# Layoff Data Analysis Using Hadoop MapReduce

# Table of Contents

## 1. Introduction

Over 300 million terabytes of data are generated in today's world on a day-to-day basis. Large corporations and institutions are utilising the data to enhance the products and services they provide. These businesses can forecast their goals and understand the market's upcoming trend with the use of the data. They manage to reduce the possibility of a product failing as a result, which eventually improves growth.

Humby, C. (2006). "Data is the new oil." Data processing is the purification process to mine useful data from the raw pool of data, which can be utilised in a variety of ways. In this assessment, I have conducted an analysis of industries laying off their employees, which can help us understand the trend and market demand, technological advancements, or shifts in consumer preferences.

The main aim of this project is to design, implement, and examine a distributed analytical technique using Hadoop MapReduce and Hadoop Distribution File System (HDFS), and the mapreduce algorithms are written in Python programming language in order to find a solution to a real-world data processing problem related to the number of layoffs in the industry. Additionally, the results and discussion of the outcome will provide insightful information.

## 2. Problem description, Dataset and Research question

This analytical analysis will help us understand the industry workforce reduction in terms of economic conditions, industry health, labour market dynamics, company performances, and many other key aspects, and this industry layoff case study analysis will also help the employees or future aspirants prepare accordingly in a particular sector or field.

The Hadoop Mapreduce distributed analytical technique makes it very easy to complete the task quickly. With the Hadoop Distributed File System (HDFS), the large data gets divided into different nodes that work simultaneously on the same problem, which is more productive and faster than any linear system.

The dataset taken for this assessment is taken from a site named Layoff.fyi, and was developed by Roger Lee, a Harvard graduate and internet entrepreneur, which contains the layoff information of major companies from 2020 until 2024 (till February 2024). The website contains information on a number of topics, including the company's role as a resource for anyone interested in keeping tabs on layoffs in the sector and its assistance programme, which offers job openings and industry trends to those impacted.

The CSV file for the data processing has been taken from the Kaggle website: https://www.kaggle.com/datasets/theakhilb/layoffs-data-2022/data, which has been adopted from Layoff.fyi using the scraping method. Table 1 contains the variable descriptions of the columns in the dataset.

| S.No | Column Name | Data Type | Description |
|---|---|---|---|
| 1 | Company | String | Name of the company that has conducted layoffs. |
| 2 | Location HQ | String | The location indicates the headquarters location of the company, i.e. London, Paris, New York, etc. |
| 3 | Industry | String | This column classifies the company as their sectors: Retaill, Finance, Healthcare, Travel, etc. |
| 4 | Percentage | Float | It displays the percentage of employees laid off relative to the total employees. |
| 5 | Date | Date | This column defines the date on which the layoffs were announced or took place.<br>i.e., from 2020-03-11 to 2024-02-29 |
| 6 | Source | Abstract | This column identifies the source of information regarding the layoffs. |
| 7 | Funds Raised | Float | This column indicates the total amount of funding the company has raised from layoffs. |
| 8 | Stage | String | This column describes the stage of development |
| 9 | Date_Added | Date | This column tells the date and time of the information added to the table. |
| 10 | Country | String | This column specifies the country in which the layoffs occurred.<br>i.e. United States, Canada, India, etc. |
| 11 | Laid_Off_Count | Integer | This column indicates the number of employees that have been laid off. |
| 12 | List_of_Employees_Laid_Off | Abstract | This is the list of the company’s layoff notifications. |

Table 1: Variable Descriptions

The original data set contains 3486 records and 12 variables. However, since COVID struck, both the global economy and layoffs have fluctuated. One never knows when their jobs may end in this uncertain environment.

As a data science student, I would like to analyse the number of layoffs that happened in the year 2023 in different industries throughout the world. Thus, the research question for this project is:

***What are the potential applications of Hadoop MapReduce in the analysis of layoff data and the extraction of industry-specific information regarding layoff trends in the year 2023?***

## 3. MapReduce Design and Implementation

A programming methodology and framework called Hadoop MapReduce is used to analyse and generate massive datasets across several distributed nodes in parallel. It is an essential part of the Apache Hadoop project, an open-source software platform for large data processing and distributed storage.

Mapreduce works in three steps:

A. **Map Phase:** The map phase divides the data into various nodes into smaller chunks and distributes them along the cluster of nodes. Each node processes the assigned portion of data using a map function.

B. **Shuffle and Sort:** In this step, the output of the map function is shuffled and sorted on the basis of keys, and then the output gets transferred to the reducers.

C. **Reduce Phase:** The reduce phase collects the value corresponding to each key, usually completing a computation or summary. The output is saved in the Hadoop Distributed File System, or it can be directed to other storage systems.

Figure 1 shows the flow diagram of the Hadoop MapReduce function in terms of an example to get a better understanding of the mapreduce process.

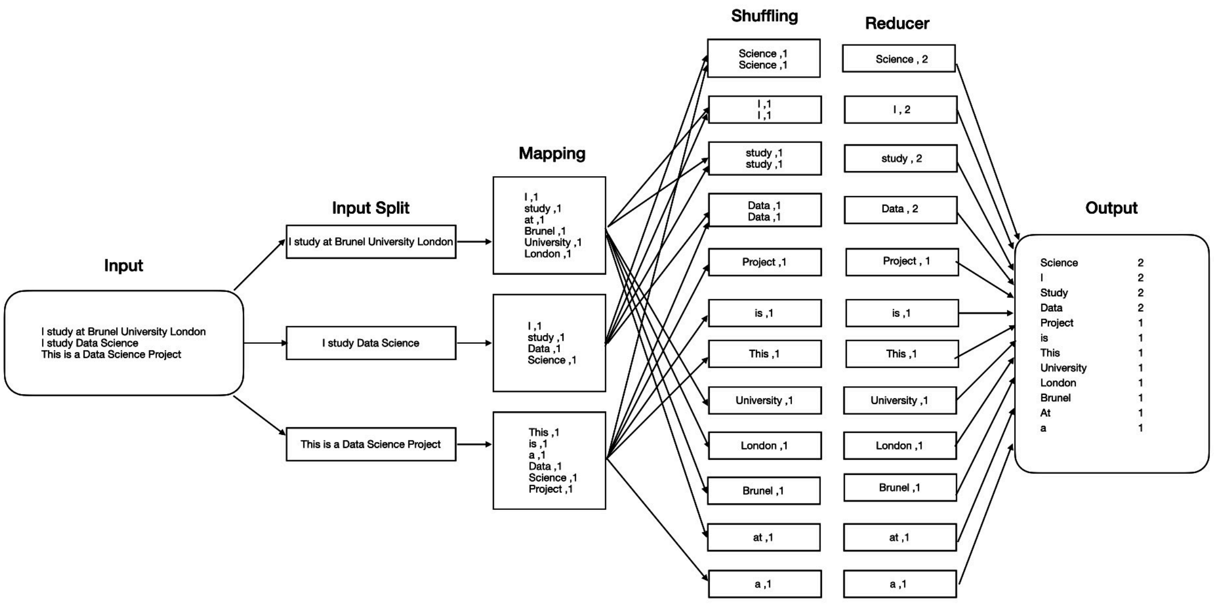


Figure 1: MapReduce Flow Diagram

The following subheading contains the process, the descriptions of the process, and the functioning of the code that is happening in the project to analyse and get insight from the data set.

