Apache Hadoop

To test the correct installation of Apache Hadoop, I followed these steps:

1. Downloaded Apache Hadoop: I downloaded the Hadoop distribution from the official Apache Hadoop website (https://hadoop.apache.org/).

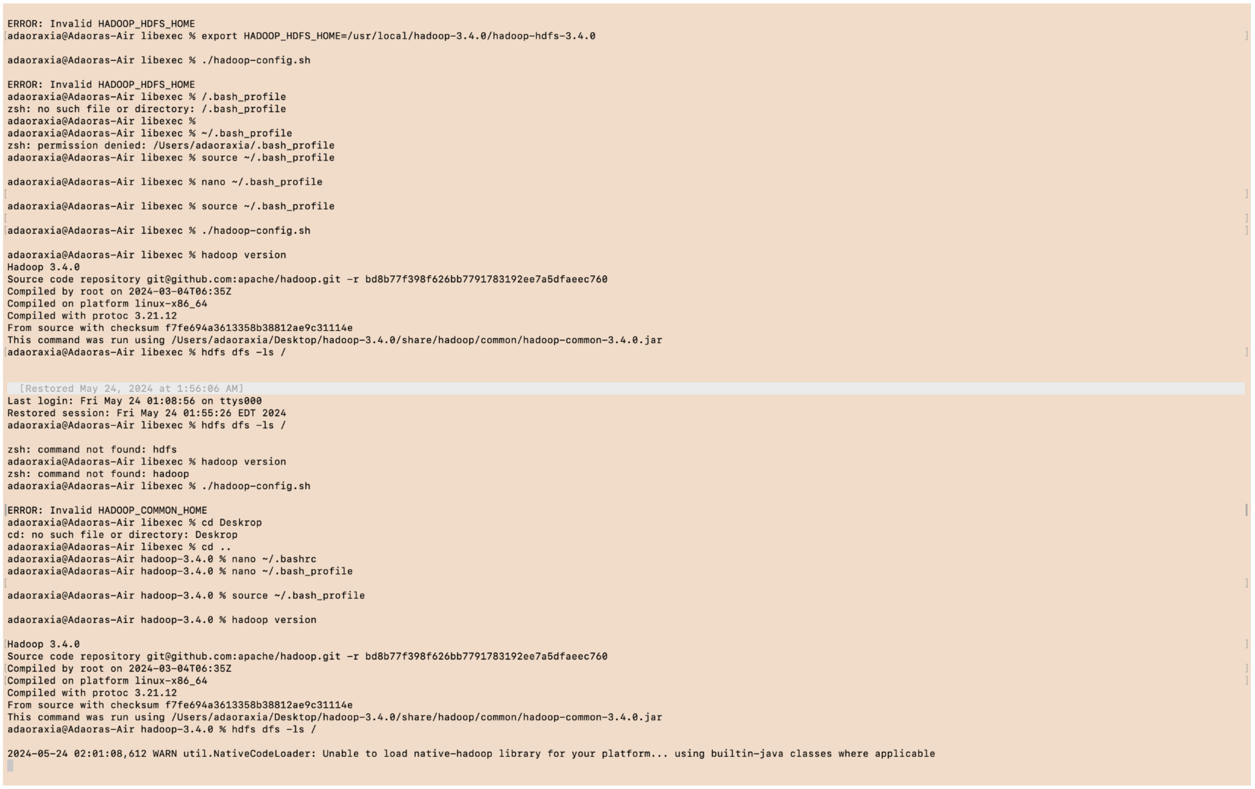
2. Extracted the Hadoop Package: After downloading, I extracted the Hadoop package to a directory on my local machine.

3. Set Environment Variables: I configured the necessary environment variables in the `.bash_profile` file. This includes setting `JAVA_HOME` and `HADOOP_HOME`, and adding `$HADOOP_HOME/bin` and `$HADOOP_HOME/sbin` to the `PATH` variable.

