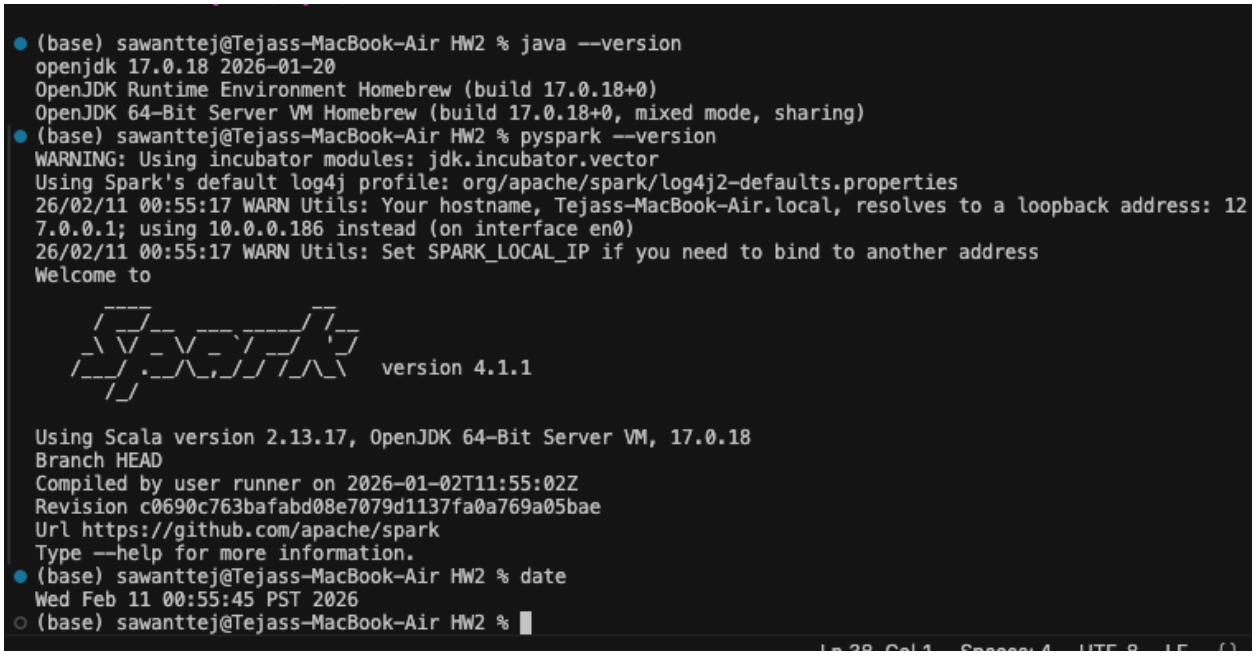
Name: Tejas Nandkishor Sawant StuID: 019101446

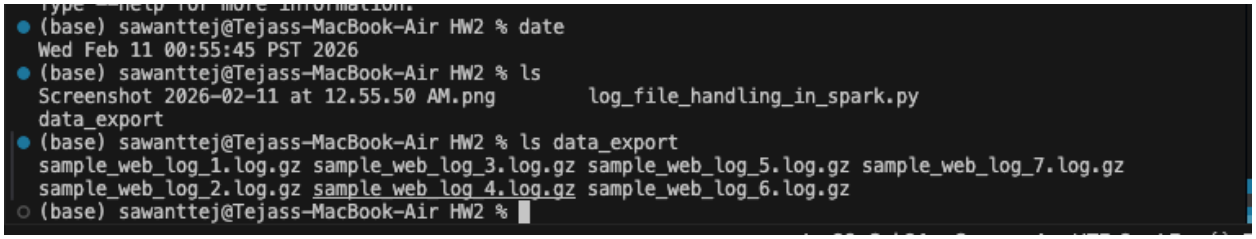
Data 228 Homework 2

1. Install Spark in your Laptop and print out “spark version” and “date” command.



This screenshot shows that Spark is successfully installed on my system. The Spark version and current system date are printed to verify the environment setup.