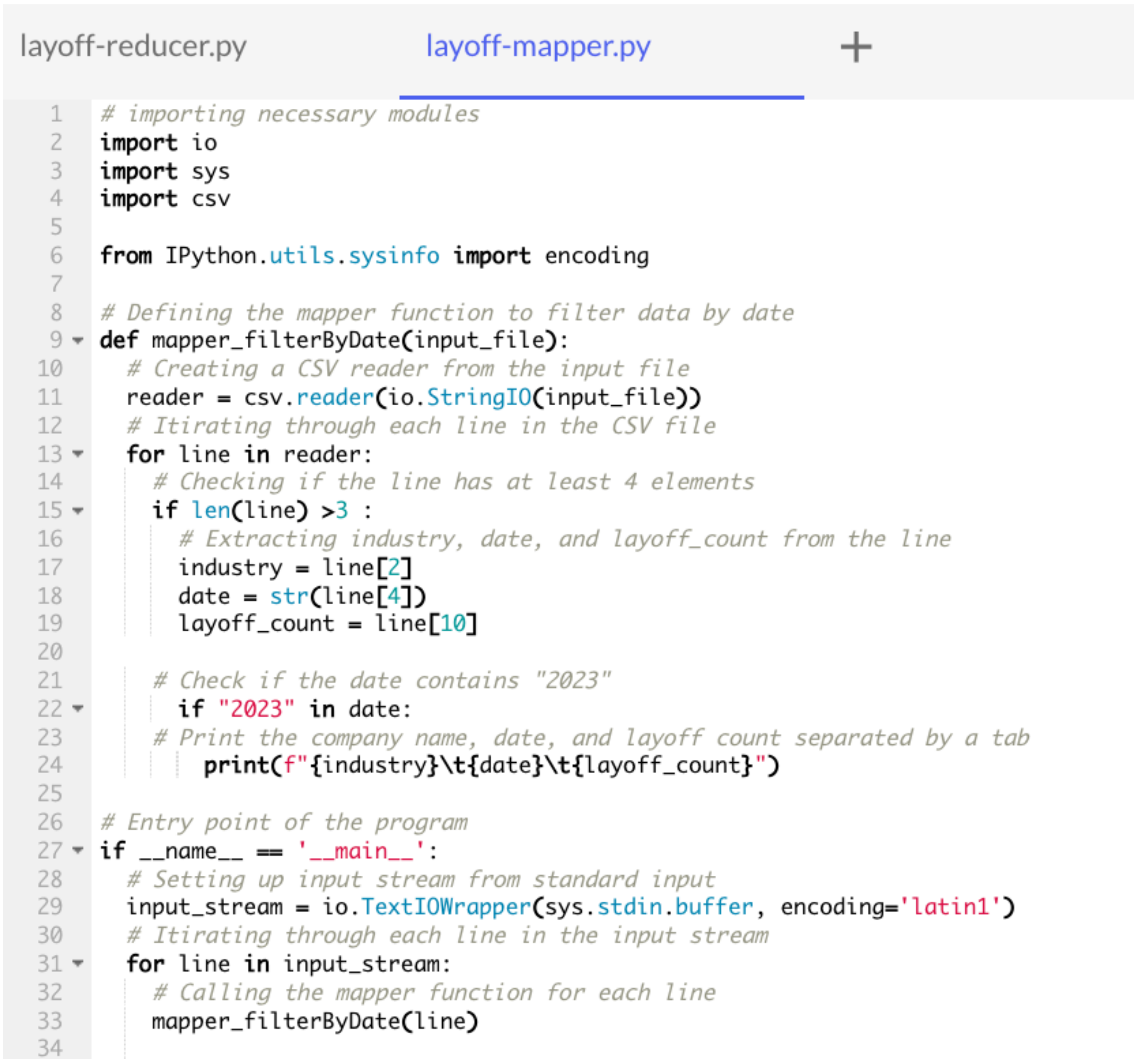
## 3.1. MapReduce Design

### 3.1.1. Mapper Program:

This section explains the mapping phase that has been applied to the case study in order to get insightful information from the layoff case study data. The mapper divides the inputs and uses a key to shuffle them, as we already know.

The Python programming language is used to write the layoff-mapper code, and I have built a function that takes the **layoff_count**, **industry**, and **date** from the **layoff_data** dataset. The code will shuffle and sort the year "**2023**" from the date as a key, and as soon as the function completes, it returns the data to the layoff-reducer of the mapreduce function.\

Figure 2 shows a screenshot of the layoff-mapper code, which reads the data set and applies the specified algorithm to the dataset. The code was created especially for the layoff data case study.



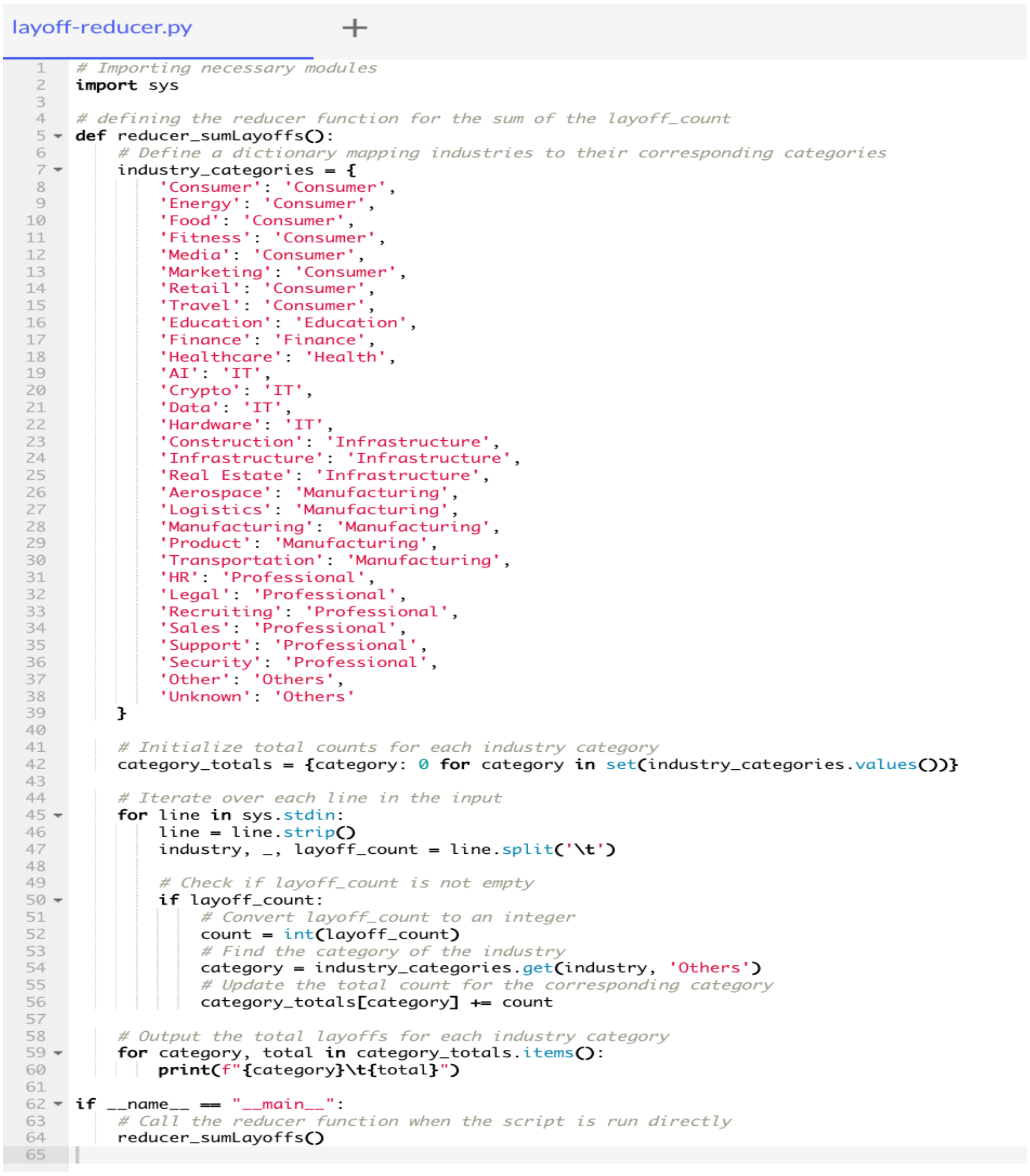
```python
# importing necessary modules
import io
import sys
import csv

from IPython.utils.sysinfo import encoding

# Defining the mapper function to filter data by date
def mapper_filterByDate(input_file):
  # Creating a CSV reader from the input file
  reader = csv.reader(io.StringIO(input_file))
  # Itirating through each line in the CSV file
  for line in reader:
    # Checking if the line has at least 4 elements
    if len(line) >3 :
      # Extracting industry, date, and layoff_count from the line
      industry = line[2]
      date = str(line[4])
      layoff_count = line[10]

    # Check if the date contains "2023"
      if "2023" in date:
    # Print the company name, date, and layoff count separated by a tab
        print(f"{industry}\t{date}\t{layoff_count}")

# Entry point of the program
if __name__ == '__main__':
  # Setting up input stream from standard input
  input_stream = io.TextIOWrapper(sys.stdin.buffer, encoding='latin1')
  # Itirating through each line in the input stream
  for line in input_stream:
    # Calling the mapper function for each line
    mapper_filterByDate(line)
```

Figure 2: layoff-mapper

## 3.2. Reducer Program:

Python has also been used to write the code for the reducer. I have built a reducer code that starts with the initialization of total counts for each industry category and concludes with a dictionary mapping industries to their respective categories.