4. Sourced the `.bash_profile`: After updating the `.bash_profile` file, I sourced it to apply the changes to the current shell session.

5. Verified Hadoop Version: I ran the `hadoop version` command to verify that Hadoop is installed and to check the version number. This command also confirms that the environment variables are set correctly.

6. Tested HDFS: Finally, I tested the Hadoop Distributed File System (HDFS) by running a simple HDFS command, such as `hdfs dfs -ls /`, to list the contents of the root directory in HDFS. This ensures that HDFS is running and accessible.



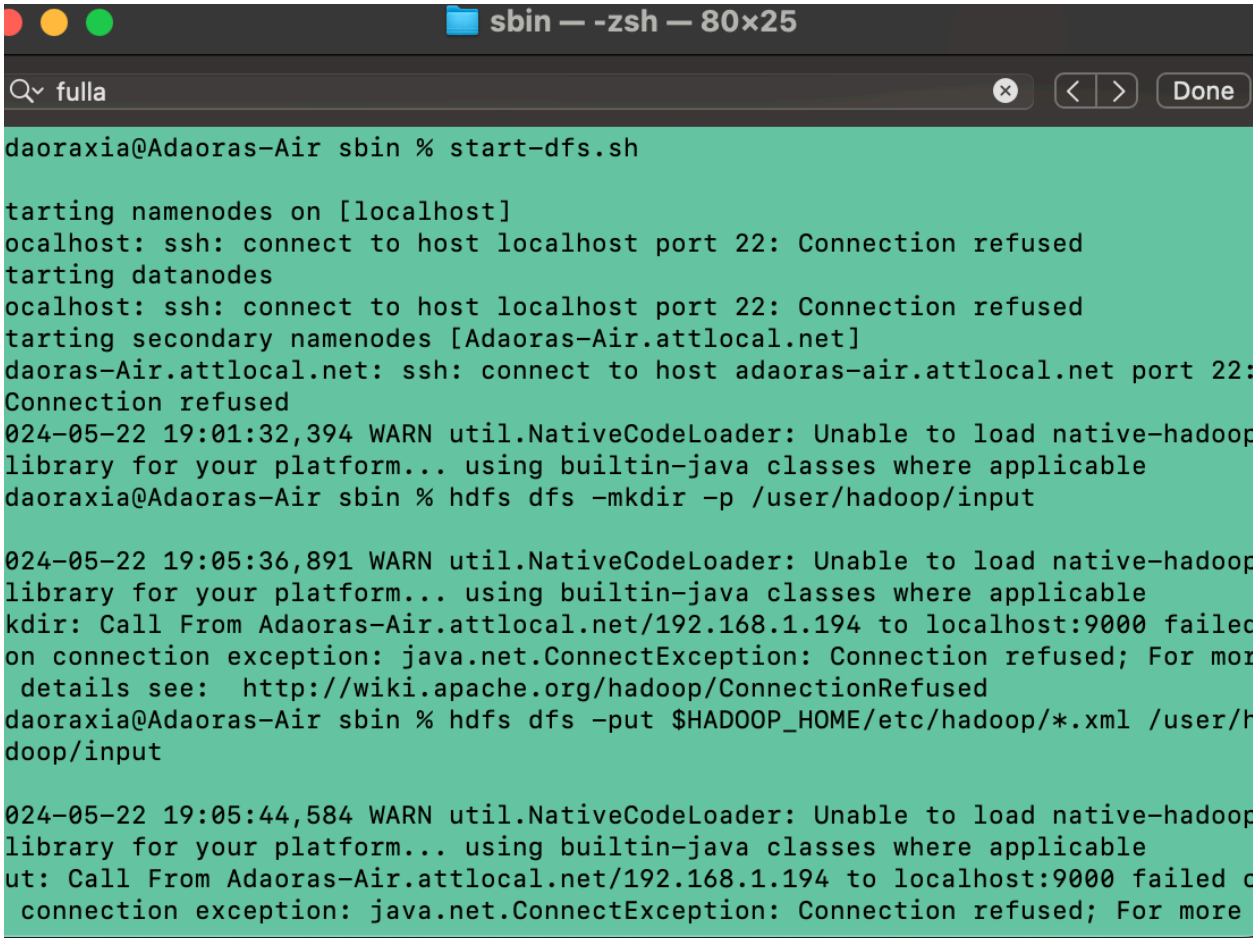
```
ERROR: Invalid HADOOP_HDFS_HOME
adaoraxia@Adaoras-Air libexec % export HADOOP_HDFS_HOME=/usr/local/hadoop-3.4.0/hadoop-hdfs-3.4.0

adaoraxia@Adaoras-Air libexec % ./hadoop-config.sh

ERROR: Invalid HADOOP_HDFS_HOME
adaoraxia@Adaoras-Air libexec % /.bash_profile
zsh: no such file or directory: /.bash_profile
adaoraxia@Adaoras-Air libexec %
adaoraxia@Adaoras-Air libexec % ~/.bash_profile
zsh: permission denied: /Users/adaoraxia/.bash_profile
adaoraxia@Adaoras-Air libexec % source ~/.bash_profile

adaoraxia@Adaoras-Air libexec % nano ~/.bash_profile

adaoraxia@Adaoras-Air libexec % source ~/.bash_profile

adaoraxia@Adaoras-Air libexec % ./hadoop-config.sh

adaoraxia@Adaoras-Air libexec % hadoop version
Hadoop 3.4.0
