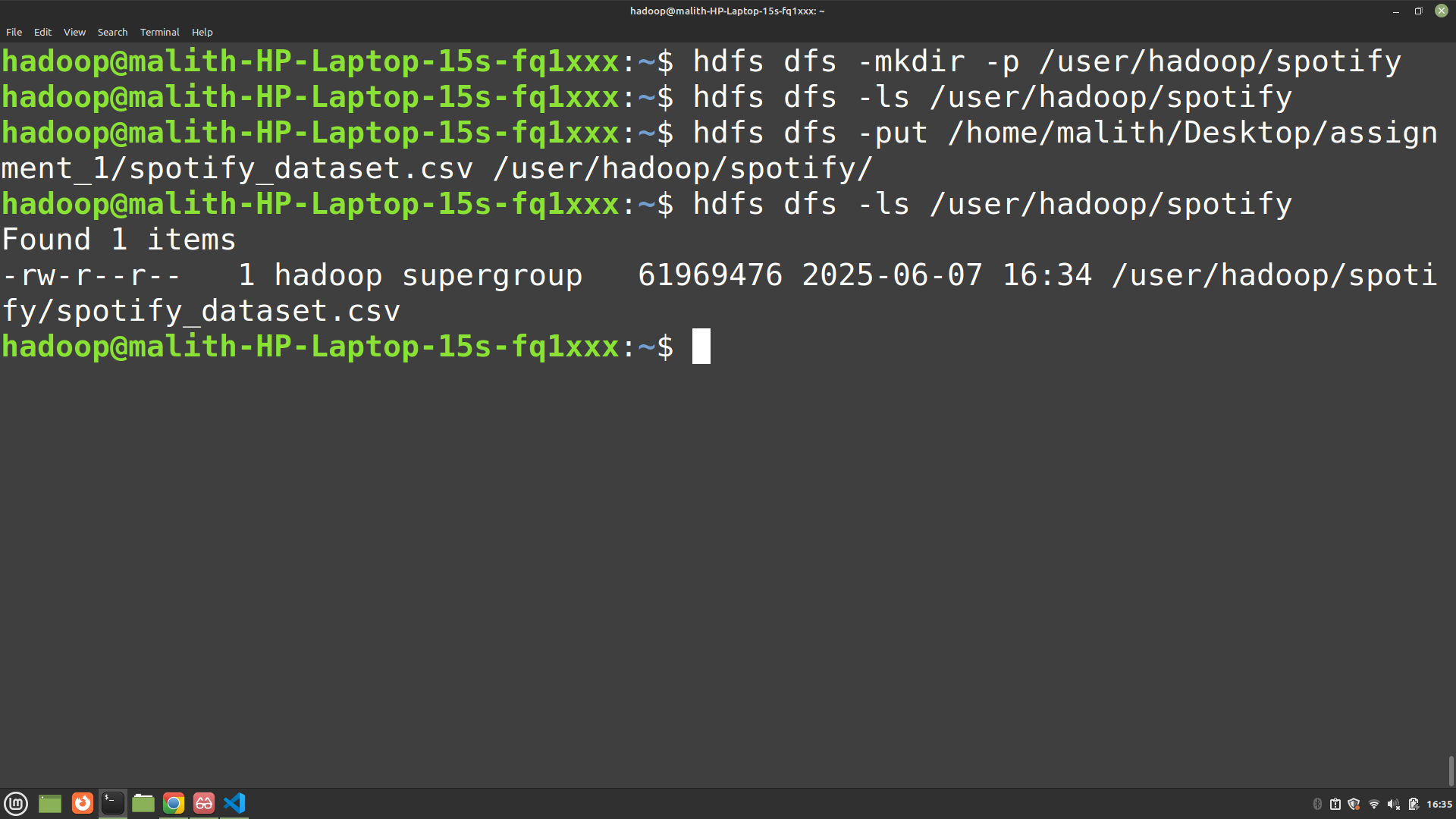


Figure 12: Evidence for Uploaded the Spotify