The **layoff-reducer** function receives the <key, value> pair generated by the **layoff-mapper**, which will count the number of industrial layoffs in the respective companies of the sector and give a sum as the industrial layoff in the particular sector or industry.



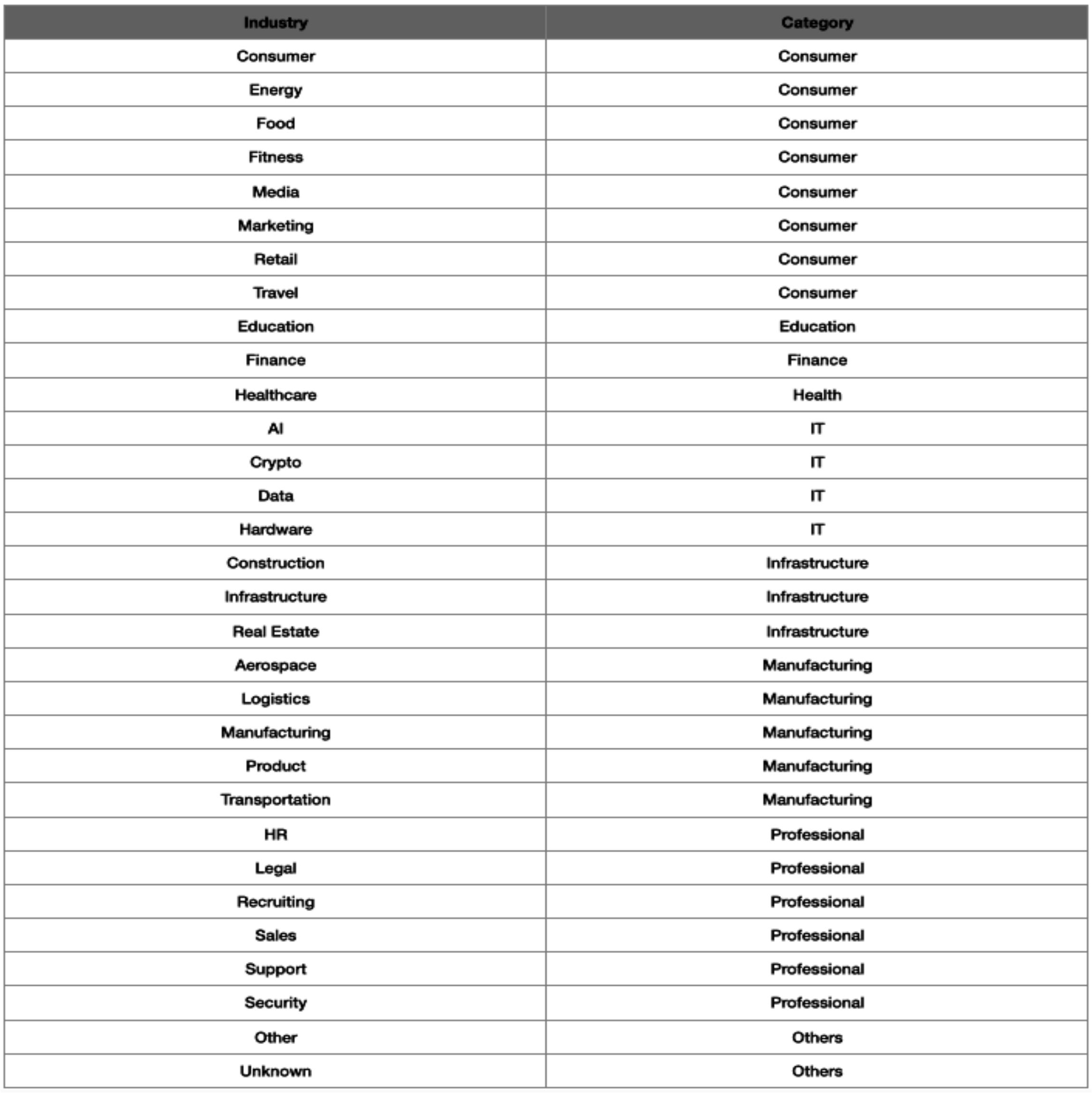
layoff-reducer.py

```python
# Importing necessary modules
import sys

# defining the reducer function for the sum of the layoff_count
def reducer_sumLayoffs():
    # Define a dictionary mapping industries to their corresponding categories
    industry_categories = {
        'Consumer': 'Consumer',
        'Energy': 'Consumer',
        'Food': 'Consumer',
        'Fitness': 'Consumer',
        'Media': 'Consumer',
        'Marketing': 'Consumer',
        'Retail': 'Consumer',
        'Travel': 'Consumer',
        'Education': 'Education',
        'Finance': 'Finance',
        'Healthcare': 'Health',
        'AI': 'IT',
        'Crypto': 'IT',
        'Data': 'IT',
        'Hardware': 'IT',
        'Construction': 'Infrastructure',
        'Infrastructure': 'Infrastructure',
        'Real Estate': 'Infrastructure',
        'Aerospace': 'Manufacturing',
        'Logistics': 'Manufacturing',
        'Manufacturing': 'Manufacturing',
        'Product': 'Manufacturing',
        'Transportation': 'Manufacturing',
        'HR': 'Professional',
        'Legal': 'Professional',
        'Recruiting': 'Professional',
        'Sales': 'Professional',
        'Support': 'Professional',
        'Security': 'Professional',
        'Other': 'Others',
        'Unknown': 'Others'
    }

    # Initialize total counts for each industry category
    category_totals = {category: 0 for category in set(industry_categories.values())}

    # Iterate over each line in the input
    for line in sys.stdin:
        line = line.strip()
        industry, _, layoff_count = line.split('\t')

        # Check if layoff_count is not empty
        if layoff_count:
            # Convert layoff_count to an integer
            count = int(layoff_count)
            # Find the category of the industry
            category = industry_categories.get(industry, 'Others')
            # Update the total count for the corresponding category
            category_totals[category] += count

    # Output the total layoffs for each industry category
    for category, total in category_totals.items():
        print(f"{category}\t{total}")

if __name__ == "__main__":
    # Call the reducer function when the script is run directly
    reducer_sumLayoffs()

```

Figure 3: layoff-reducer

After the completion of the layoff-reducer process, the output is saved in the HDFS as a text file, which gives the count of total layoffs in each industry as per the categories they fall under throughout the year 2023.

