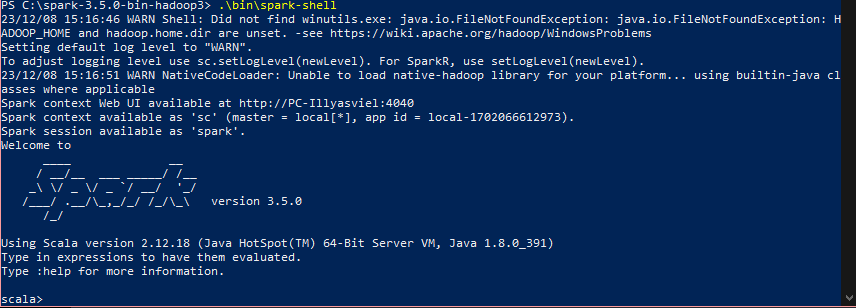
1. Install the spark and JDK.  
   A computer screen shot of a blue screen

   Description automatically generated
2. Start the spark shell  
   
3. Load and Cache the README.md File  
   A blue screen with white text

   Description automatically generated
4. Use the flatMap function to split each line into words.

For simplicity, we'll split words by spaces and remove punctuation. A more robust approach would use a regular expression to accurately handle word boundaries.:



1. Map each word to a key-value pair where the key is the word, and the value is 1 (indicating one occurrence of the word).  
   
2. Reduce by Key to Count Words:  
   Use the reduceByKey function to count the occurrences of each word.,   
   
3. Collect and Analyze the Results:

Collect the results to your driver program using the collect method.

This will give you an array of (word, count) pairs.  
A computer screen shot of white text

Description automatically generated

1. Then iterate over words to find:

Count of "Hadoop"

Most common word and its count

Least common word and its count

A computer screen shot of a blue screen

Description automatically generated

* **Occurrences of "Hadoop"**: The word "Hadoop" appears 5 times in the file.
* **Most Common Word**: The most common word appears to be **,** (a comma) with 41 occurrences. This indicates that punctuation was not fully removed in the word splitting process. To get a more meaningful result, you might consider enhancing the word-splitting logic to better handle punctuation.
* **Least Common Word**: The least common word is "integration," occurring just once.

1. It took 0.6 second.  
   A screenshot of a computer

   Description automatically generated

The code used during the lab:

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\MikeZihanMa> where spark-shell

PS C:\Users\MikeZihanMa> cd ..

PS C:\Users> cd ..

PS C:\> ls

Directory: C:\

Mode LastWriteTime Length Name

---- ------------- ------ ----

d----- 2021/9/20 15:00 Coq

d----- 2022/4/24 20:43 cs310

d----- 2021/9/28 3:16 data

d----- 2023/10/7 19:54 Intel

d----- 2023/12/2 17:54 java8

d----- 2023/12/5 3:12 New folder

d----- 2023/12/2 17:54 New folder (2)

d----- 2019/9/5 12:47 NVIDIA

d----- 2019/12/7 4:14 PerfLogs

d----- 2023/12/2 17:54 Program Files

d-r--- 2023/10/13 11:42 Program Files (x86)

d----- 2023/12/8 14:39 spark-3.5.0-bin-hadoop3

d----- 2020/3/11 22:26 symcache

d----- 2020/10/28 4:17 temp

d-r--- 2021/3/28 3:16 Users

d----- 2023/11/17 0:45 Windows

-a---- 2021/11/30 23:32 963 mysql\_install\_log.txt

PS C:\> cd .\spark-3.5.0-bin-hadoop3\

PS C:\spark-3.5.0-bin-hadoop3> ls

Directory: C:\spark-3.5.0-bin-hadoop3

Mode LastWriteTime Length Name

---- ------------- ------ ----

d----- 2023/12/8 14:38 bin

d----- 2023/12/8 14:38 conf

d----- 2023/12/8 14:37 data

d----- 2023/12/8 14:37 examples

d----- 2023/12/8 14:38 jars

d----- 2023/12/8 14:37 kubernetes

d----- 2023/12/8 14:37 licenses

d----- 2023/12/8 14:37 python

d----- 2023/12/8 14:37 R

d----- 2023/12/8 14:37 sbin

d----- 2023/12/8 14:37 yarn

-a---- 2023/9/8 22:08 22916 LICENSE

-a---- 2023/9/8 22:08 57842 NOTICE

-a---- 2023/9/8 22:08 4605 README.md

-a---- 2023/9/8 22:08 166 RELEASE

PS C:\spark-3.5.0-bin-hadoop3> java -version

java version "1.8.0\_391"