2. Download the 7 log files.



3. Configure Spark environment as shown in the demo code
    a. Create the input DataFrame (df) from 7 files and parsed DataFrame(log_df)

```
2
3   from pyspark.sql import SparkSession
4   import pyspark.sql.functions as F
5   from datetime import datetime
6   |   ⌘I for Command, ⌘L for Agent
7
8   spark = SparkSession.builder.appName("HandleLogFiles").getOrCreate()
9
10  # Printing Spark version and current time
11  print("Spark version: ", spark.version)
12  print("Current time: ", datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
13
14  # Load all .gz files in the directory into a DataFrame
15  df = spark.read.text("data_export/*.gz")
16
17  # Check the number of partitions
18  print("Number of partitions: ", df.rdd.getNumPartitions())
19  df.show(truncate=False)
20
```

In this step, I created a Spark session and loaded all 7 log files into a DataFrame. The output shows the number of partitions and a sample of the raw log data.

```
        at py4j.clientserver.ClientServerConnection.run(ClientServerConnection.java:108)
        at java.base/java.lang.Thread.run(Thread.java:840)
Number of partitions:  7
+-----------------------------------------------------------------------------------------+
|value                                                                                    |
+-----------------------------------------------------------------------------------------+
|123.45.67.89 - - [05/Nov/2024:02:08:16 +0000] "DELETE /cart HTTP/1.1" 500 242           |
|192.168.1.1 - - [04/Nov/2024:21:23:39 +0000] "POST /checkout HTTP/1.1" 404 2781         |
|234.56.78.90 - - [05/Nov/2024:07:06:19 +0000] "GET /api/data HTTP/1.1" 301 3758         |
|192.168.1.1 - - [04/Nov/2024:20:03:56 +0000] "POST /home HTTP/1.1" 200 1837             |
|192.168.1.1 - - [04/Nov/2024:21:25:05 +0000] "GET /products/123 HTTP/1.1" 200 3430      |
|234.56.78.90 - - [04/Nov/2024:07:38:10 +0000] "GET /api/data HTTP/1.1" 404 3729         |
|123.45.67.89 - - [04/Nov/2024:12:33:22 +0000] "PUT /api/data HTTP/1.1" 404 799          |
|192.168.1.1 - - [04/Nov/2024:07:37:46 +0000] "GET /api/data HTTP/1.1" 500 309           |
|123.45.67.89 - - [04/Nov/2024:21:52:36 +0000] "POST /checkout HTTP/1.1" 301 2375        |
|123.45.67.89 - - [04/Nov/2024:08:36:44 +0000] "DELETE /api/data HTTP/1.1" 404 3449      |
|192.168.1.1 - - [05/Nov/2024:03:15:43 +0000] "GET /api/data HTTP/1.1" 200 2319          |
|234.56.78.90 - - [05/Nov/2024:01:26:03 +0000] "DELETE /home HTTP/1.1" 500 1168          |
|234.56.78.90 - - [05/Nov/2024:03:26:33 +0000] "DELETE /cart HTTP/1.1" 500 1262          |
|123.45.67.89 - - [04/Nov/2024:20:46:25 +0000] "PUT /home HTTP/1.1" 301 4401             |
|123.45.67.89 - - [05/Nov/2024:08:07:51 +0000] "GET /api/data HTTP/1.1" 301 3736         |
|123.45.67.89 - - [04/Nov/2024:21:01:30 +0000] "DELETE /cart HTTP/1.1" 404 2418          |
|123.45.67.89 - - [04/Nov/2024:09:40:29 +0000] "POST /api/data HTTP/1.1" 301 3260        |
|234.56.78.90 - - [04/Nov/2024:09:23:42 +0000] "GET /home HTTP/1.1" 200 1488             |
|192.168.1.1 - - [04/Nov/2024:11:53:57 +0000] "POST /products/123 HTTP/1.1" 200 2627     |
|234.56.78.90 - - [05/Nov/2024:01:26:01 +0000] "PUT /cart HTTP/1.1" 500 4406             |
+-----------------------------------------------------------------------------------------+
only showing top 20 rows
```

```
# Create a parsed dataframe (log_df)
# Extract the necessary information from log data using regular expressions
pattern = r'(\d+\.\d+\.\d+\.\d+) - - \[(.*?)\] "(.*?) (.*?) HTTP.*" (\d+) (\d+)'
log_df = df.select(
    F.regexp_extract("value", pattern, 1).alias("ip"),
    F.regexp_extract("value", pattern, 2).alias("timestamp"),
    F.regexp_extract("value", pattern, 3).alias("method"),
    F.regexp_extract("value", pattern, 4).alias("url"),
    F.regexp_extract("value", pattern, 5).alias("status").cast("int"),
    F.regexp_extract("value", pattern, 6).alias("size").cast("int")
)
    ⌘I for Command, ⌘L for Agent
# Filtering out non-matching lines
log_df = log_df.filter(
    (F.col("status").isNotNull()) & F.col("size").isNotNull()
)

log_df.show(5, truncate=False)
```