CSV dataset to HDFS

1. Made sure both scripts were executable and had the correct shebang.

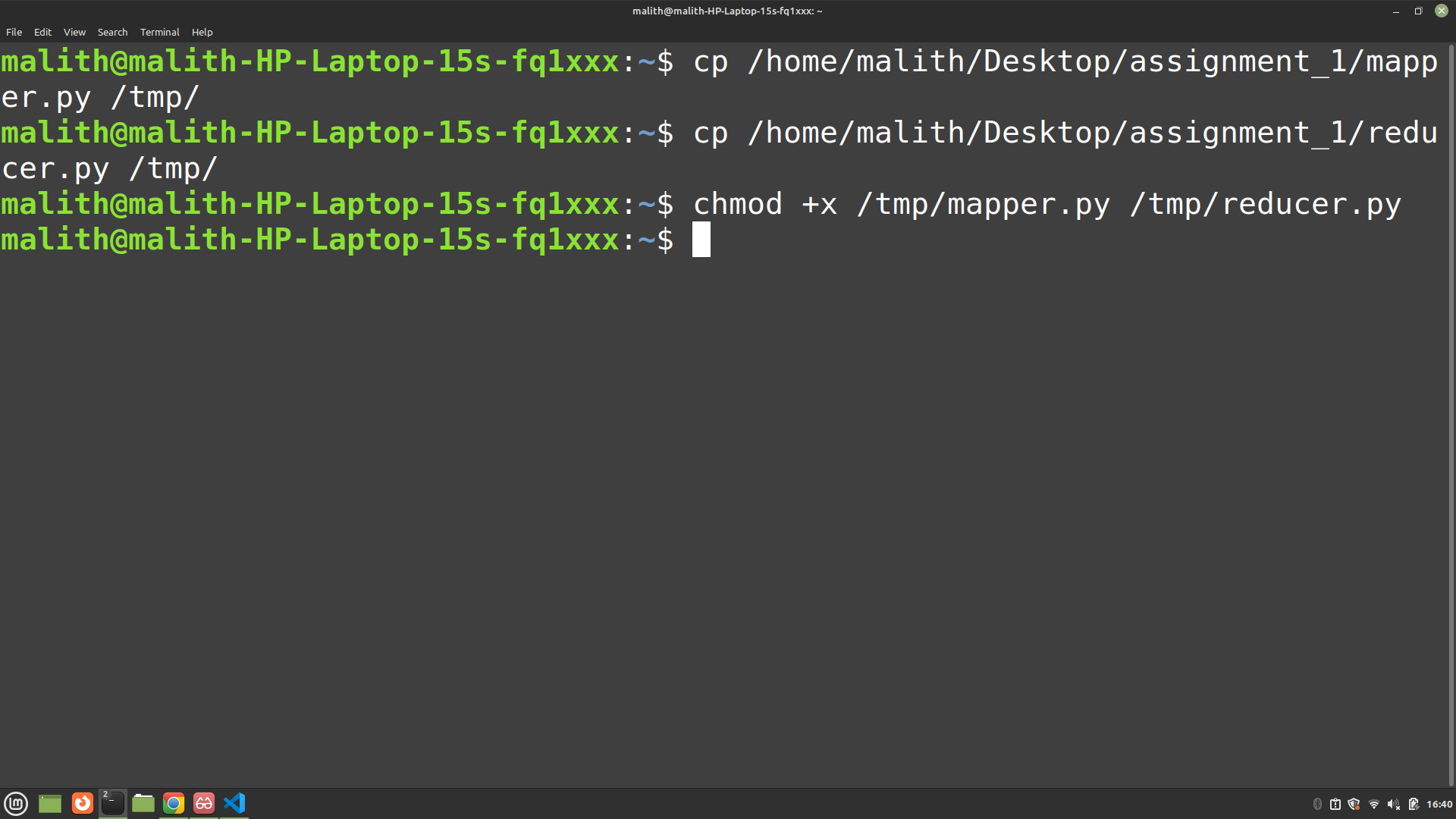
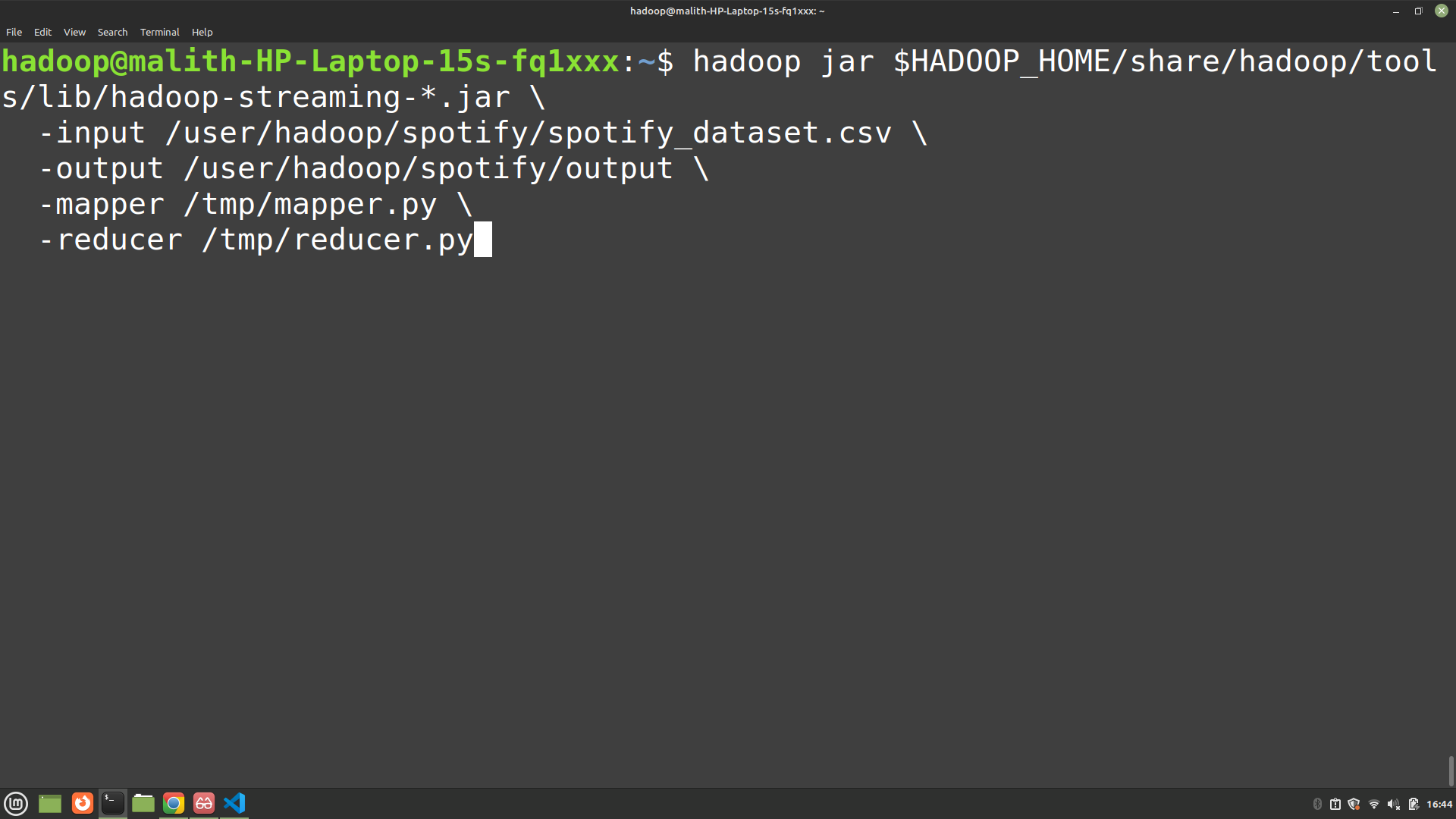