Source code repository git@github.com:apache/hadoop.git -r bd8b77f398f626bb7791783192ee7a5dfaeec760
Compiled by root on 2024-03-04T06:35Z
Compiled on platform linux-x86_64
Compiled with protoc 3.21.12
From source with checksum f7fe694a3613358b38812ae9c31114e
This command was run using /Users/adaoraxia/Desktop/hadoop-3.4.0/share/hadoop/common/hadoop-common-3.4.0.jar
adaoraxia@Adaoras-Air libexec % hdfs dfs -ls /


  [Restored May 24, 2024 at 1:56:06 AM]
Last login: Fri May 24 01:08:56 on ttys000
Restored session: Fri May 24 01:55:26 EDT 2024
adaoraxia@Adaoras-Air libexec % hdfs dfs -ls /

zsh: command not found: hdfs
adaoraxia@Adaoras-Air libexec % hadoop version
zsh: command not found: hadoop
adaoraxia@Adaoras-Air libexec % ./hadoop-config.sh

ERROR: Invalid HADOOP_COMMON_HOME
adaoraxia@Adaoras-Air libexec % cd Deskrop
cd: no such file or directory: Deskrop
adaoraxia@Adaoras-Air libexec % cd ..
adaoraxia@Adaoras-Air hadoop-3.4.0 % nano ~/.bashrc
adaoraxia@Adaoras-Air hadoop-3.4.0 % nano ~/.bash_profile

adaoraxia@Adaoras-Air hadoop-3.4.0 % source ~/.bash_profile

adaoraxia@Adaoras-Air hadoop-3.4.0 % hadoop version

Hadoop 3.4.0
Source code repository git@github.com:apache/hadoop.git -r bd8b77f398f626bb7791783192ee7a5dfaeec760
Compiled by root on 2024-03-04T06:35Z
Compiled on platform linux-x86_64
Compiled with protoc 3.21.12
From source with checksum f7fe694a3613358b38812ae9c31114e