The industry categories are specified in the layoff-reducer code as well; the sectors are assigned according to the table that follows.



| Industry | Category |
|---|---|
| Consumer | Consumer |
| Energy | Consumer |
| Food | Consumer |
| Fitness | Consumer |
| Media | Consumer |
| Marketing | Consumer |
| Retail | Consumer |
| Travel | Consumer |
| Education | Education |
| Finance | Finance |
| Healthcare | Health |
| AI | IT |
| Crypto | IT |
| Data | IT |
| Hardware | IT |
| Construction | Infrastructure |
| Infrastructure | Infrastructure |
| Real Estate | Infrastructure |
| Aerospace | Manufacturing |
| Logistics | Manufacturing |
| Manufacturing | Manufacturing |
| Product | Manufacturing |
| Transportation | Manufacturing |
| HR | Professional |
| Legal | Professional |
| Recruiting | Professional |
| Sales | Professional |
| Support | Professional |
| Security | Professional |
| Other | Others |
| Unknown | Others |

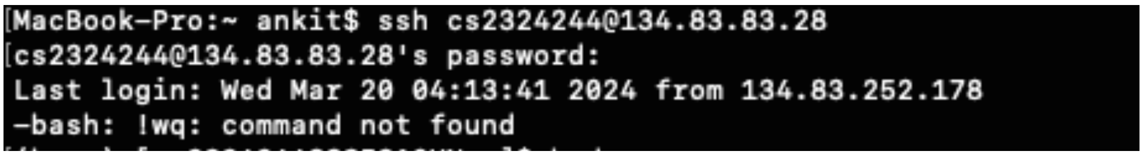
Table 2: Industry Categorisation

## 3.2. Implementation

### A. Implementation using Brunel Remote Access Services (VPN) on Terminal:

Setting up the environment is required before executing the Hadoop software to count the number of layoffs made by industries in 2023 in the layoffs_data.csv file. A brief tutorial on using Hadoop on the server to finish this project is given in this part.

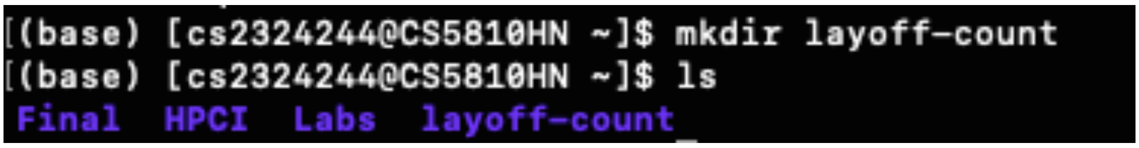
A.1. To begin, open a terminal or CMD and connect to the Brunel server with your username and password using ssh.



```
[MacBook-Pro:~ ankit$ ssh cs2324244@134.83.83.28
[cs2324244@134.83.83.28's password:
 Last login: Wed Mar 20 04:13:41 2024 from 134.83.252.178
 -bash: !wq: command not found
```

Figure 4.1: ssh command

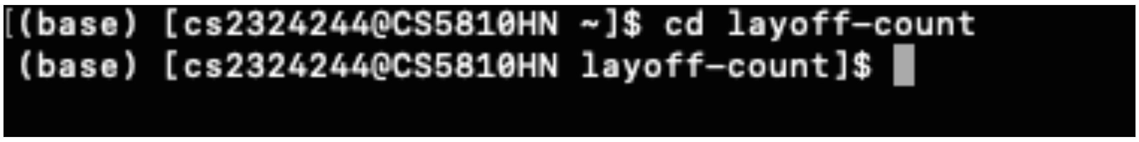
A.2. Creating a folder to save the layoff count and project-related files.



```
[(base) [cs2324244@CS5810HN ~]$ mkdir layoff-count
[(base) [cs2324244@CS5810HN ~]$ ls
 Final   HPCI   Labs   layoff-count
```

Figure 4.2: Creating folders

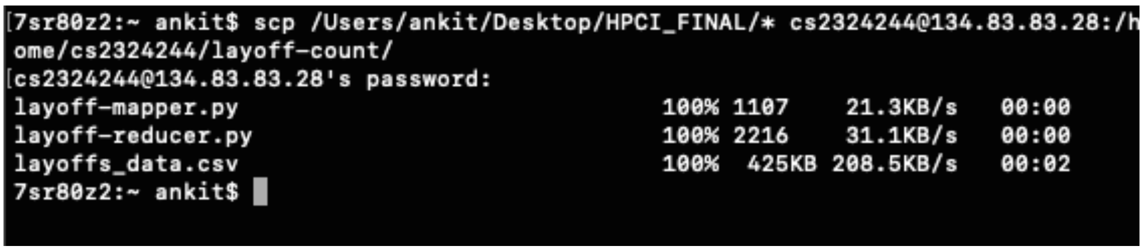
A.3. Change the directory to the layoff count.



```
[(base) [cs2324244@CS5810HN ~]$ cd layoff-count
 (base) [cs2324244@CS5810HN layoff-count]$
```

Figure 4.3: Changing directory

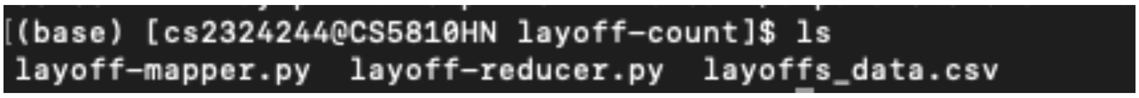
A.4. Open a new Terminal or CMD and use the scp command to copy the files to the server.



```
[7sr80z2:~ ankit$ scp /Users/ankit/Desktop/HPCI_FINAL/* cs2324244@134.83.83.28:/h
 ome/cs2324244/layoff-count/
[cs2324244@134.83.83.28's password:
 layoff-mapper.py                            100% 1107     21.3KB/s   00:00
 layoff-reducer.py                           100% 2216     31.1KB/s   00:00
 layoffs_data.csv                            100%  425KB 208.5KB/s   00:02
 7sr80z2:~ ankit$
```

Figure 4.4: Scp command

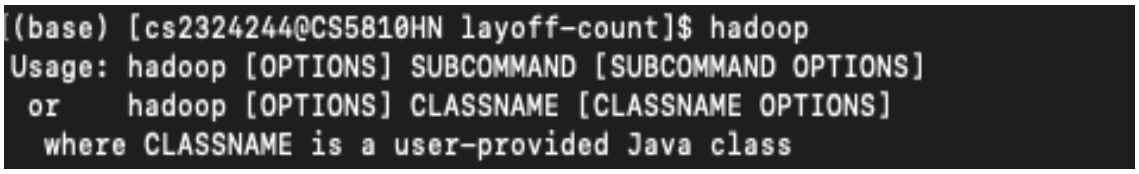
A.5. Use the ls command to list the files in the directory.



```
[(base) [cs2324244@CS5810HN layoff-count]$ ls
layoff-mapper.py  layoff-reducer.py  layoffs_data.csv
```

Figure 4.5: list command

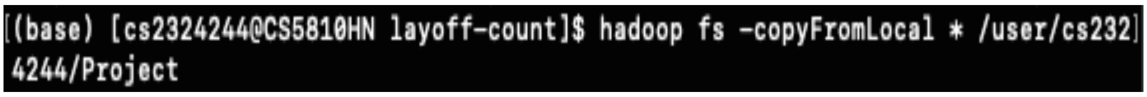
A.6. Use the Hadoop command to activate Hadoop properties, which are pre-installed on the Brunel server.



```
[(base) [cs2324244@CS5810HN layoff-count]$ hadoop
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
 or    hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
  where CLASSNAME is a user-provided Java class
```

Figure 4.6: Hadoop command

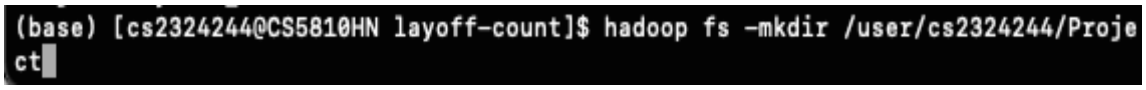
A.7. Copy these files to the Hadoop file system, HDFS.



```
[(base) [cs2324244@CS5810HN layoff-count]$ hadoop fs -copyFromLocal * /user/cs232]
4244/Project
```

