Big Data Analytics

Journal

Submitted By

Tagadghar Shailesh Tagadghar

Roll No: 31031523034

MSc CS – Part II

Department Of Computer Science

Somaiya Vidyavihar University

SK Somaiya college

Name : Shailesh A. Tagadghar

Roll No : 31031423034

Index :

Name : Shailesh A. Tagadghar

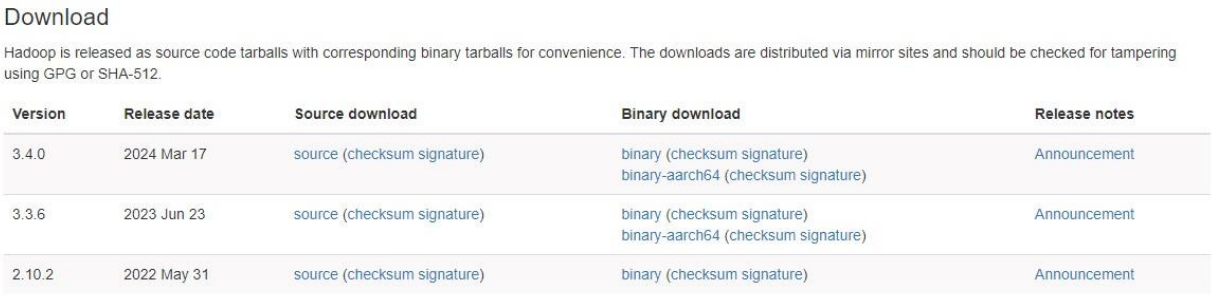
Roll No : 31031423034

Practical 1

Aim : Installation of Hadoop in Windows

Steps :

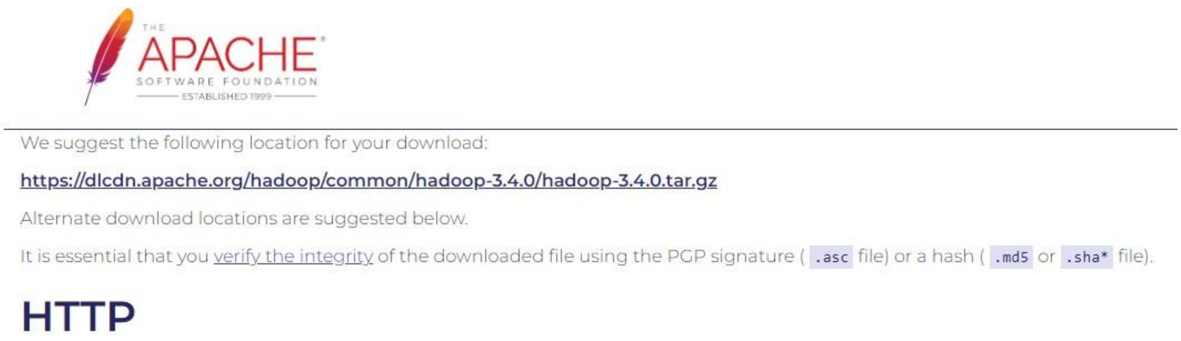
1. Download Binary File for Windows. https://hadoop.apache.org/releases.html



Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.4.0	2024 Mar 17	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement
3.3.6	2023 Jun 23	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement
2.10.2	2022 May 31	source (checksum signature)	binary (checksum signature)	Announcement



We suggest the following location for your download:

https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz

Alternate download locations are suggested below.

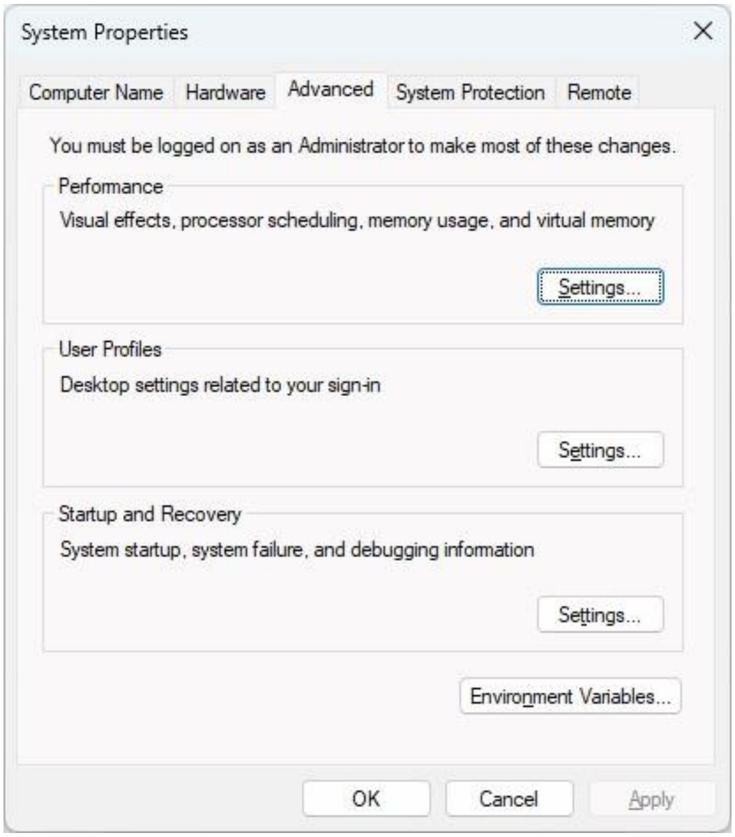
It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