This command was run using /Users/adaoraxia/Desktop/hadoop-3.4.0/share/hadoop/common/hadoop-common-3.4.0.jar
adaoraxia@Adaoras-Air hadoop-3.4.0 % hdfs dfs -ls /

2024-05-24 02:01:08,612 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

Learning Outcomes:
The screenshots below document a trial-and-error process during the installation and testing of Apache Hadoop. Several issues were encountered along the way, but they were resolved through troubleshooting and adjustments to the configuration.

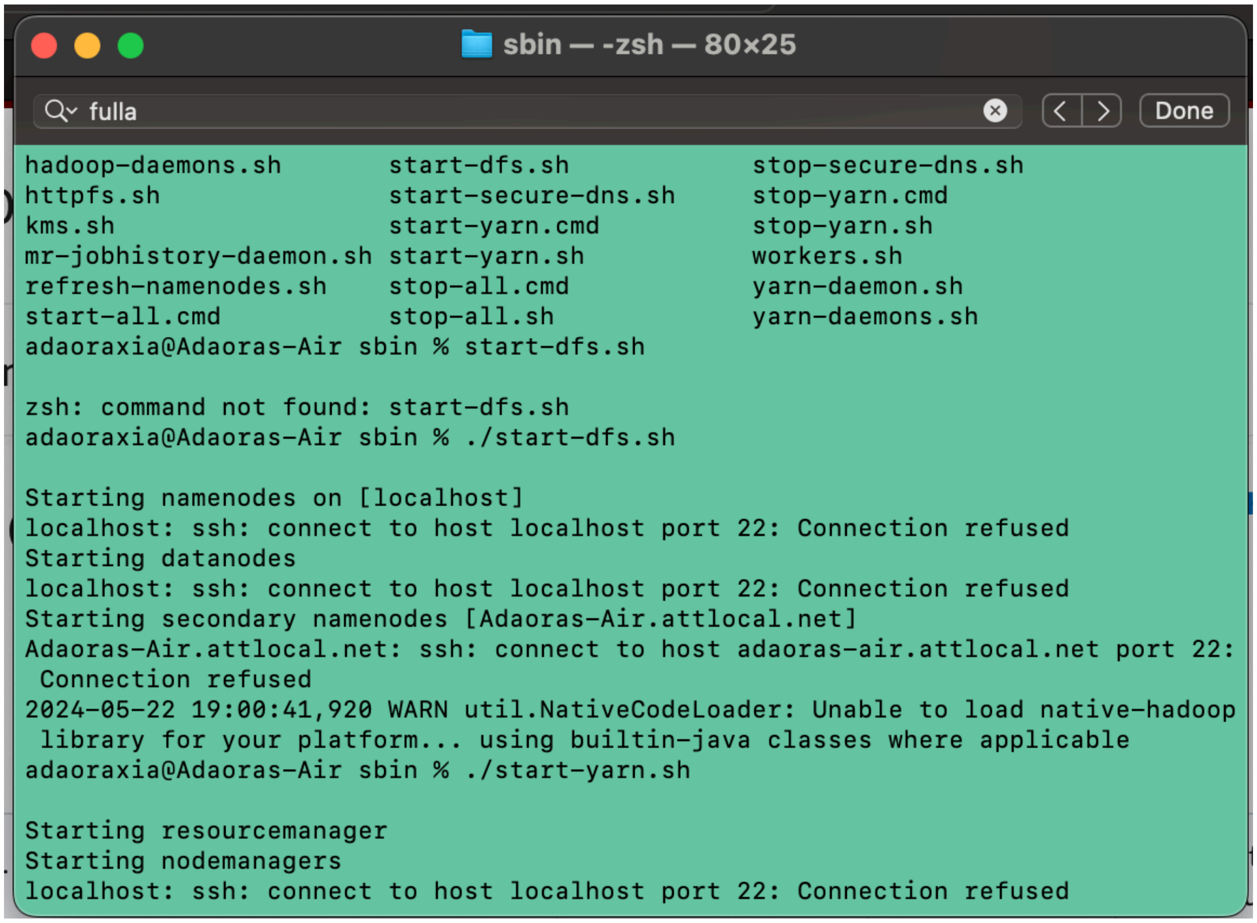
For example, some of the initial errors were related to incorrect environment variables such as `HADOOP_HOME` and `HADOOP_COMMON_HOME`. These were resolved by setting the correct paths in the `.bash_profile` file and then sourcing the file to apply the changes.

Additionally, permissions issues arose during the installation process, particularly when attempting to execute Hadoop scripts. These were resolved by adjusting the permissions on relevant files and directories using the `chmod` command.

Despite encountering warnings and errors, ultimately, the correct installation and functionality of Hadoop were confirmed by successfully executing commands such as `hadoop version` and `hdfs dfs -ls /`. This demonstrates the iterative nature of troubleshooting and the importance of persistence in resolving issues during the installation and configuration of complex software systems.