Figure 4.7: -copyFromLocal command

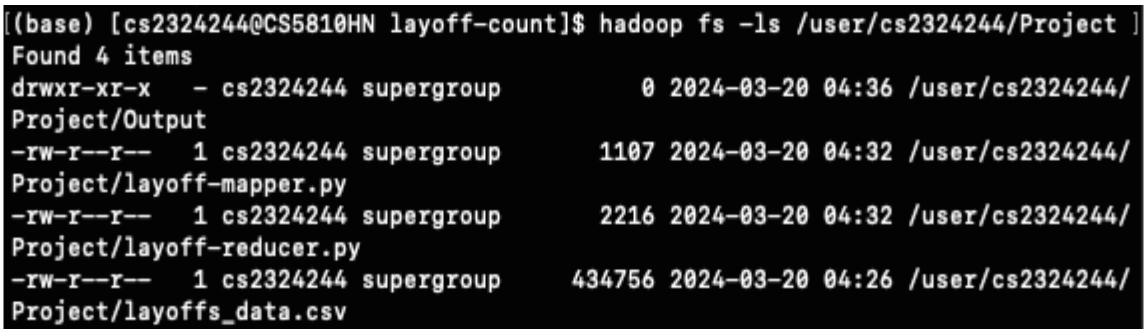
A.8. Make a directory output to save the output data.



```
(base) [cs2324244@CS5810HN layoff-count]$ hadoop fs -mkdir /user/cs2324244/Proje
ct
```

Figure 4.8: mkdir command

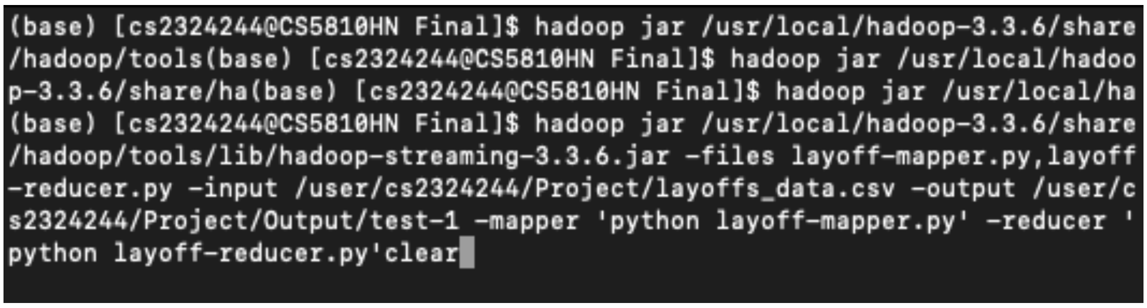
A.9. Check that the files are copied into the Hadoop folder.



```
[(base) [cs2324244@CS5810HN layoff-count]$ hadoop fs -ls /user/cs2324244/Project ]
Found 4 items
drwxr-xr-x   - cs2324244 supergroup          0 2024-03-20 04:36 /user/cs2324244/
Project/Output
-rw-r--r--   1 cs2324244 supergroup       1107 2024-03-20 04:32 /user/cs2324244/
Project/layoff-mapper.py
-rw-r--r--   1 cs2324244 supergroup       2216 2024-03-20 04:32 /user/cs2324244/
Project/layoff-reducer.py
-rw-r--r--   1 cs2324244 supergroup     434756 2024-03-20 04:26 /user/cs2324244/
Project/layoffs_data.csv
```

Figure 4.9: ls command

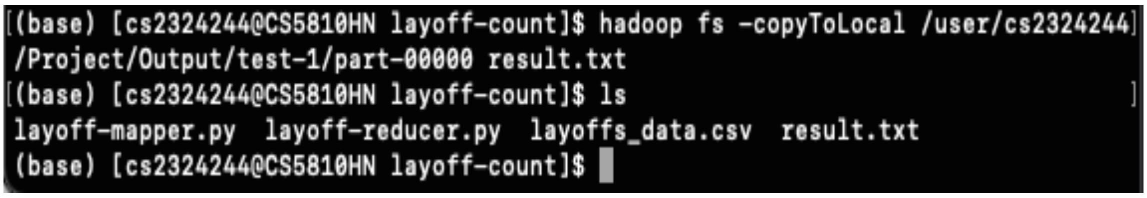
A.10. Now run the mapreduce job using Hadoop streaming. For that, a snippet of the command is attached below.



```
(base) [cs2324244@CS5810HN Final]$ hadoop jar /usr/local/hadoop-3.3.6/share
/hadoop/tools(base) [cs2324244@CS5810HN Final]$ hadoop jar /usr/local/hadoo
p-3.3.6/share/ha(base) [cs2324244@CS5810HN Final]$ hadoop jar /usr/local/ha
(base) [cs2324244@CS5810HN Final]$ hadoop jar /usr/local/hadoop-3.3.6/share
/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files layoff-mapper.py,layoff
-reducer.py -input /user/cs2324244/Project/layoffs_data.csv -output /user/c
s2324244/Project/Output/test-1 -mapper 'python layoff-mapper.py' -reducer '
python layoff-reducer.py'clear
```

Figure 4.10: Running the MapReduce job

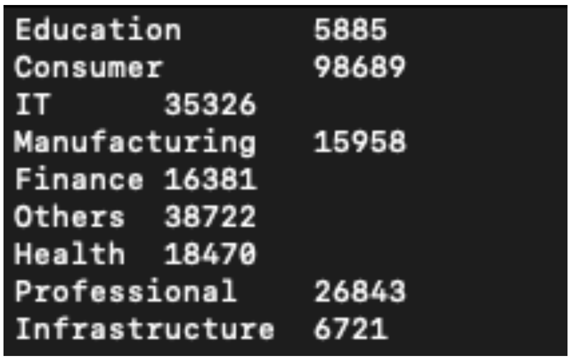
A.11. Copy the result from the Hadoop file system to your local system as a text file, and list the items to check in the file.



```
[(base) [cs2324244@CS5810HN layoff-count]$ hadoop fs -copyToLocal /user/cs2324244]
/Project/Output/test-1/part-00000 result.txt
[(base) [cs2324244@CS5810HN layoff-count]$ ls                                    ]
layoff-mapper.py  layoff-reducer.py  layoffs_data.csv  result.txt
(base) [cs2324244@CS5810HN layoff-count]$
```

Figure 4.11: Copying the result from the Hadoop file system to local system

A.12. Use the “less” command to visualise the result file. To exit less, press ‘q’.



```
Education       5885
Consumer        98689
IT      35326
Manufacturing   15958
Finance 16381
Others  38722
Health  18470
Professional    26843
Infrastructure  6721
```

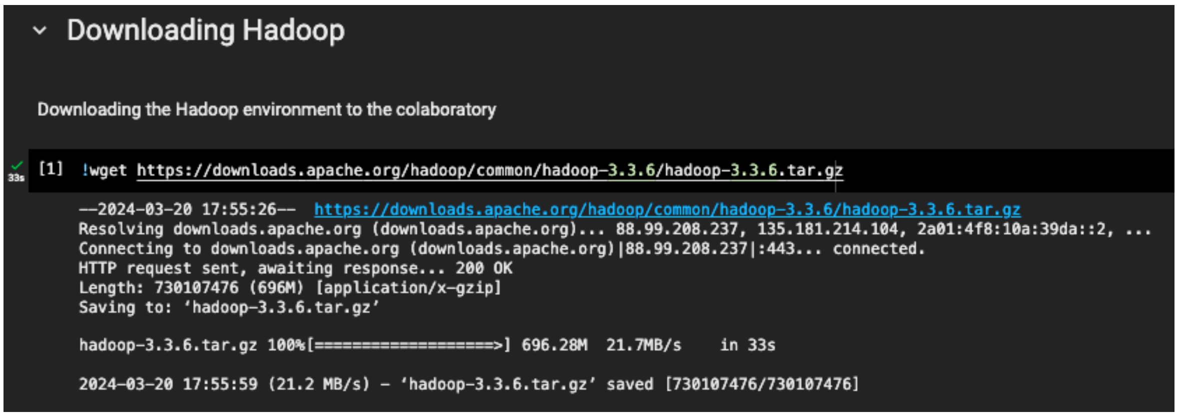
Figure 4.12: result.txt

## B. Implementation using Google Colaboratory Notebook:

For this, we will open a web browser to run the MapReduce algorithm using Python and Google Colab.