HTTP

https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz

https://www.oracle.com/java/technologies/javase-downloads.html

2. Extract the file using Winrar.
3. Go to “Edit Environment Variables” and Click Environment Variables.



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

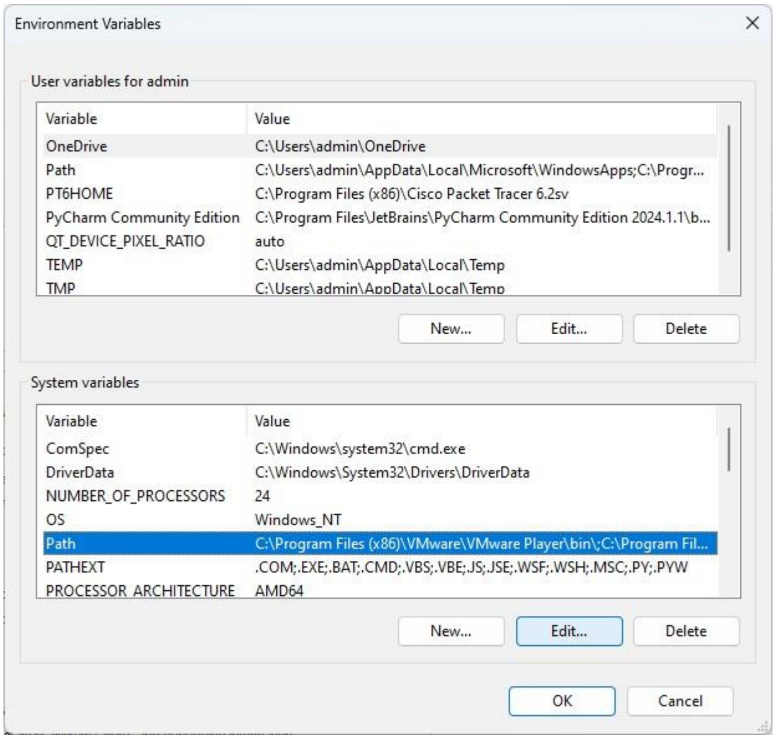
Startup and Recovery

System startup, system failure, and debugging information

4. Under System Variables click “New” and set “Variable name” as JAVA_HOME and “Variable value” as the path of your JAVA JDK.

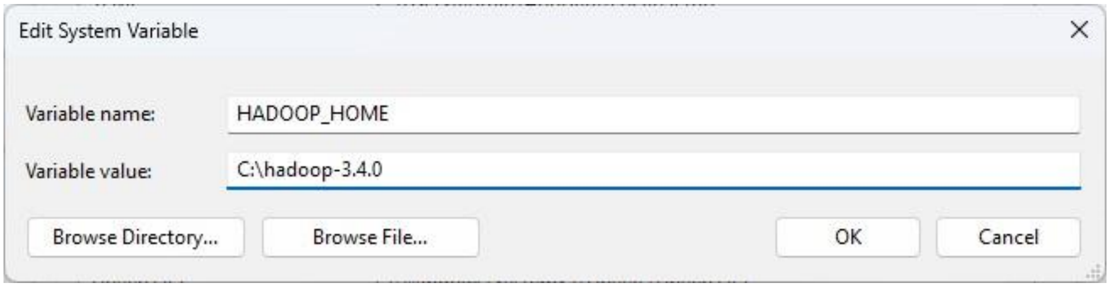
   C:\Program Files\Java\jdk-21
5. Similarly add “HADOOP_HOME” variable.
6. Download the bin folder from the below link https://drive.google.com/drive/folders/1iURNbow2IglhAhSy3sfY5xxVfAg33NBW
7. Extract the bin archive and replace the bin folder in Hadoop folder with the bin folder in this archive.
8. Check if “winutils” is working. If you get any dll error then download that dll and paste in the Windows -> System32 folder.
9. Move the Hadoop folder to C drive.
10. Create a data folder in the hadoop home directory and add the folders datanode and namenode to it.
11. Add the following path to “Path” under “System Variables” in “Edit Environment Variables”

C:\hadoop-3.4.0\sbin



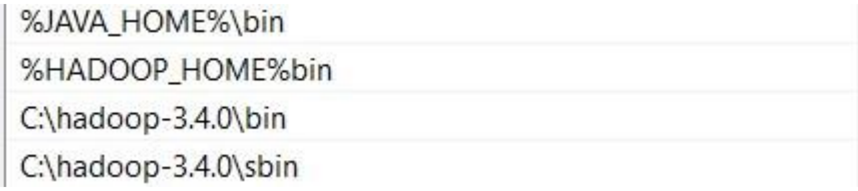
Variable	Value
OneDrive	C:\Users\admin\OneDrive
Path	C:\Users\admin\AppData\Local\Microsoft\WindowsApps;C:\Progr...
PT6HOME	C:\Program Files (x86)\Cisco Packet Tracer 6.2sv
PyCharm Community Edition	C:\Program Files\JetBrains\PyCharm Community Edition 2024.1.1\b...
QT_DEVICE_PIXEL_RATIO	auto
TEMP	C:\Users\admin\AppData\Local\Temp
TMP	C:\Users\admin\AppData\Local\Temp

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	24
OS	Windows_NT
Path	C:\Program Files (x86)\VMware\VMware Player\bin\;C:\Program Fil...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64



Variable name: HADOOP_HOME

Variable value: C:\hadoop-3.4.0



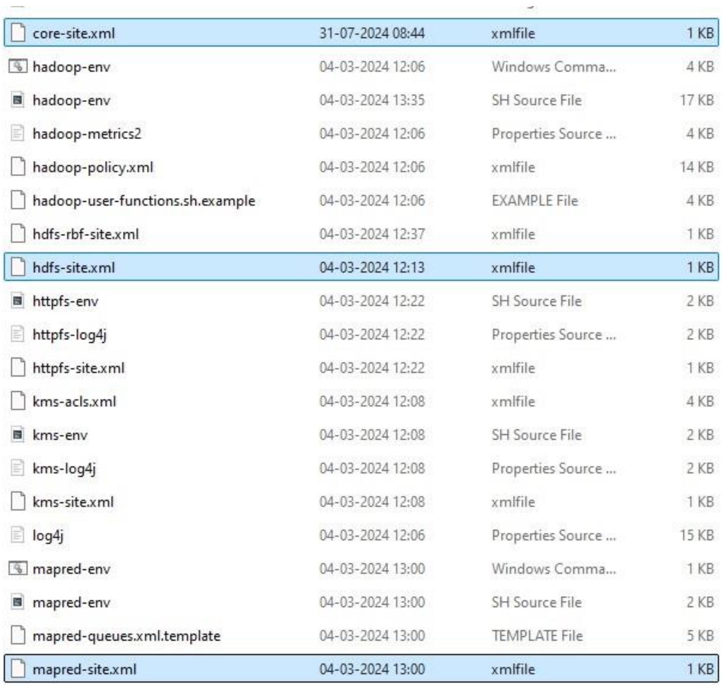
```
%JAVA_HOME%\bin
%HADOOP_HOME%bin
C:\hadoop-3.4.0\bin
C:\hadoop-3.4.0\sbin
```