Java(TM) SE Runtime Environment (build 1.8.0\_391-b13)

Java HotSpot(TM) 64-Bit Server VM (build 25.391-b13, mixed mode)

PS C:\spark-3.5.0-bin-hadoop3> .\bin\spark-shell

23/12/08 15:16:46 WARN Shell: Did not find winutils.exe: java.io.FileNotFoundException: java.io.FileNotFoundException: HADOOP\_HOME and hadoop.home.dir are unset. -see https://wiki.apache.org/hadoop/WindowsProblems

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

23/12/08 15:16:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Spark context Web UI available at http://PC-Illyasviel:4040

Spark context available as 'sc' (master = local[\*], app id = local-1702066612973).

Spark session available as 'spark'.

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 3.5.0

/\_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0\_391)

Type in expressions to have them evaluated.

Type :help for more information.

scala> val readmeFile = sc.textFile("C:\spark-3.5.0-bin-hadoop3\README.md")

<console>:1: error: invalid escape character

val readmeFile = sc.textFile("C:\spark-3.5.0-bin-hadoop3\README.md")

^

<console>:1: error: invalid escape character

val readmeFile = sc.textFile("C:\spark-3.5.0-bin-hadoop3\README.md")

^

scala> val readmeFile = sc.textFile("C:/spark-3.5.0-bin-hadoop3/README.md")

readmeFile: org.apache.spark.rdd.RDD[String] = C:/spark-3.5.0-bin-hadoop3/README.md MapPartitionsRDD[1] at textFile at <console>:23

scala> readmeFile.cache()

res0: readmeFile.type = C:/spark-3.5.0-bin-hadoop3/README.md MapPartitionsRDD[1] at textFile at <console>:23

scala> val words = readmeFile.flatMap(line => line.split("\\s+"))

words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:23

scala> val wordPairs = words.map(word => (word.replaceAll("[,.!?:;]", "").toLowerCase, 1))

wordPairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:23

scala> val wordCounts = wordPairs.reduceByKey((a, b) => a + b)

wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

scala> val results = wordCounts.collect()

results: Array[(String, Int)] = Array((scala,3), (package,2), (this,3), (integration,1), (hive,2), (its,1), (guide,1), (guide](https//sparkapacheorg/contributinghtml),1), ([run,1), (spark](#building-spark),1), (general,2), (have,1), (pre-built,1), (locally,3), (changed,1), (enabling,1), (several,1), (only,1), ([[pypi,1), (basic,1), (first,1), (documentation,5), (testing,1), (graph,1), (info,1), ("yarn",1), ([project,1), (prefer,1), (engine,2), (version,2), (file,1), (analysis,1), (sql,2), (/dev/run-tests,1), (test,1), (hadoop-supported,1), (tips,1), (example,5), (are,1), (params,1), (scala>,1), (provides,1), (refer,2), (configure,1), (["specifying,1), (can,6), (build,3), (when,1), (easiest,1), (how,3), (thread,1), (large-scale,1), ([params]`,1), (interactive,2), (there,1), (maven](https...

scala> var hadoopCount = 0

hadoopCount: Int = 0

scala> var mostCommonWord = ""

mostCommonWord: String = ""

scala> var mostCommonCount = 0

mostCommonCount: Int = 0

scala> var leastCommonWord = ""

leastCommonWord: String = ""

scala> var leastCommonCount = Int.MaxValue

leastCommonCount: Int = 2147483647

scala>

scala> for ((word, count) <- results) {

| if (word == "hadoop") hadoopCount = count

| if (count > mostCommonCount) {

| mostCommonWord = word

| mostCommonCount = count

| }

| if (count < leastCommonCount && count > 0) {

| leastCommonWord = word

| leastCommonCount = count

| }

| }

scala>

scala> println(s"Hadoop Count: $hadoopCount")

Hadoop Count: 5

scala> println(s"Most Common Word: $mostCommonWord, Count: $mostCommonCount")

Most Common Word: , Count: 41

scala> println(s"Least Common Word: $leastCommonWord, Count: $leastCommonCount")

Least Common Word: integration, Count: 1

scala>

The GPT used for questioning: https://chat.openai.com/share/9deb9500-f9d5-456e-a99d-eccdf23f7163