B.1. Open a new notebook on Google Colab.

B.2. Download the Hadoop environment to the colaboratory notebook.



Downloading Hadoop

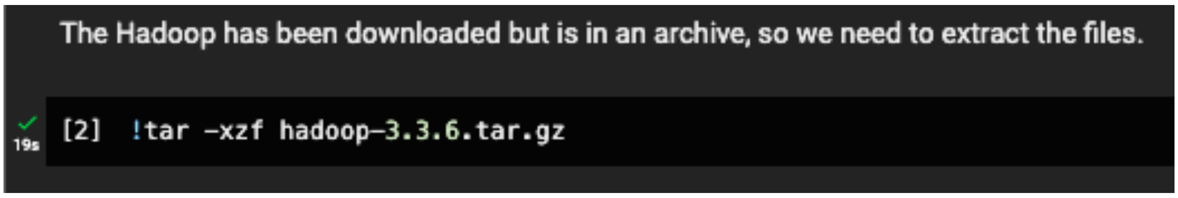
Downloading the Hadoop environment to the colaboratory

```
[1]  !wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
```

```
--2024-03-20 17:55:26--  https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.208.237, 135.181.214.104, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|88.99.208.237|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 730107476 (696M) [application/x-gzip]
Saving to: 'hadoop-3.3.6.tar.gz'

hadoop-3.3.6.tar.gz 100%[===================>] 696.28M  21.7MB/s    in 33s

2024-03-20 17:55:59 (21.2 MB/s) - 'hadoop-3.3.6.tar.gz' saved [730107476/730107476]
```

Figure 5.1: Downloading Hadoop environment

B.3. The Hadoop has been downloaded but is in an archive, so we need to extract the files.

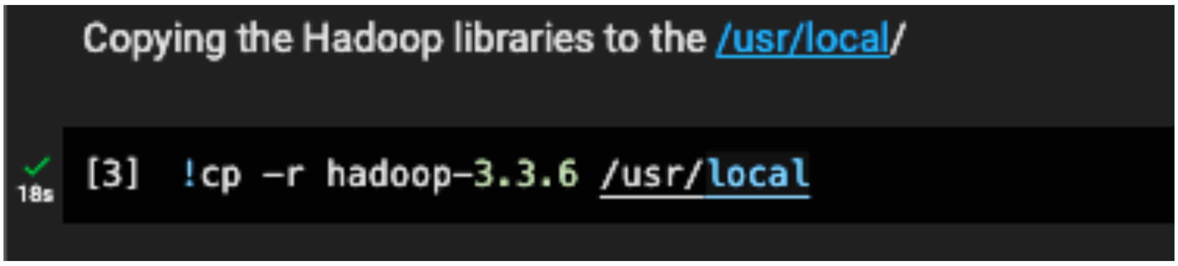


The Hadoop has been downloaded but is in an archive, so we need to extract the files.

```
[2]  !tar -xzf hadoop-3.3.6.tar.gz
```

Figure 5.2: extracting the Hadoop files

B.4. Copying the Hadoop libraries to the /usr/local/

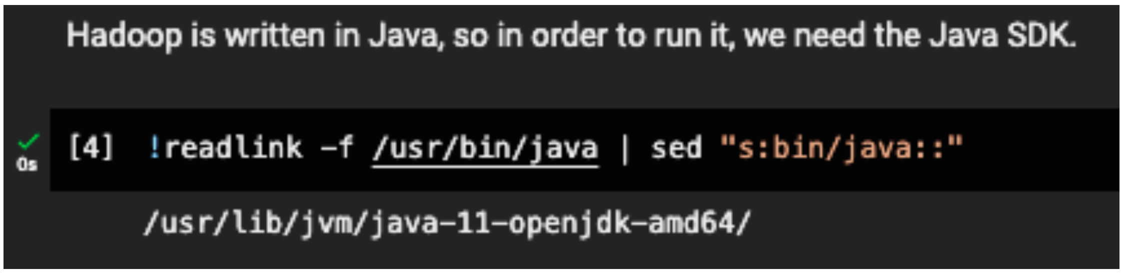


Copying the Hadoop libraries to the /usr/local/

```
[3]  !cp -r hadoop-3.3.6 /usr/local
```

Figure 5.3: Copying the Hadoop libraries

B.5. Hadoop is written in Java, so in order to run it, we need the Java SDK.

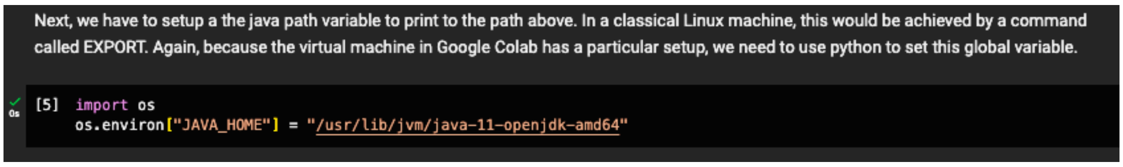


Hadoop is written in Java, so in order to run it, we need the Java SDK.

```
[4] !readlink -f /usr/bin/java | sed "s:bin/java::"
    /usr/lib/jvm/java-11-openjdk-amd64/
```

Figure 5.4: getting the java SDK file

B.6. Next, we have to setup the java path variable to print to the path above. In a classical Linux machine, this would be achieved by a command called EXPORT. Again, because the virtual machine in Google Colab has a particular setup, we need to use Python to set this global variable.

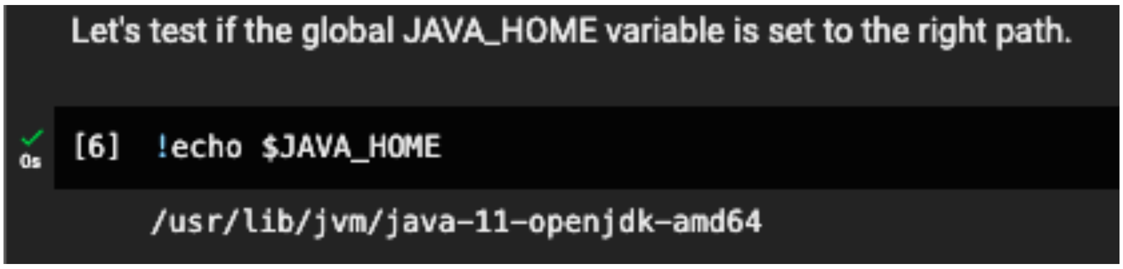


Next, we have to setup a the java path variable to print to the path above. In a classical Linux machine, this would be achieved by a command called EXPORT. Again, because the virtual machine in Google Colab has a particular setup, we need to use python to set this global variable.

```
[5] import os
    os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"
```

Figure 5.5: Importing java environment

B.7. Let's test if the global JAVA_HOME variable is set to the right path.

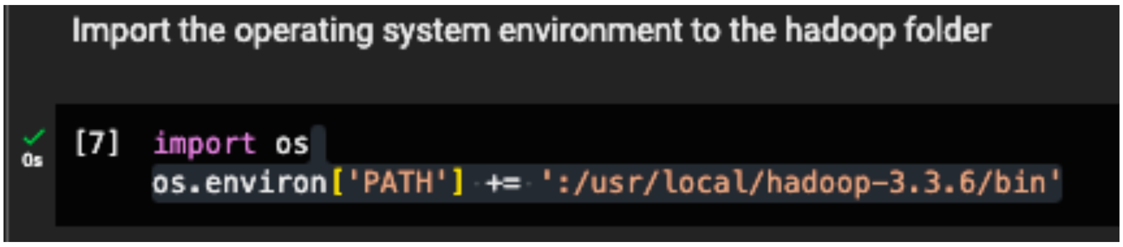


Let's test if the global JAVA_HOME variable is set to the right path.

```
[6] !echo $JAVA_HOME
    /usr/lib/jvm/java-11-openjdk-amd64
```

Figure 5.6: Checking the Java_Home variable

B.8. Import the operating system environment to the Hadoop folder.