12.If your PC username has spaces in it then go to hadoop-env.cmd and find this line

set HADOOP_IDENT_STRING=%USERNAME%

13. Change the above line to your PC username instead of %USERNAME% but WITHOUT SPACES set HADOOP_IDENT_STRING=DeepShah

14. Make the changes to the following files as given, in “etc/hadoop” folder of hadoop home.



core-site.xml	31-07-2024 08:44	xmlfile	1 KB
hadoop-env	04-03-2024 12:06	Windows Comma...	4 KB
hadoop-env	04-03-2024 13:35	SH Source File	17 KB
hadoop-metrics2	04-03-2024 12:06	Properties Source ...	4 KB
hadoop-policy.xml	04-03-2024 12:06	xmlfile	14 KB
hadoop-user-functions.sh.example	04-03-2024 12:06	EXAMPLE File	4 KB
hdfs-rbf-site.xml	04-03-2024 12:37	xmlfile	1 KB
hdfs-site.xml	04-03-2024 12:13	xmlfile	1 KB
httpfs-env	04-03-2024 12:22	SH Source File	2 KB
httpfs-log4j	04-03-2024 12:22	Properties Source ...	2 KB
httpfs-site.xml	04-03-2024 12:22	xmlfile	1 KB
kms-acls.xml	04-03-2024 12:08	xmlfile	4 KB
kms-env	04-03-2024 12:08	SH Source File	2 KB
kms-log4j	04-03-2024 12:08	Properties Source ...	2 KB
kms-site.xml	04-03-2024 12:08	xmlfile	1 KB
log4j	04-03-2024 12:06	Properties Source ...	15 KB
mapred-env	04-03-2024 13:00	Windows Comma...	1 KB
mapred-env	04-03-2024 13:00	SH Source File	2 KB
mapred-queues.xml.template	04-03-2024 13:00	TEMPLATE File	5 KB
mapred-site.xml	04-03-2024 13:00	xmlfile	1 KB

core-site.xml

```
<configuration>

        <property>

                <name>fs.default.name</name>

                 <value>hdfs://localhost:9000</value>

        </property>
</configuration>
```

mapred-site.xml

```
<configuration>

        <property>

            <name>mapred.framework.name</name>
                <value>yarn</value>
        </property>
</configuration> hdfs-
site.xml <configuration>

        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>

        <property>
                <name>dfs.namenode.name.dir</name>
                <value>C:\hadoop-3.4.0\data\namenode</value>
        </property>

        <property>
                <name>dfs.datanode.data.dir</name>
                <value>C:\hadoop-3.4.0\data\datanode</value>
        </property>
</configuration>
```

yarn-site.xml

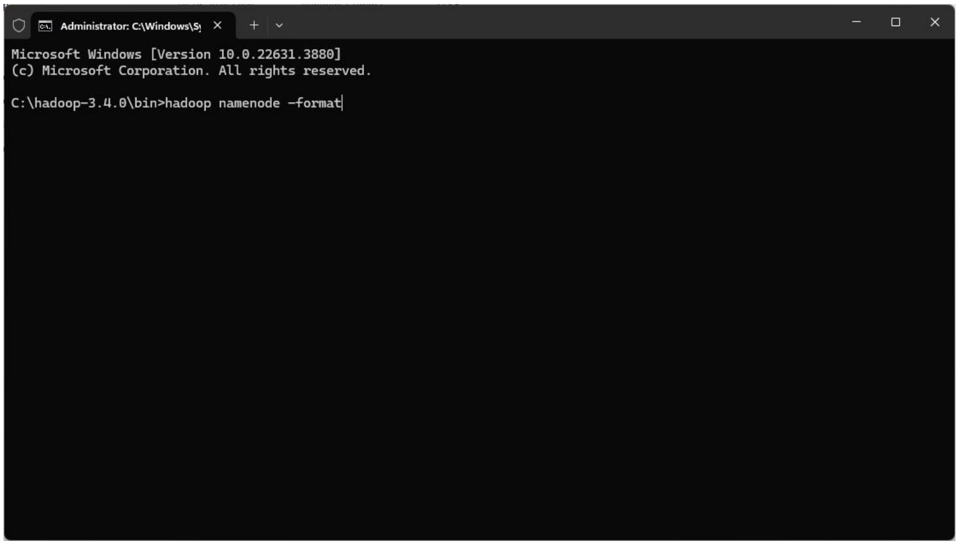
```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.auxservice.mapreduce.shuffle.class</name>
```

<value>org.apache.hadoop.mapred.shuffleHandler</value> </property> </configuration>