Figure 13: Evidence for Made sure both scripts were executable and had the correct shebang.

1. Ran the Hadoop Streaming job:



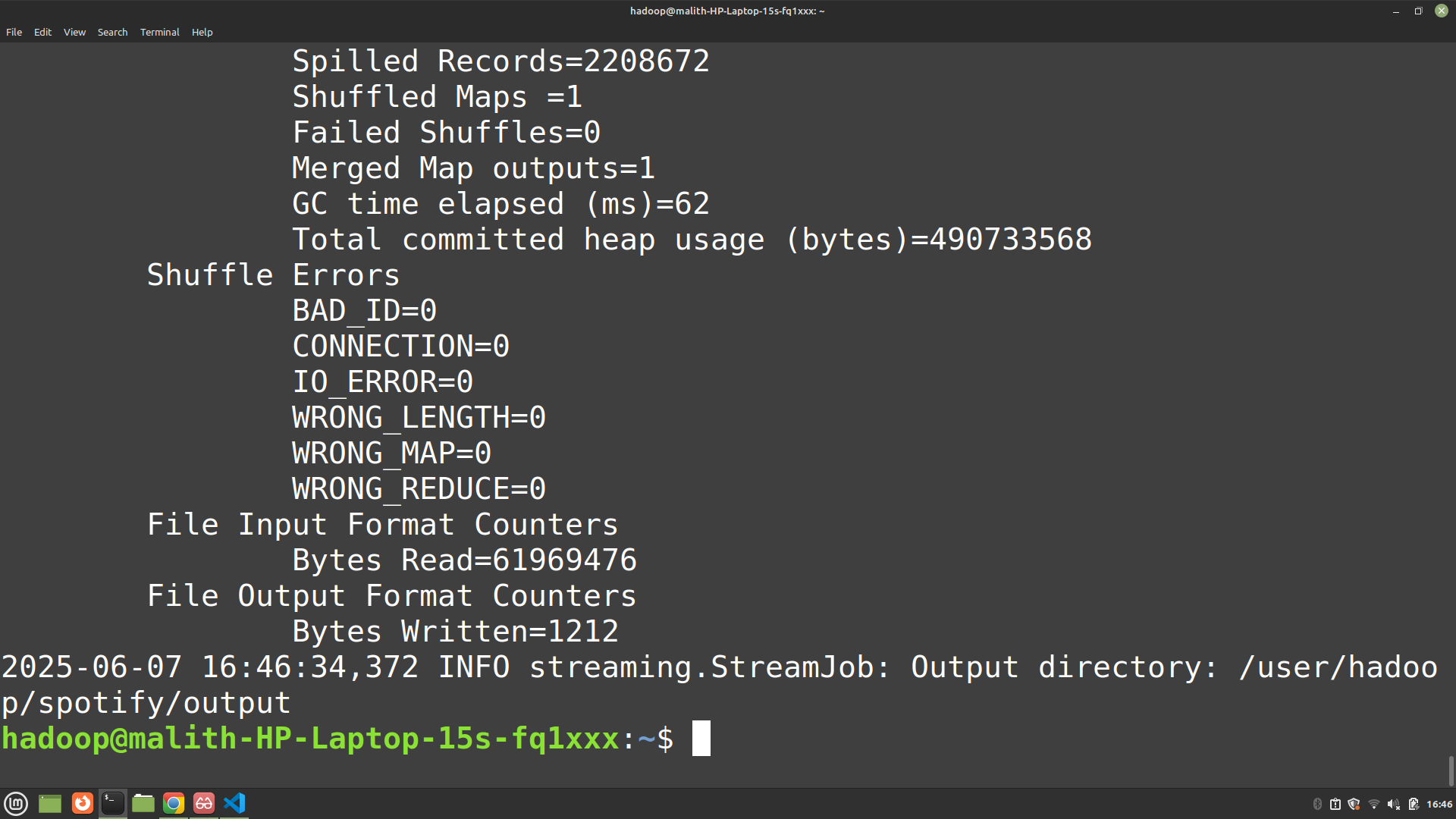
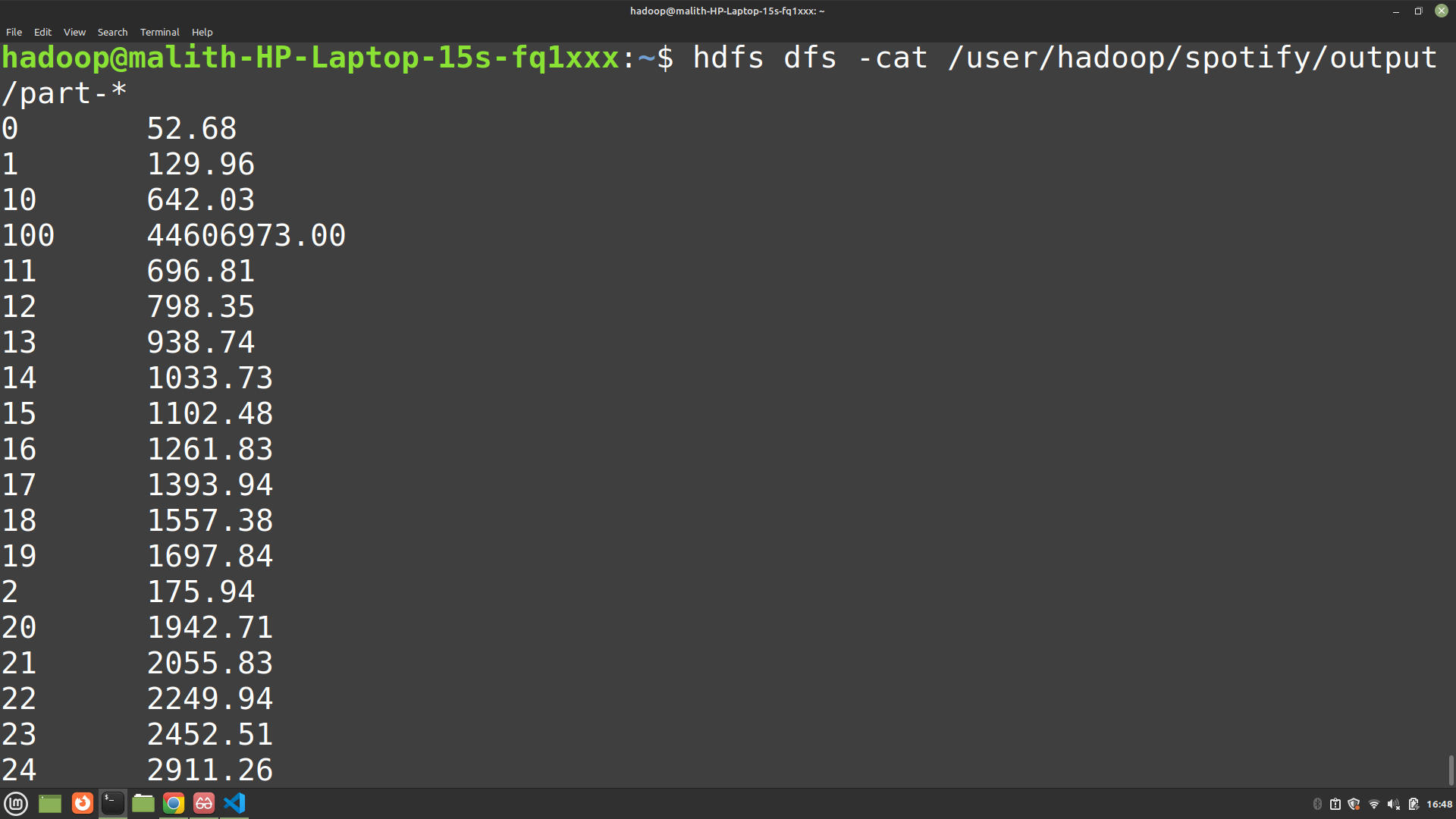