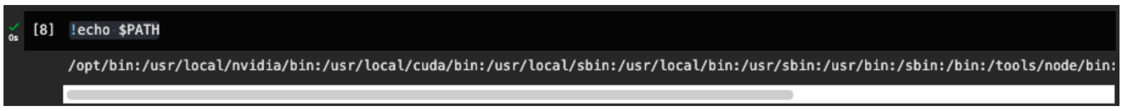


Import the operating system environment to the hadoop folder

```
[7] import os
    os.environ['PATH'] += ':/usr/local/hadoop-3.3.6/bin'
```

Figure 5.7: Importing the os to the Hadoop folder

B.9. Use the !echo command to check the path of the environment.

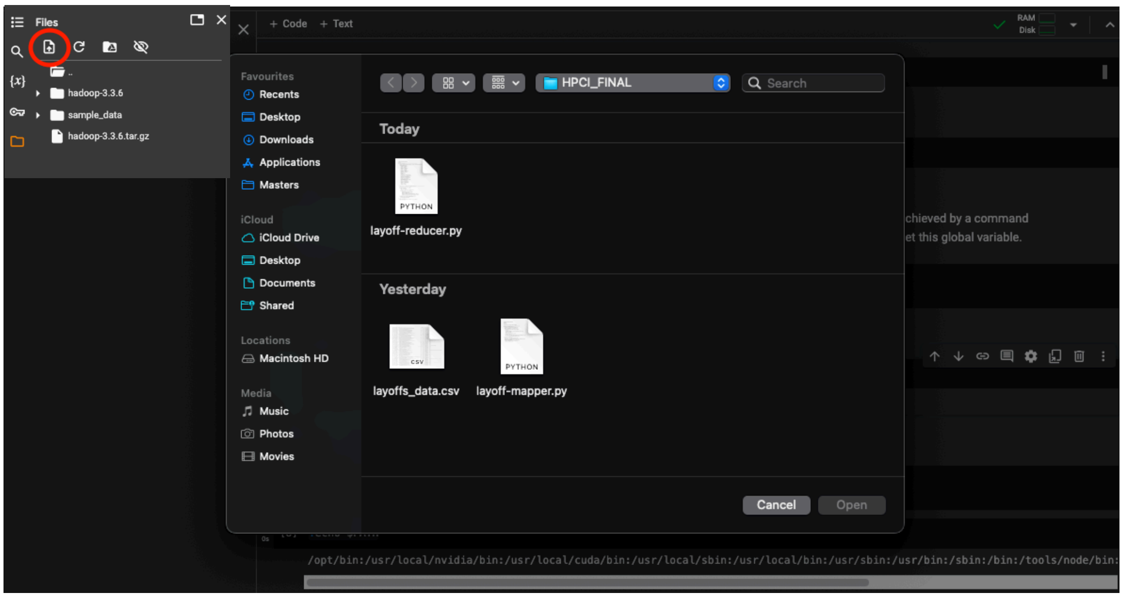


```
[8] !echo $PATH
/opt/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/bin:
```

Figure 5.8: Checking the path of the environment

As we have completed downloading Hadoop and its properties to the environment, we will now move forward to uploading the files from the system and running MapReduce.

B.10. Upload the layoffs-data.csv, layoff-mapper.py, and layoff-reducer.py files from the desktop to the content folder of the Hadoop environment.



```
/opt/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/bin:
```

Figure 5.9: Uploading the files to the Hadoop folder

B.11. The uploaded files will be shown in the Files.

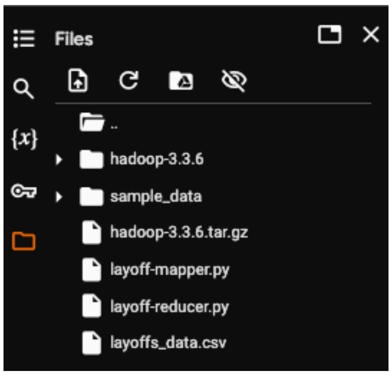
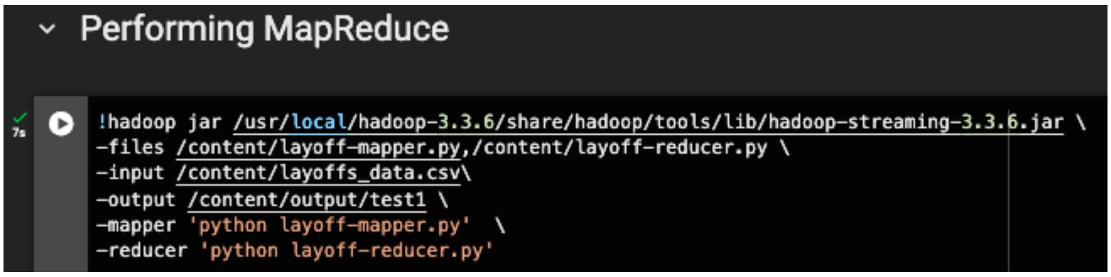


Figure 5.10: Uploaded files location

B.12. Performing MapReduce with the following code.



Performing MapReduce

```
!hadoop jar /usr/local/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
-files /content/layoff-mapper.py,/content/layoff-reducer.py \
-input /content/layoffs_data.csv\
-output /content/output/test1 \
-mapper 'python layoff-mapper.py'  \
-reducer 'python layoff-reducer.py'
```

Figure 5.11: Performing MapReduce

B.13. The output/test1 contains the part-00000 text file of the result performed by the MapReduce.

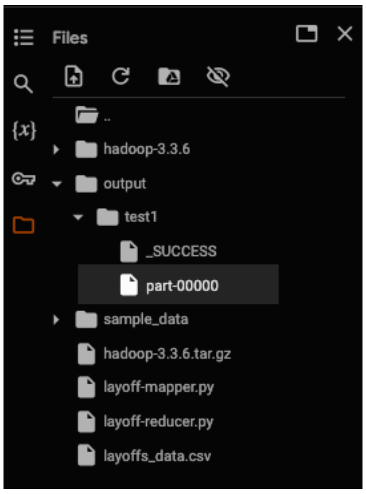
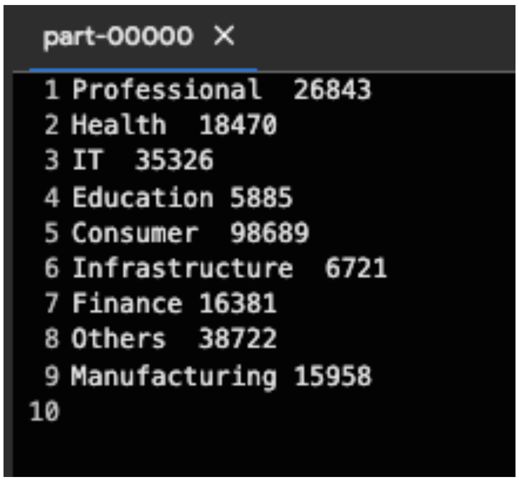


Figure 5.12: Output folder creation
after the code execution

B.14. The output comes in this format on the colaboratory page.



part-00000

```
Professional	26843
Health	18470
IT	35326
Education	5885
Consumer	98689
Infrastructure	6721
Finance	16381
Others	38722
Manufacturing	15958
```

Figure 5.13: part-00000 file which contains the
result of the MapReduce

The output is a list of the industries and the count of layoffs done in their sector throughout the year 2023.

## 4. Result and Evaluation

Implementing the MapReduce framework for the analysis of layoff data yielded valuable insights into layoff patterns across industries. The analysis uncovered several significant conclusions:

4.1. Consumer Sectors: A significant number of layoffs in industries directly related to consumers were found in the investigation, indicating possible difficulties for businesses that meet the demands and preferences of their clientele. This might be a sign of changes in consumer behaviour, recessions, or other issues affecting companies that deal with customers.