15. Go to hadoop-env.cmd file in /etc/hadoop folder and replace the set JAVA_HOME=%JAVA_HOME% line with the following:

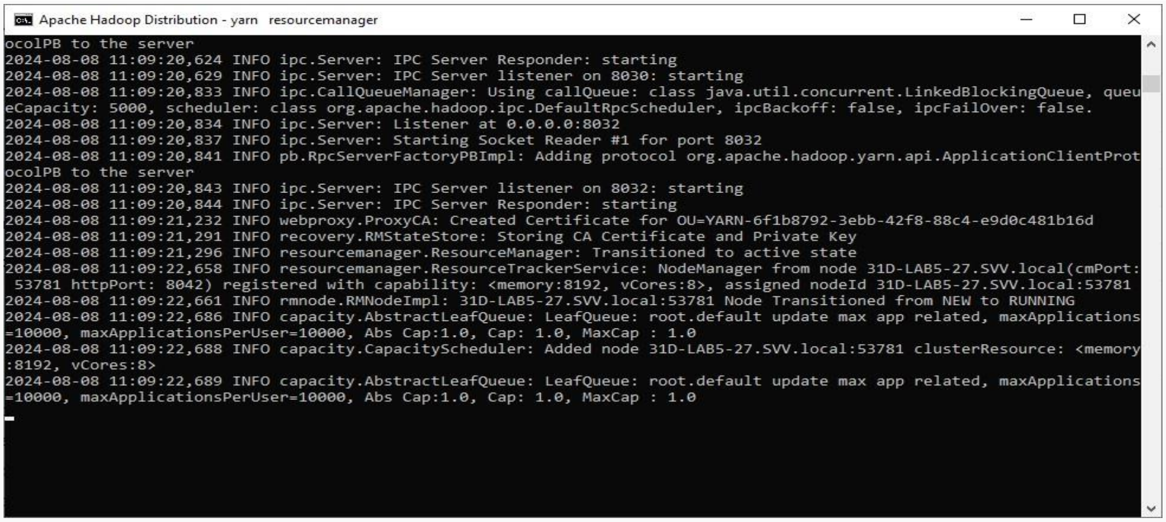
set JAVA_HOME=C:\Progra~1\Java\jdk-21

16. Restart your PC for the changes to take effect.
17. Go to Admin Command prompt and type “hadoop” to see if the server is recognized.
18. Type “hdfs namenode -format” to format the namenode.

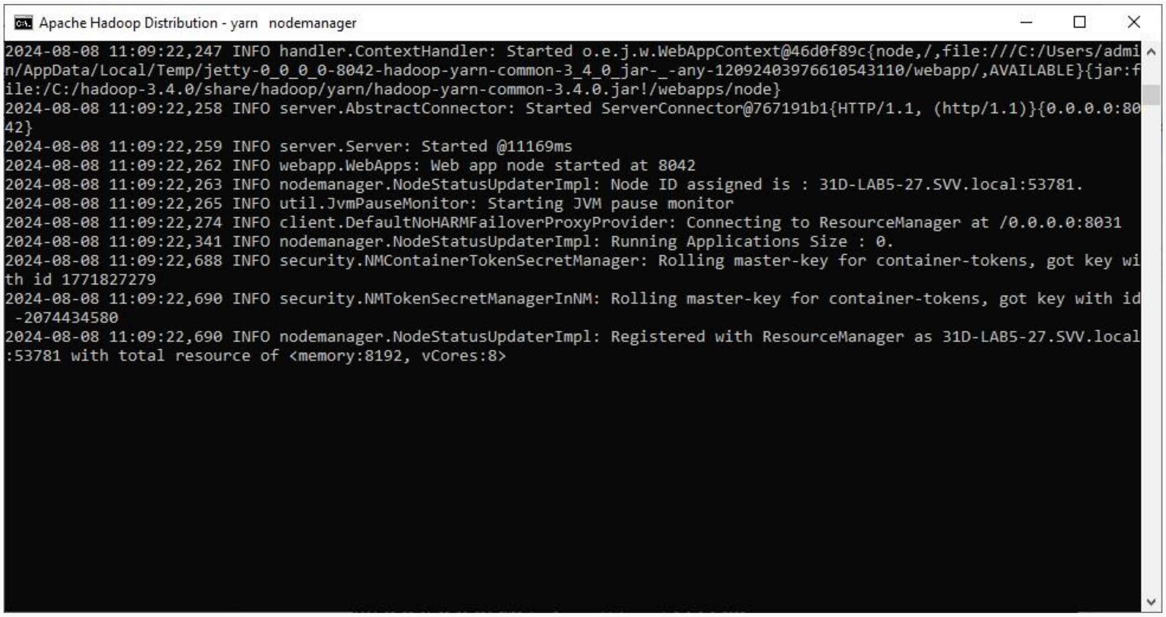


```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\hadoop-3.4.0\bin>hadoop namenode -format
```

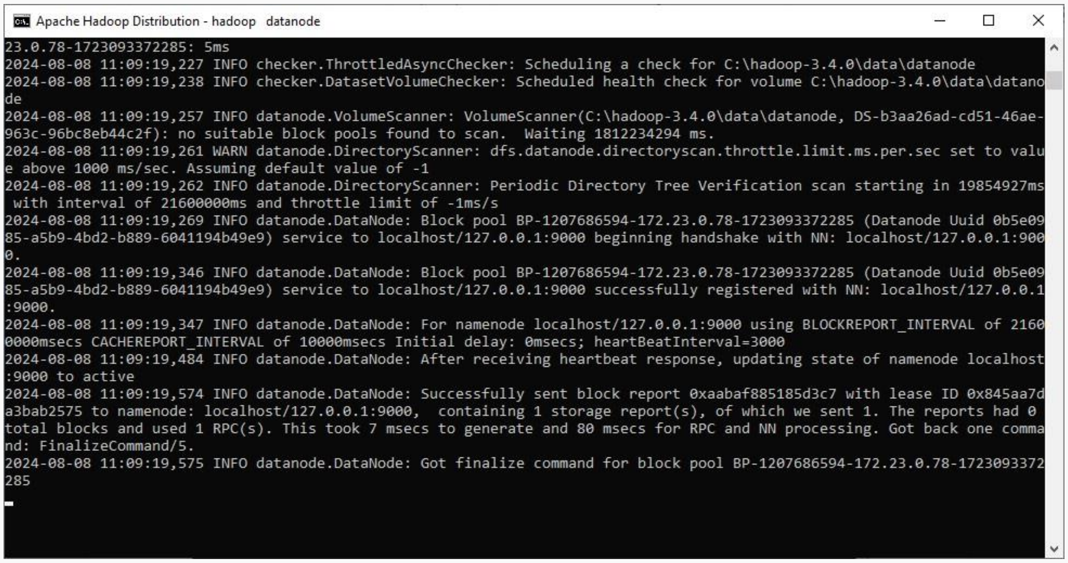
19. Type start-dfs.cmd and start-yarn.cmd to start all hadoop processes