Figure 14: Evidence for Ran the Hadoop Streaming Job

1. Retrieved and viewed the output:



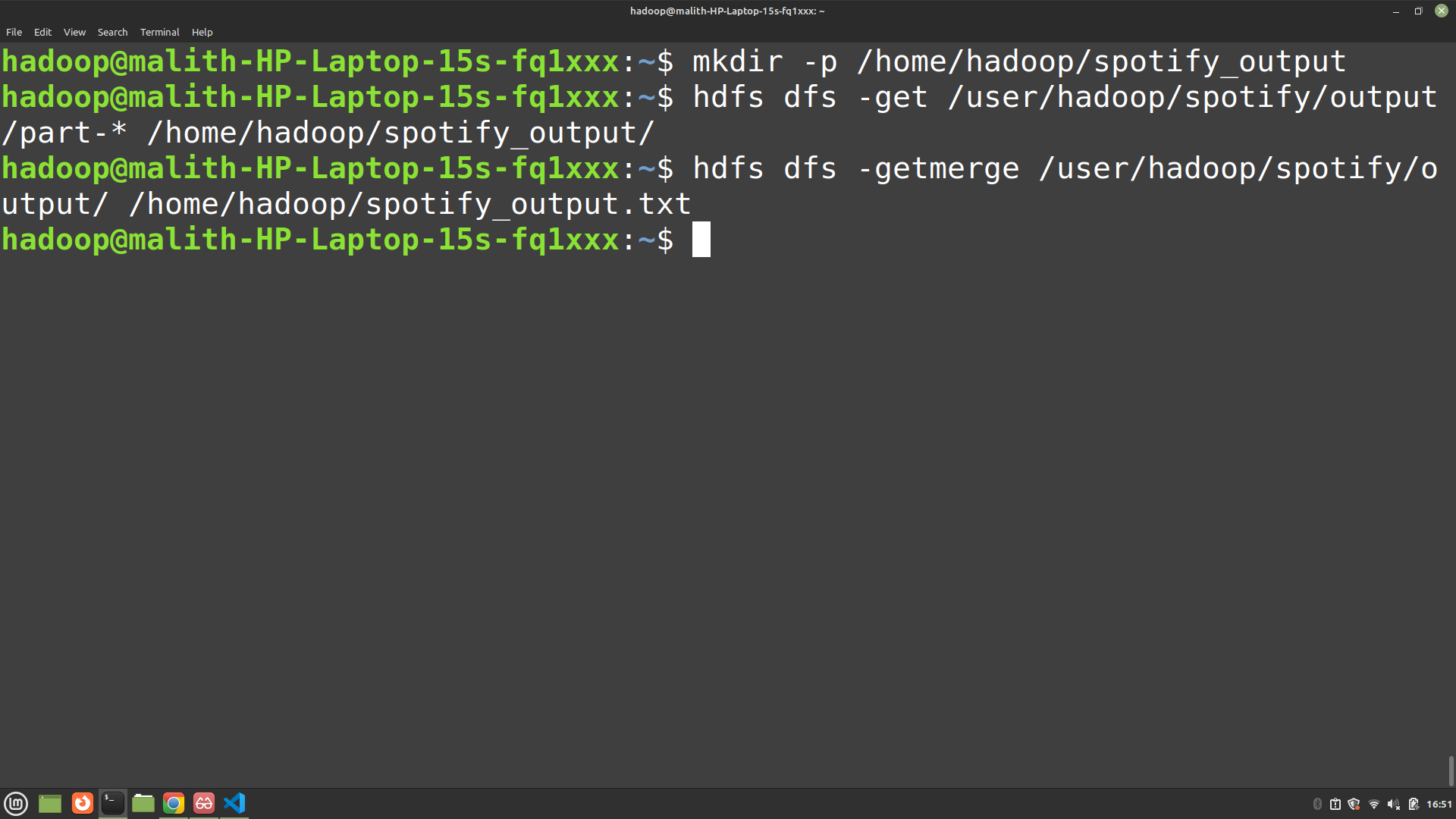


Figure 15: Evidence for Retrieved and viewed the output

1. Input/Output Samples:

* Input (CSV):