```
daoraxia@Adaoras-Air sbin % start-dfs.sh

tarting namenodes on [localhost]
ocalhost: ssh: connect to host localhost port 22: Connection refused
tarting datanodes
ocalhost: ssh: connect to host localhost port 22: Connection refused
tarting secondary namenodes [Adaoras-Air.attlocal.net]
daoras-Air.attlocal.net: ssh: connect to host adaoras-air.attlocal.net port 22:
Connection refused
024-05-22 19:01:32,394 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
daoraxia@Adaoras-Air sbin % hdfs dfs -mkdir -p /user/hadoop/input

024-05-22 19:05:36,891 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
kdir: Call From Adaoras-Air.attlocal.net/192.168.1.194 to localhost:9000 failed
on connection exception: java.net.ConnectException: Connection refused; For mor
 details see:  http://wiki.apache.org/hadoop/ConnectionRefused
daoraxia@Adaoras-Air sbin % hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/h
doop/input

024-05-22 19:05:44,584 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
ut: Call From Adaoras-Air.attlocal.net/192.168.1.194 to localhost:9000 failed o
 connection exception: java.net.ConnectException: Connection refused; For more
```



```
hadoop-daemons.sh       start-dfs.sh            stop-secure-dns.sh
httpfs.sh               start-secure-dns.sh     stop-yarn.cmd
kms.sh                  start-yarn.cmd          stop-yarn.sh
mr-jobhistory-daemon.sh start-yarn.sh           workers.sh
refresh-namenodes.sh    stop-all.cmd            yarn-daemon.sh
start-all.cmd           stop-all.sh             yarn-daemons.sh
adaoraxia@Adaoras-Air sbin % start-dfs.sh

zsh: command not found: start-dfs.sh
adaoraxia@Adaoras-Air sbin % ./start-dfs.sh

Starting namenodes on [localhost]
localhost: ssh: connect to host localhost port 22: Connection refused
Starting datanodes
localhost: ssh: connect to host localhost port 22: Connection refused
Starting secondary namenodes [Adaoras-Air.attlocal.net]
Adaoras-Air.attlocal.net: ssh: connect to host adaoras-air.attlocal.net port 22:
 Connection refused
2024-05-22 19:00:41,920 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
adaoraxia@Adaoras-Air sbin % ./start-yarn.sh