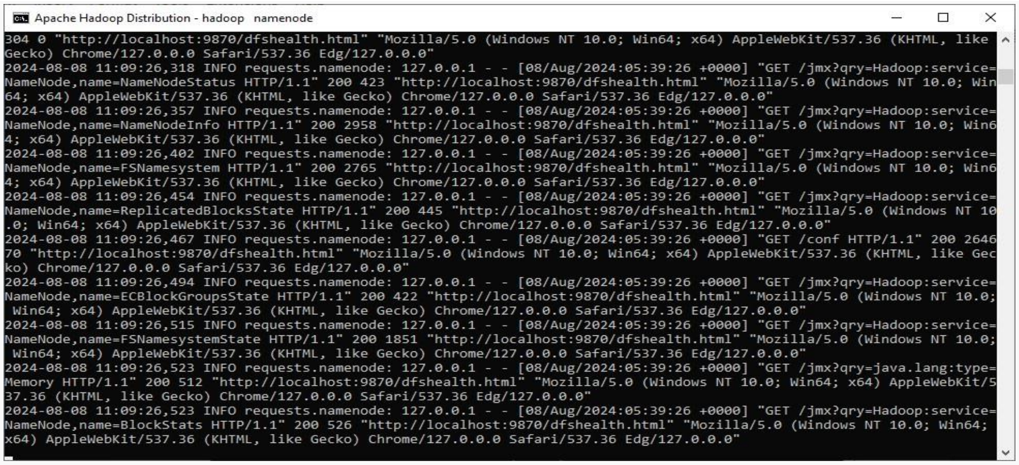
```
ocolPB to the server
2024-08-08 11:09:20,624 INFO ipc.Server: IPC Server Responder: starting
2024-08-08 11:09:20,629 INFO ipc.Server: IPC Server listener on 8030: starting
2024-08-08 11:09:20,833 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queu
eCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false, ipcFailOver: false.
2024-08-08 11:09:20,834 INFO ipc.Server: Listener at 0.0.0.0:8032
2024-08-08 11:09:20,837 INFO ipc.Server: Starting Socket Reader #1 for port 8032
2024-08-08 11:09:20,841 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.ApplicationClientProt
ocolPB to the server
2024-08-08 11:09:20,843 INFO ipc.Server: IPC Server listener on 8032: starting
2024-08-08 11:09:20,844 INFO ipc.Server: IPC Server Responder: starting
2024-08-08 11:09:21,232 INFO webproxy.ProxyCA: Created Certificate for OU=YARN-6f1b8792-3ebb-42f8-88c4-e9d0c481b16d
2024-08-08 11:09:21,291 INFO recovery.RMStateStore: Storing CA Certificate and Private Key
2024-08-08 11:09:21,296 INFO resourcemanager.ResourceManager: Transitioned to active state
2024-08-08 11:09:22,658 INFO resourcemanager.ResourceTrackerService: NodeManager from node 31D-LAB5-27.SVV.local(cmPort:
 53781 httpPort: 8042) registered with capability: <memory:8192, vCores:8>, assigned nodeId 31D-LAB5-27.SVV.local:53781
2024-08-08 11:09:22,661 INFO rmnode.RMNodeImpl: 31D-LAB5-27.SVV.local:53781 Node Transitioned from NEW to RUNNING
2024-08-08 11:09:22,686 INFO capacity.AbstractLeafQueue: LeafQueue: root.default update max app related, maxApplications
=10000, maxApplicationsPerUser=10000, Abs Cap:1.0, Cap: 1.0, MaxCap : 1.0
2024-08-08 11:09:22,688 INFO capacity.CapacityScheduler: Added node 31D-LAB5-27.SVV.local:53781 clusterResource: <memory
:8192, vCores:8>
2024-08-08 11:09:22,689 INFO capacity.AbstractLeafQueue: LeafQueue: root.default update max app related, maxApplications
=10000, maxApplicationsPerUser=10000, Abs Cap:1.0, Cap: 1.0, MaxCap : 1.0
```



```
2024-08-08 11:09:22,247 INFO handler.ContextHandler: Started o.e.j.w.WebAppContext@46d0f89c{node,/,file:///C:/Users/admi
n/AppData/Local/Temp/jetty-0_0_0_0-8042-hadoop-yarn-common-3_4_0_jar-_-any-12092403976610543110/webapp/,AVAILABLE}{jar:f
ile:/C:/hadoop-3.4.0/share/hadoop/yarn/hadoop-yarn-common-3.4.0.jar!/webapps/node}
2024-08-08 11:09:22,258 INFO server.AbstractConnector: Started ServerConnector@767191b1{HTTP/1.1, (http/1.1)}{0.0.0.0:80
42}
2024-08-08 11:09:22,259 INFO server.Server: Started @11169ms
2024-08-08 11:09:22,262 INFO webapp.WebApps: Web app node started at 8042
2024-08-08 11:09:22,263 INFO nodemanager.NodeStatusUpdaterImpl: Node ID assigned is : 31D-LAB5-27.SVV.local:53781.
2024-08-08 11:09:22,265 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2024-08-08 11:09:22,274 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8031
2024-08-08 11:09:22,341 INFO nodemanager.NodeStatusUpdaterImpl: Running Applications Size : 0.
2024-08-08 11:09:22,688 INFO security.NMContainerTokenSecretManager: Rolling master-key for container-tokens, got key wi
th id 1771827279
2024-08-08 11:09:22,690 INFO security.NMTokenSecretManagerInNM: Rolling master-key for container-tokens, got key with id
 -2074434580
2024-08-08 11:09:22,690 INFO nodemanager.NodeStatusUpdaterImpl: Registered with ResourceManager as 31D-LAB5-27.SVV.local
:53781 with total resource of <memory:8192, vCores:8>
```