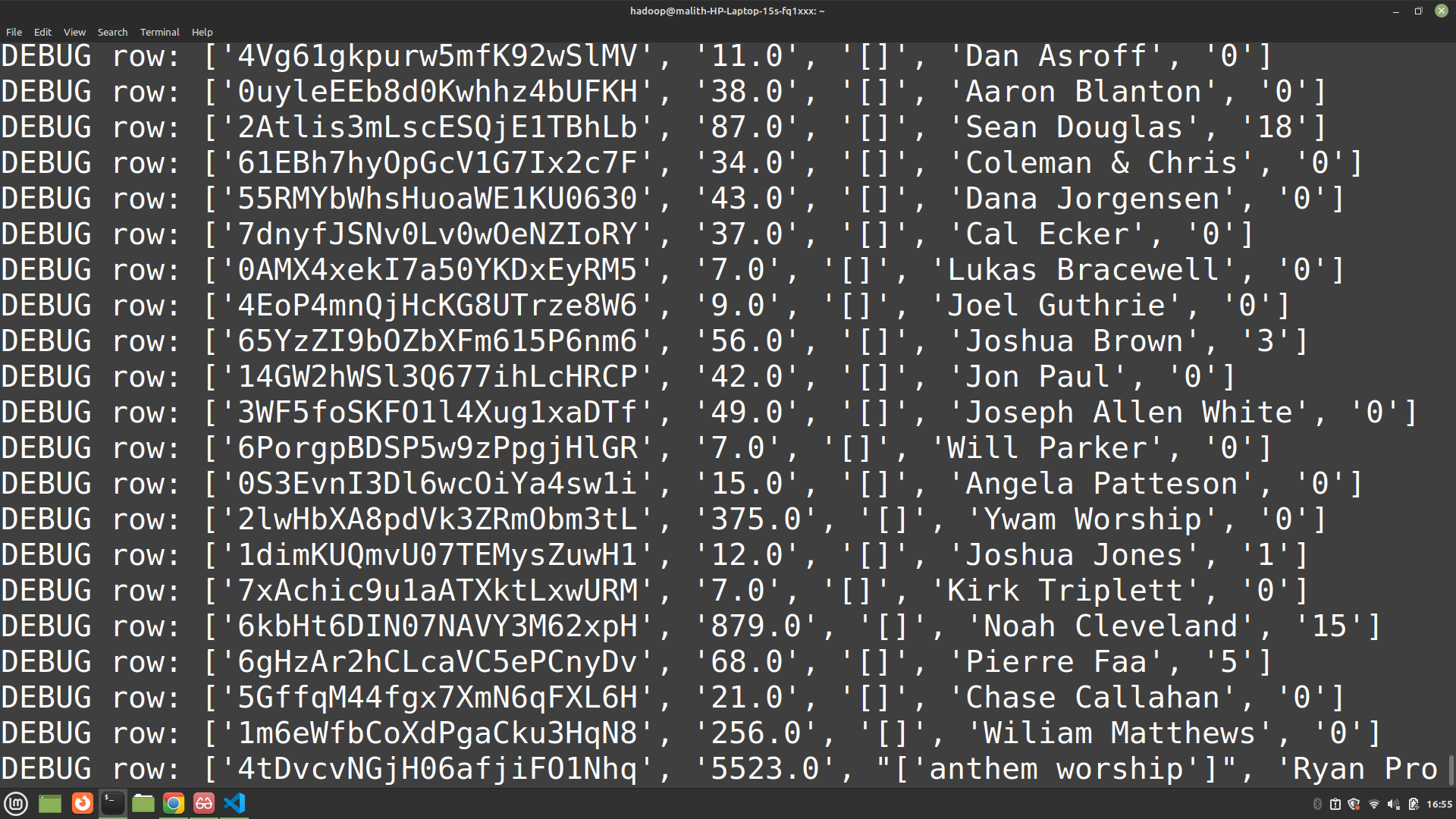


Figure 16: Evidence for Input

* Output:



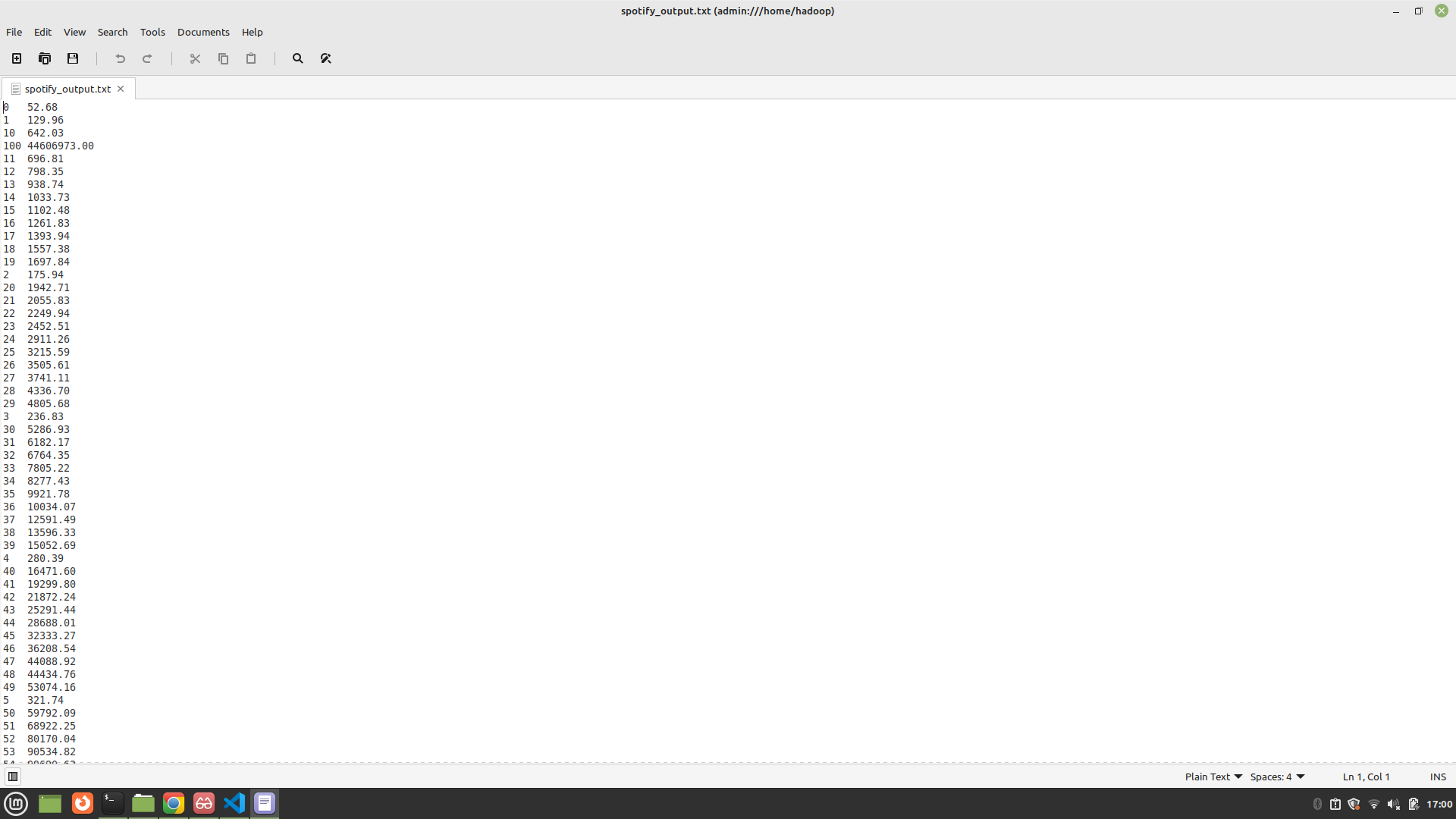


Figure 17: Evidence for Output

# Results & Interpretation

Key Findings:

* The MapReduce job successfully computed the average number of followers for each popularity score among Spotify artists. The results show that higher popularity scores generally correspond to higher average follower counts, with some variation at the lower and upper ends of the popularity spectrum.

Patterns and Insights:

* Most artists with low popularity scores have relatively few followers, but there are exceptions.
* As popularity increases, the average follower count also increases, indicating a positive correlation between these two metrics.
* Some popularity scores have outliers with extremely high follower counts, skewing their averages.

Performance:

* The job processed a large dataset efficiently using Hadoop Streaming and Python scripts. No critical bottlenecks were observed, and the output was produced in a reasonable timeframe.

Improvements/Future Work:

* Extend the analysis to groups by genres or countries if such data is available.
* Visualize the popularity-follower relationship with graphs.
* Handle missing or anomalous data more robustly.

# Conclusion

This assignment provided hands-on experience with Hadoop MapReduce and large-scale data analysis using Python and Hadoop Streaming. We learned how to preprocess data, write custom mapper and reducer scripts, and interpret distributed computation results. The process reinforced the importance of data cleaning, correct script permissions, and debugging skills when working with big data tools. Overall, the project deepened my understanding of distributed computing and its application to real-world datasets like Spotify’s artist metadata.

# References

|  |  |
| --- | --- |
| [1] | L. Neziri, "Spotify Datasets," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/lehaknarnauli/spotify-datasets. [Accessed 7 June 2025]. |