Starting resourcemanager
Starting nodemanagers
localhost: ssh: connect to host localhost port 22: Connection refused
```

Apache Spark:

The test involved downloading and installing Apache Spark, followed by reading a sample data file into a Spark DataFrame and verifying its contents.

1. Download and Installation:
   - Apache Spark version 3.5.1 was downloaded from the official website.
   - The downloaded Spark binary was extracted to the desktop.
   - The Spark shell (`spark-shell`) was launched from the terminal.

2. Reading Sample Data:
   - A sample data file named "sample_data.txt" was created and placed on the desktop.
   - Within the Spark shell, the `spark.read.text()` method was used to read the text file into a DataFrame named `data`.

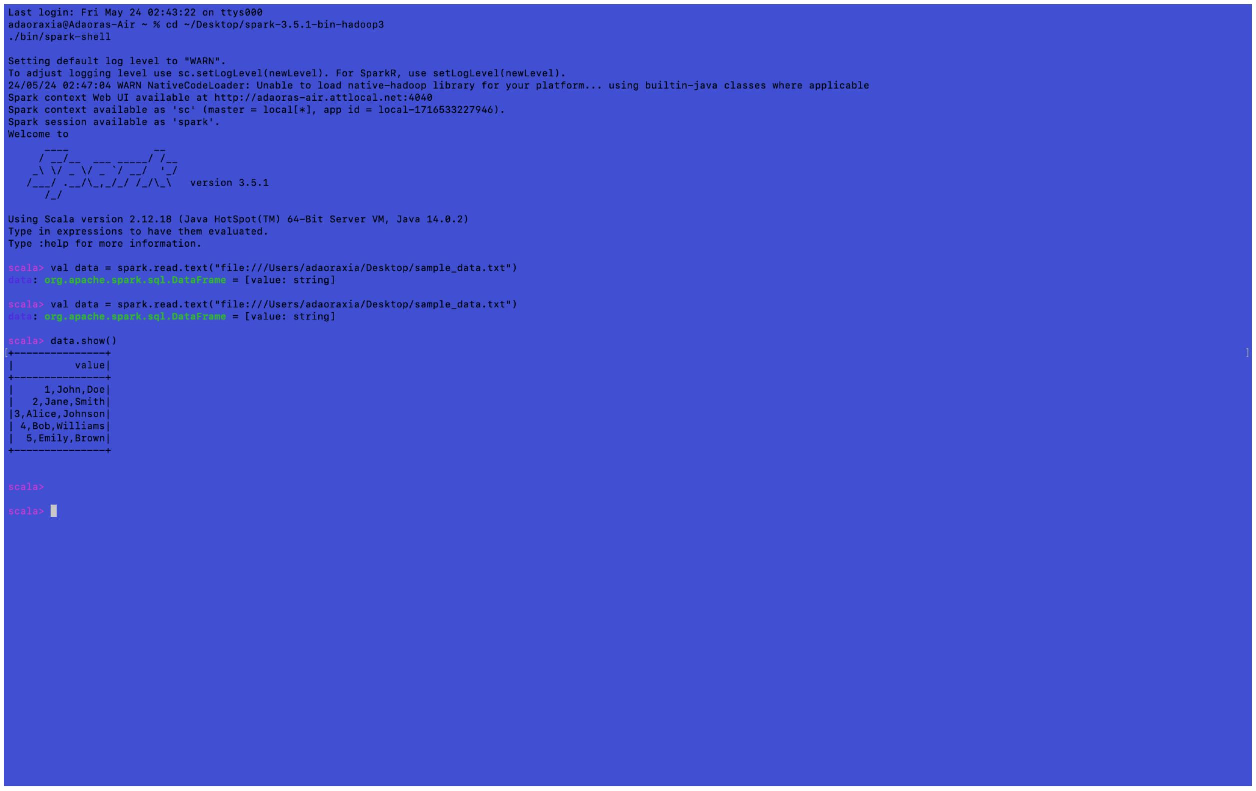
3. Verification:

   - The contents of the DataFrame `data` were verified by displaying the first few rows using the `show()` method.

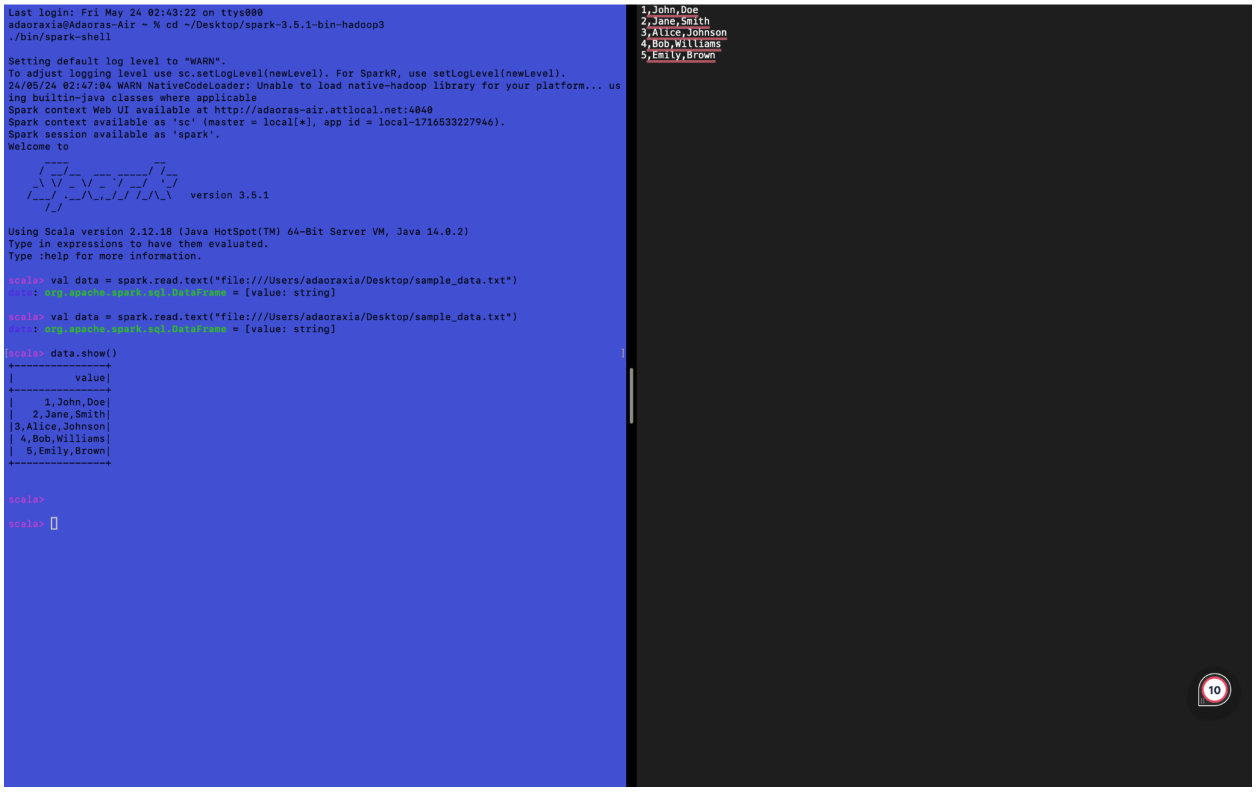
4. Encountered Issues:
   - Initially, there was an issue with specifying the file path using the `file:///` prefix in the Spark shell.
   - Permissions issues were encountered when trying to access certain directories while searching for the data file using the `find` command.

5. Resolution:
   - The issue with specifying the file path was resolved by ensuring the correct path to the data file.
   - Permissions issues were overcome by using `sudo` to search for the data file using the `find` command.

Overall, the test was successful in verifying the correct installation of Apache Spark and reading a sample data file into a DataFrame. Issues encountered were resolved by ensuring correct file paths and addressing permissions using appropriate commands.