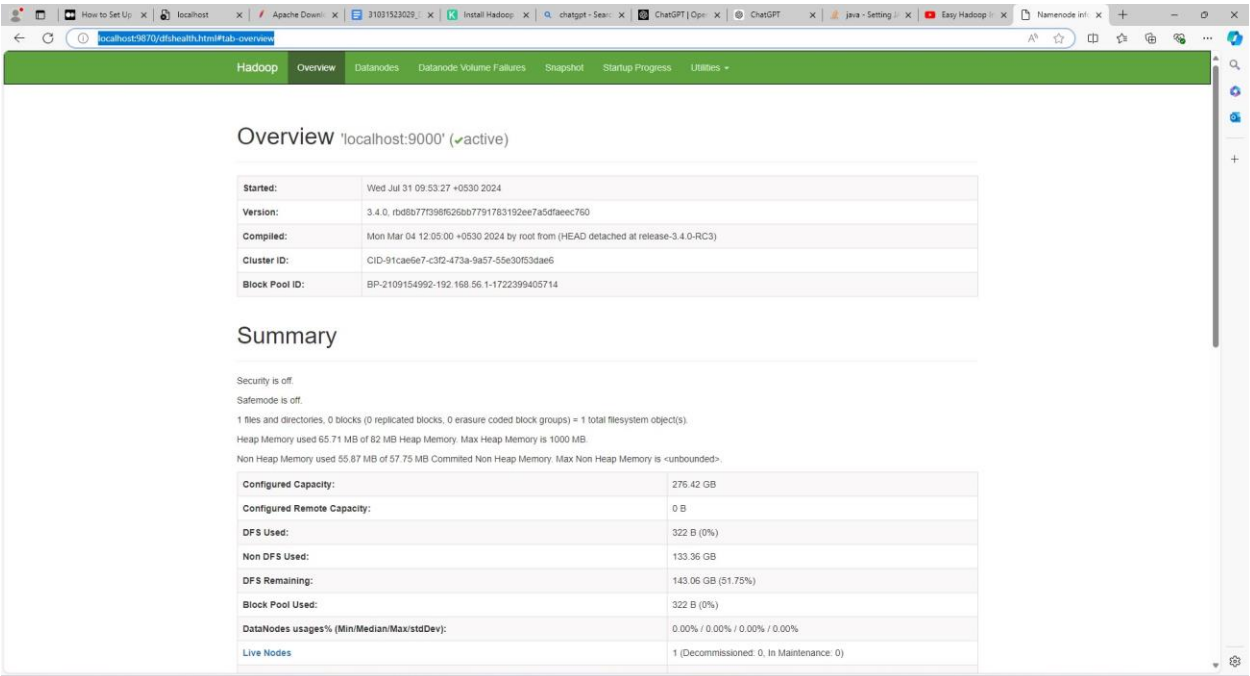
```
23.0.78-1723093372285: 5ms
2024-08-08 11:09:19,227 INFO checker.ThrottledAsyncChecker: Scheduling a check for C:\hadoop-3.4.0\data\datanode
2024-08-08 11:09:19,238 INFO checker.DatasetVolumeChecker: Scheduled health check for volume C:\hadoop-3.4.0\data\datano
de
2024-08-08 11:09:19,257 INFO datanode.VolumeScanner: VolumeScanner(C:\hadoop-3.4.0\data\datanode, DS-b3aa26ad-cd51-46ae-
963c-96bc8eb44c2f): no suitable block pools found to scan.  Waiting 1812234294 ms.
2024-08-08 11:09:19,261 WARN datanode.DirectoryScanner: dfs.datanode.directoryscan.throttle.limit.ms.per.sec set to valu
e above 1000 ms/sec. Assuming default value of -1
2024-08-08 11:09:19,262 INFO datanode.DirectoryScanner: Periodic Directory Tree Verification scan starting in 19854927ms
 with interval of 21600000ms and throttle limit of -1ms/s
2024-08-08 11:09:19,269 INFO datanode.DataNode: Block pool BP-1207686594-172.23.0.78-1723093372285 (Datanode Uuid 0b5e09
85-a5b9-4bd2-b889-6041194b49e9) service to localhost/127.0.0.1:9000 beginning handshake with NN: localhost/127.0.0.1:900
0.
2024-08-08 11:09:19,346 INFO datanode.DataNode: Block pool BP-1207686594-172.23.0.78-1723093372285 (Datanode Uuid 0b5e09
85-a5b9-4bd2-b889-6041194b49e9) service to localhost/127.0.0.1:9000 successfully registered with NN: localhost/127.0.0.1
:9000.
2024-08-08 11:09:19,347 INFO datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 2160
0000msecs CACHEREPORT_INTERVAL of 10000msecs Initial delay: 0msecs; heartBeatInterval=3000
2024-08-08 11:09:19,484 INFO datanode.DataNode: After receiving heartbeat response, updating state of namenode localhost
:9000 to active
2024-08-08 11:09:19,574 INFO datanode.DataNode: Successfully sent block report 0xaabaf885185d3c7 with lease ID 0x845aa7d
a3bab2575 to namenode: localhost/127.0.0.1:9000,  containing 1 storage report(s), of which we sent 1. The reports had 0
total blocks and used 1 RPC(s). This took 7 msecs to generate and 80 msecs for RPC and NN processing. Got back one comma
nd: FinalizeCommand/5.
2024-08-08 11:09:19,575 INFO datanode.DataNode: Got finalize command for block pool BP-1207686594-172.23.0.78-1723093372
285
```