Here, I parsed the raw log data using regular expression to extract relevant fields such as IP address, timestamp, method, URL, status, and size.

```
+------------+--------------------------+------+------------+------+----+
|ip          |timestamp                 |method|url         |status|size|
+------------+--------------------------+------+------------+------+----+
|123.45.67.89|05/Nov/2024:02:08:16 +0000|DELETE|/cart       |500   |242 |
|192.168.1.1 |04/Nov/2024:21:23:39 +0000|POST  |/checkout   |404   |2781|
|234.56.78.90|05/Nov/2024:07:06:19 +0000|GET   |/api/data   |301   |3758|
|192.168.1.1 |04/Nov/2024:20:03:56 +0000|POST  |/home       |200   |1837|
|192.168.1.1 |04/Nov/2024:21:25:05 +0000|GET   |/products/123|200   |3430|
+------------+--------------------------+------+------------+------+----+
only showing top 5 rows
```

4. Compute status and method combination counts using DataFrame operations.
    a. Count the occurrences of each unique (status, method) pair
    b. Sort the result in descending order by count
    c. Print the result

```
39
40
41  # DataFrame Operations
42  # Counting each unique (status, method) pair and sort desc by count
43  status_method_counts_df = (
44      log_df.groupBy("status", "method")
45      .count()
46      .orderBy(F.desc("count"))
47  )
48
49  # print the outcome
50  print("\n=== DataFrame result:(status, method) counts ===")
51  status_method_counts_df.show(truncate=False)
52
53
```

Using DataFrame operations, I grouped the data by status and method, counted the occurrences for each pair, and sorted them in decreasing order by count.

```
=== DataFrame result:(status, method) counts ===
+------+------+------+
|status|method|count |
+------+------+------+
|500   |GET   |438616|
|404   |GET   |438261|
|200   |GET   |438130|
|200   |PUT   |437899|
|404   |POST  |437857|
|301   |DELETE|437716|
|500   |POST  |437709|
|200   |POST  |437597|
|200   |DELETE|437495|
|500   |PUT   |437407|
|301   |PUT   |437309|
|301   |POST  |437250|
|301   |GET   |436962|
|404   |PUT   |436834|
|500   |DELETE|436516|
|404   |DELETE|436442|
+------+------+------+
```

5. Repeat step 4 using Spark SQL instead of DataFrame operations.

```
53
54  # Spark SQL
55  # Repeating the same operation as above using SQL
56
57  log_df.createOrReplaceTempView("logs")
58
59  status_method_counts_sql_df = spark.sql("""
60  SELECT
61      status,
62      method,
63      COUNT(1) as count
64  FROM
65      logs
66  GROUP BY
67      status,
68      method
69  ORDER BY
70      count DESC
71  """)
72
73  # print the outcome
74  print("\n=== Spark SQL result:(status, method) counts ===")
75  status_method_counts_sql_df.show(truncate=False)
76
77  spark.stop()
78
```

```
=== Spark SQL result:(status, method) counts ===
+------+------+------+
|status|method|count |
+------+------+------+
|500   |GET   |438616|
|404   |GET   |438261|
|200   |GET   |438130|
|200   |PUT   |437899|
|404   |POST  |437857|
|301   |DELETE|437716|
|500   |POST  |437709|
|200   |POST  |437597|
|200   |DELETE|437495|
|500   |PUT   |437407|
|301   |PUT   |437309|
|301   |POST  |437250|
|301   |GET   |436962|
|404   |PUT   |436834|
|500   |DELETE|436516|
|404   |DELETE|436442|
+------+------+------+
```

Name: Tejas Nandkishor Sawant StuID: 019101446

GitHub Link to the Code: log_file_handling_in_spark.py