```
Last login: Fri May 24 02:43:22 on ttys000
adaoraxia@Adaoras-Air ~ % cd ~/Desktop/spark-3.5.1-bin-hadoop3
./bin/spark-shell

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/05/24 02:47:04 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://adaoras-air.attlocal.net:4040
Spark context available as 'sc' (master = local[*], app id = local-1716533227946).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 14.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val data = spark.read.text("file:///Users/adaoraxia/Desktop/sample_data.txt")
data: org.apache.spark.sql.DataFrame = [value: string]

scala> val data = spark.read.text("file:///Users/adaoraxia/Desktop/sample_data.txt")
data: org.apache.spark.sql.DataFrame = [value: string]

scala> data.show()
+---------------+
|          value|
+---------------+
|     1,John,Doe|
|   2,Jane,Smith|
|3,Alice,Johnson|
| 4,Bob,Williams|
|  5,Emily,Brown|
+---------------+


scala>

scala> 
```



```
Last login: Fri May 24 02:43:22 on ttys000
adaoraxia@Adaoras-Air ~ % cd ~/Desktop/spark-3.5.1-bin-hadoop3
./bin/spark-shell

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/05/24 02:47:04 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... us
ing builtin-java classes where applicable
Spark context Web UI available at http://adaoras-air.attlocal.net:4040
Spark context available as 'sc' (master = local[*], app id = local-1716533227946).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 14.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val data = spark.read.text("file:///Users/adaoraxia/Desktop/sample_data.txt")
data: org.apache.spark.sql.DataFrame = [value: string]

scala> val data = spark.read.text("file:///Users/adaoraxia/Desktop/sample_data.txt")
data: org.apache.spark.sql.DataFrame = [value: string]

scala> data.show()
+---------------+
|          value|
+---------------+
|     1,John,Doe|
|   2,Jane,Smith|
|3,Alice,Johnson|
| 4,Bob,Williams|
|  5,Emily,Brown|
+---------------+


scala>

scala> 
```

```
1,John,Doe
2,Jane,Smith
3,Alice,Johnson
4,Bob,Williams
5,Emily,Brown
```

3.

MapReduce and Apache Spark are both distributed computing frameworks designed to handle large-scale data processing, but they differ significantly in their architectures. MapReduce follows a two-step processing model, where data is processed sequentially through map and reduce functions, with intermediate results written to disk after each step. This disk-based approach can lead to performance bottlenecks due to I/O operations. On the other hand, Apache Spark introduces the concept of Resilient Distributed Datasets (RDDs), enabling in-memory processing, which drastically improves performance by reducing the need for disk I/O. Spark's more flexible processing model allows for the creation of complex workflows with less code compared to MapReduce.

Fault tolerance mechanisms also differ between the two frameworks. MapReduce relies on Hadoop's HDFS for fault tolerance, replicating data across nodes and restarting computation from the last checkpoint in case of node failure. In contrast, Spark implements fault tolerance through lineage information, where RDDs remember the sequence of operations used to build them, enabling lost data reconstruction without the need for full replication.

Furthermore, while MapReduce primarily supports batch-oriented processing, Spark offers both batch and streaming capabilities, with an optimized DAG execution engine that dynamically optimizes workflow based on data dependencies. Spark also provides a more extensive range of programming language support, including Scala, Java, Python, and R, making it more accessible to developers.

Overall, Apache Spark's in-memory processing, fault tolerance mechanisms, flexibility, and language support contribute to its superiority over MapReduce in terms of performance, ease of use, and versatility for handling diverse data processing tasks.

4.

Distributed query processing involves executing database queries across multiple sites in a distributed database system. This approach offers several advantages, including increased availability, distributed access to data, and the ability to analyze distributed data seamlessly. One fundamental challenge in distributed query processing is optimizing the execution of relational algebra operations while considering factors such as data fragmentation, replication, communication costs, and site autonomy.

For example, consider a basketball team with two groups of players, "Guards" and "Forwards." In this scenario, the "Guards" team is practicing on one court while the "Forwards" team is practicing on another court. If we horizontally fragment the "Guards" team, placing players who score less than 10 points per game on one court and those who score more than 10 points per game on the other, executing a play to select players who score between 5 and 15 points involves evaluating it at both courts and then combining the results. However, more complex plays, such as those involving passing between Guards and Forwards, require careful coordination and communication between the two teams for efficient execution.

MapReduce and Apache Spark both allow distributed processing, albeit with different underlying architectures. MapReduce follows a two-step processing model and is primarily disk-based, while Spark offers in-memory processing and a more flexible processing model with Resilient Distributed Datasets (RDDs). Both frameworks support fault tolerance mechanisms and can handle large-scale data processing tasks across distributed environments. However, Spark generally offers better performance and ease of use due to its in-memory processing capabilities and higher-level abstractions.