Apache Hadoop Distribution - hadoop namenode

```
304 0 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,318 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=NameNodeStatus HTTP/1.1" 200 423 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win
64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,357 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=NameNodeInfo HTTP/1.1" 200 2958 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,402 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=FSNamesystem HTTP/1.1" 200 2765 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,454 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=ReplicatedBlocksState HTTP/1.1" 200 445 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10
.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,467 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /conf HTTP/1.1" 200 2646
70 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,494 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=ECBlockGroupsState HTTP/1.1" 200 422 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0;
 Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,515 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=FSNamesystemState HTTP/1.1" 200 1851 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0;
 Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,523 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=java.lang:type=
Memory HTTP/1.1" 200 512 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
37.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
2024-08-08 11:09:26,523 INFO requests.namenode: 127.0.0.1 - - [08/Aug/2024:05:39:26 +0000] "GET /jmx?qry=Hadoop:service=
NameNode,name=BlockStats HTTP/1.1" 200 526 "http://localhost:9870/dfshealth.html" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.0.0"
```

20. Go to your browser and type localhost:9870 to view Hadoop Page.



localhost:9870/dfshealth.html#tab-overview

Overview 'localhost:9000' (✓active)

Started:	Wed Jul 31 09:53:27 +0530 2024
Version:	3.4.0, rbd8b77f398f626bb7791783192ee7a5dfaeec760
Compiled:	Mon Mar 04 12:05:00 +0530 2024 by root from (HEAD detached at release-3.4.0-RC3)
Cluster ID:	CID-91cae6e7-c3f2-473a-9a57-55e30f53dae6
Block Pool ID:	BP-2109154992-192.168.56.1-1722399405714

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

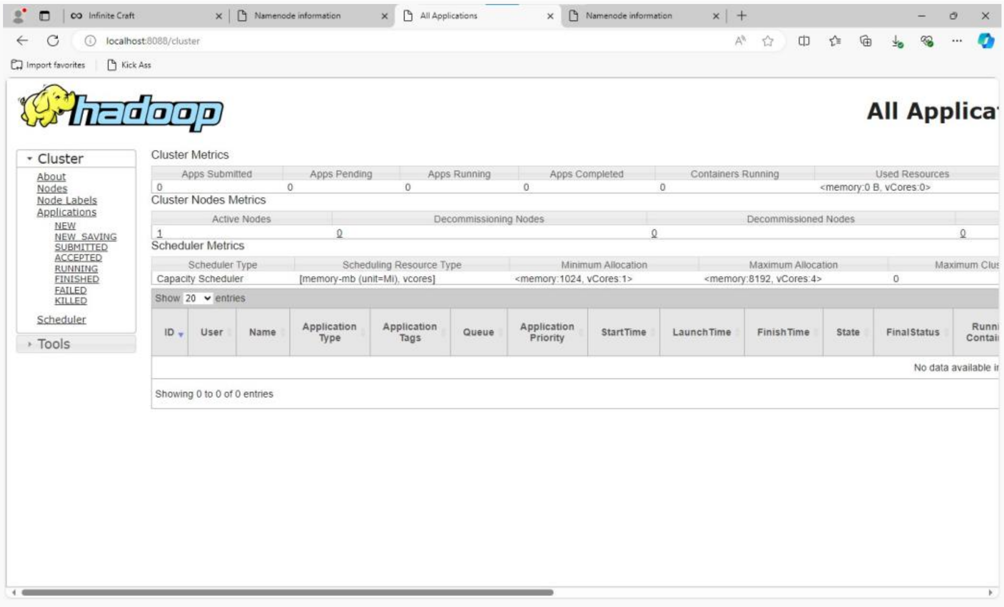
Heap Memory used 65.71 MB of 82 MB Heap Memory. Max Heap Memory is 1000 MB.

Non Heap Memory used 55.87 MB of 57.75 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	276.42 GB
Configured Remote Capacity:	0 B
DFS Used:	322 B (0%)
Non DFS Used:	133.36 GB
DFS Remaining:	143.06 GB (51.75%)
Block Pool Used:	322 B (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)

21. If your yarn process doesnt start then use java version 11.

22. After the yarn process has started visit localhost:8088



All Applica

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources
0	0	0	0	0	<memory:0 B, vCores:0>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	
1	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Clus
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Runni Contai
												No data available i