4.2. Finance and Healthcare Sectors: The banking and healthcare industries have experienced large layoffs, which highlights how these businesses are impacted by changes in regulations and the state of the economy. Reductions in the workforce in these sectors may have been influenced by regulatory uncertainty, healthcare changes, and unstable economies.

4.3. IT Sectors: Layoffs were common in a number of IT-related industries, such as bitcoin, data, and artificial intelligence (AI). This tendency is a reflection of the dynamic nature of technology-driven businesses, where workforce reorganisations and restructuring initiatives may be necessary due to market swings, quick advancements, and changing priorities.

4.4. Manufacturing and Infrastructure: Moderate layoffs in the manufacturing and infrastructure sectors were detected by the research, indicating possible disruptions in supply chains and infrastructure projects. The labour downsizing in these industries may have been impacted by economic factors, technical improvements, and global market dynamics.

4.5. Professional Services: Professional services industries like sales, human resources (HR), and law have experienced layoffs; these industries demonstrate the wider effects of organisational restructuring. In these service-oriented businesses, staff cutbacks may have resulted from shifts in business strategies, increased market competition, and efficiency initiatives.

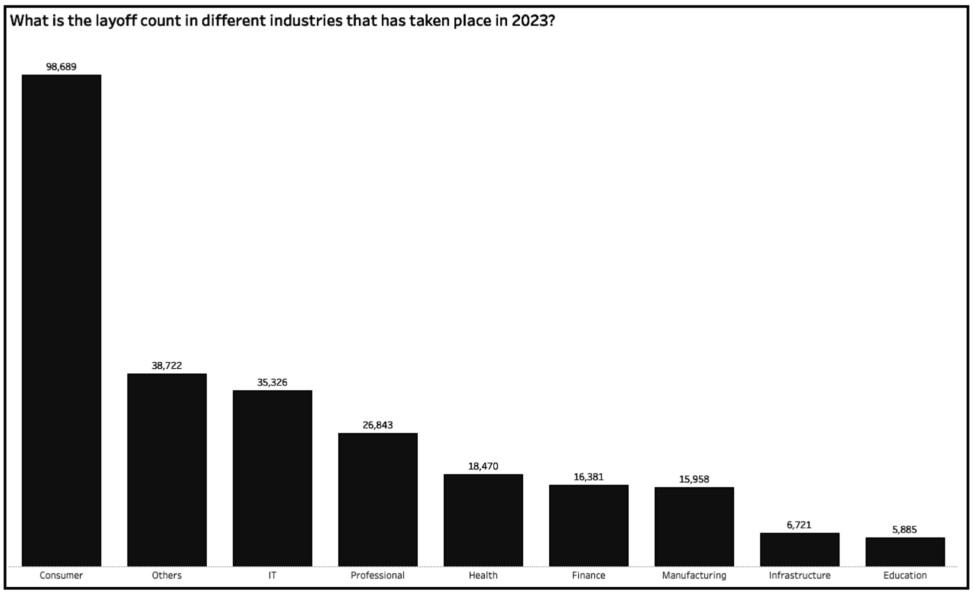


Figure 6: Layoff graph in different industries

The insights obtained from MapReduce align with the analysis offered by the graphs created with Tableau software as a point of reference to verify the accuracy of the MapReduce result, and it demonstrates the similarity and reliability for using big data analysis for more efficient and accurate purposes.

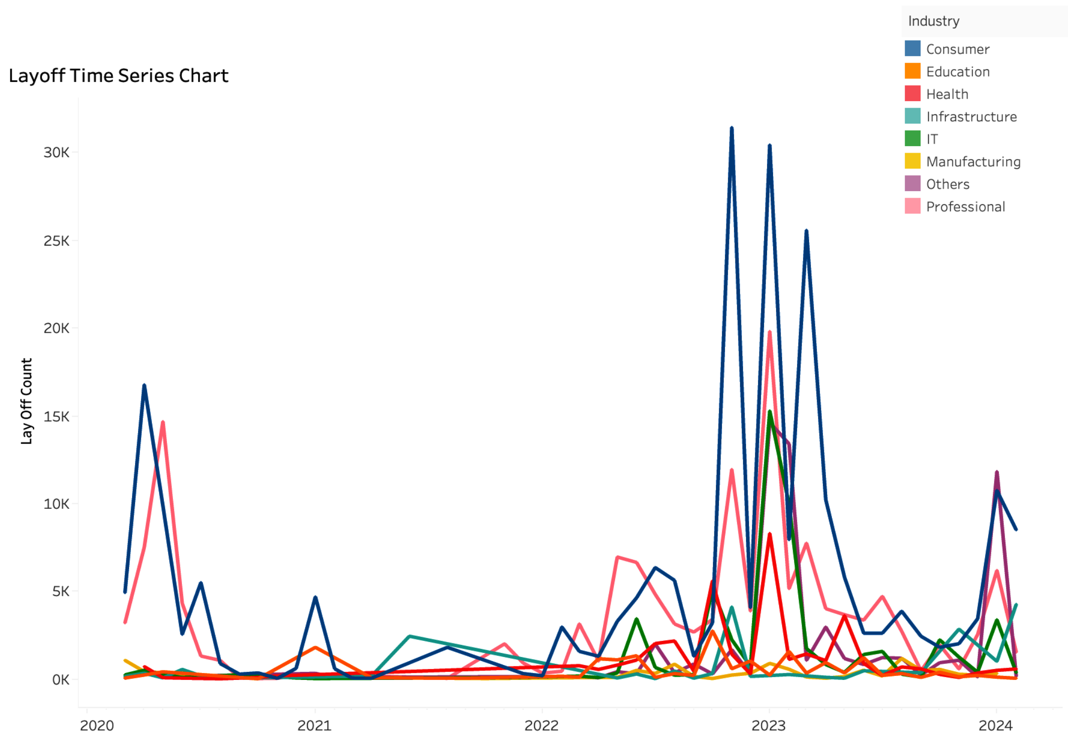


Figure 7: Layoff Time Series Graph

All things considered, the study offers insightful information about how layoffs are distributed and how big they are in different companies and sectors. The aforementioned data can provide valuable insights for strategic decision-making procedures and policy development initiatives that target workforce issues, enhance economic adaptability, and advance sustainable development. Businesses, legislators, and other stakeholders can better understand market trends and carry out focused interventions to lessen the effects of layoffs and boost worker development programmes by utilising data-driven insights from MapReduce analysis.

## 5. Critical Reflection

For the study of layoff data, Hadoop MapReduce offers several advantages, including fault tolerance, parallel processing, and scalability. However, this approach has its limitations and challenges as well. One major obstacle is the complexity of designing and implementing MapReduce algorithms, especially for non-programmers. Moreover, the distribution of data and cluster configuration may affect the speed at which MapReduce jobs are processed, hence influencing real-time analytics and decision-making.

Furthermore, the accuracy and dependability of the analysis are significantly influenced by the completeness and quality of the incoming data. Incomplete or inaccurate data might lead to biassed findings and conclusions. Preprocessing and data purification are therefore essential to ensure the integrity of the analysis.

Despite these challenges, the MapReduce framework is still a useful tool for managing and processing large datasets, including data from layoffs. To further enhance the effectiveness and efficiency of analytics based on MapReduce, further advanced techniques and optimisations can be explored in future studies. Machine learning techniques, for example, can be used to forecast models and identify abnormalities such as decision tree, random forests, neural networks and etc.

To increase the models' capacity for prediction, future research can concentrate on including more dynamic information, such as real-time market data or economic indicators. Furthermore, putting these models to use in a production setting where they can continuously forecast could greatly improve workforce management and strategic planning.

Finally, this research demonstrates how layoff data analysis using Hadoop MapReduce can yield valuable insights about market patterns. By employing this technology, companies, policymakers, and scholars can gain a deeper understanding of the dynamics of the labour market and make well-informed decisions to address opportunities and challenges in the evolving economic landscape.