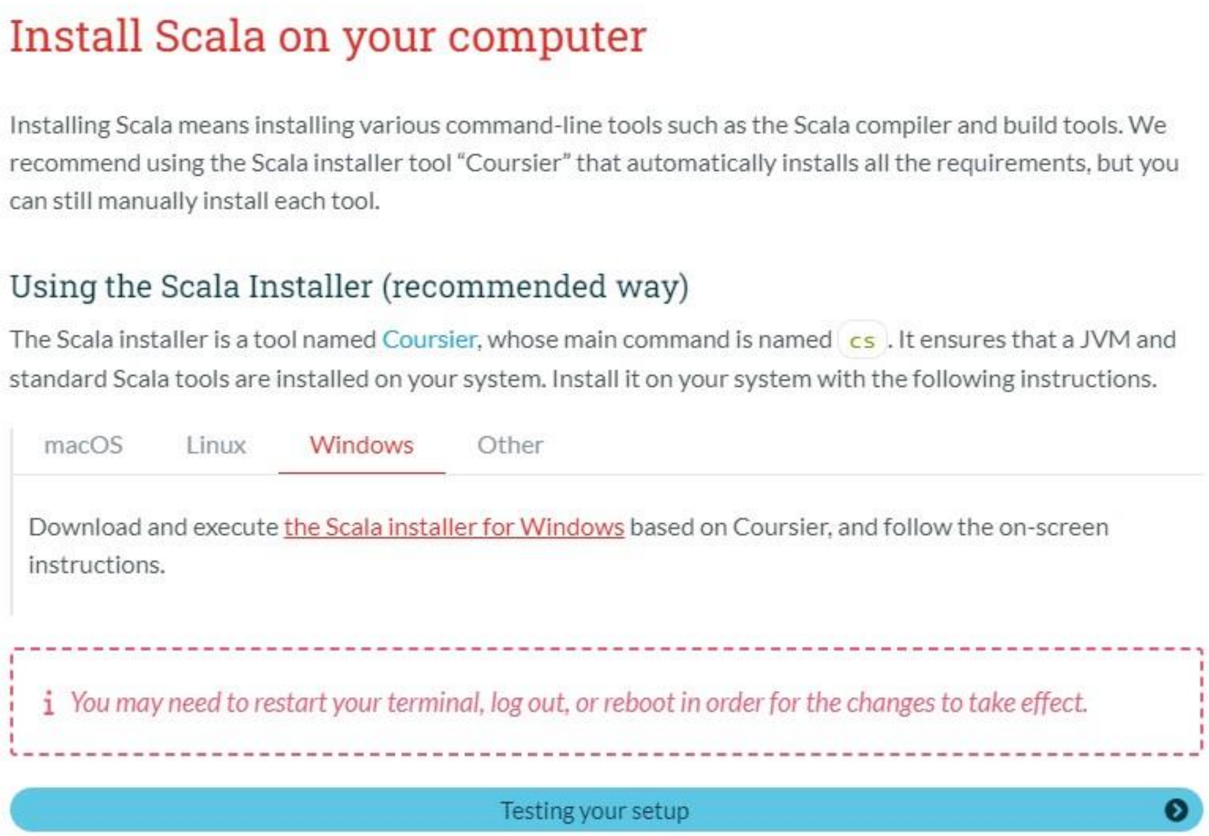
Showing 0 to 0 of 0 entries

Practical 2

Aim : Getting started with Scala

Steps :

1. Get Scala for Windows from the below link Getting Started | Scala Documentation (scala-lang.org)



Install Scala on your computer

Installing Scala means installing various command-line tools such as the Scala compiler and build tools. We recommend using the Scala installer tool "Coursier" that automatically installs all the requirements, but you can still manually install each tool.

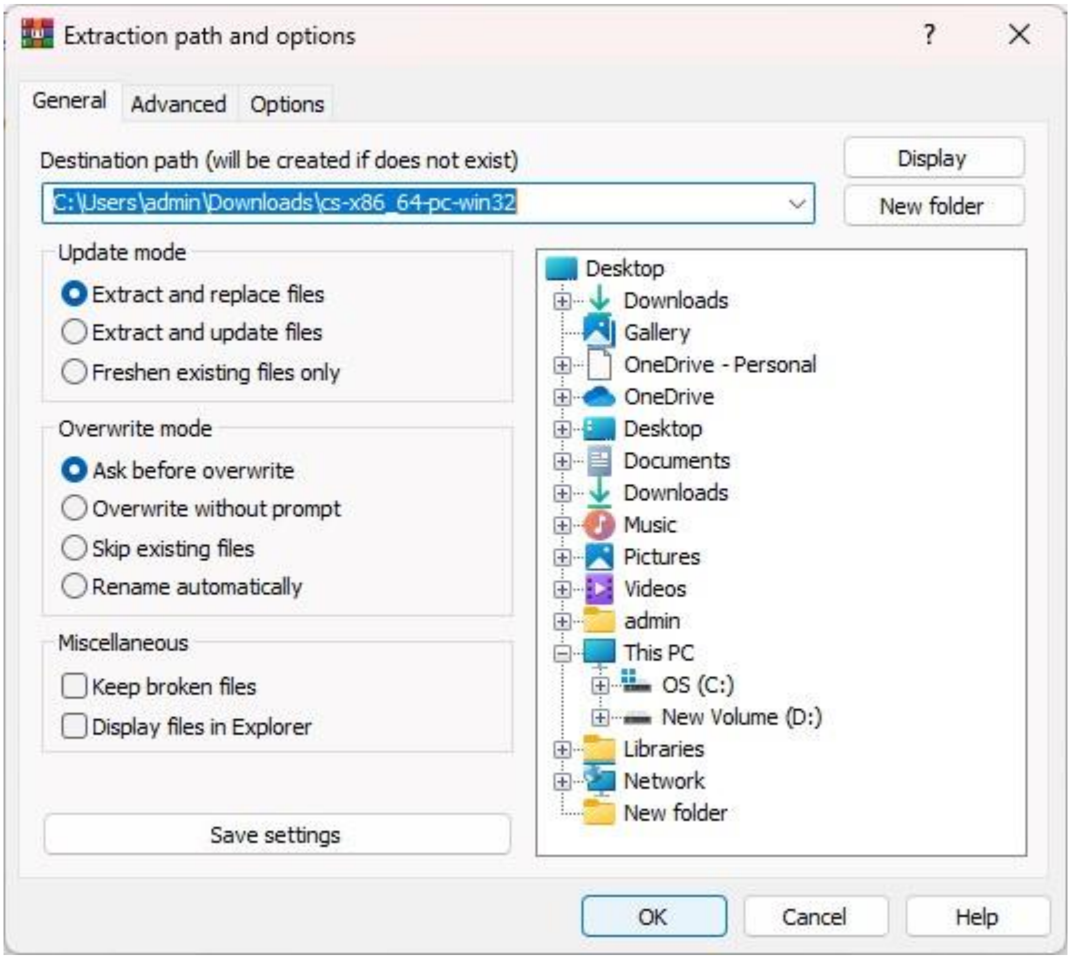
Using the Scala Installer (recommended way)

The Scala installer is a tool named Coursier, whose main command is named `cs`. It ensures that a JVM and standard Scala tools are installed on your system. Install it on your system with the following instructions.

Download and execute the Scala installer for Windows based on Coursier, and follow the on-screen instructions.

i You may need to restart your terminal, log out, or reboot in order for the changes to take effect.

2. Extract the given archive and run the executable file.



Extraction path and options

General | Advanced | Options

Destination path (will be created if does not exist)

C:\Users\admin\Downloads\cs-x86_64-pc-win32

Update mode

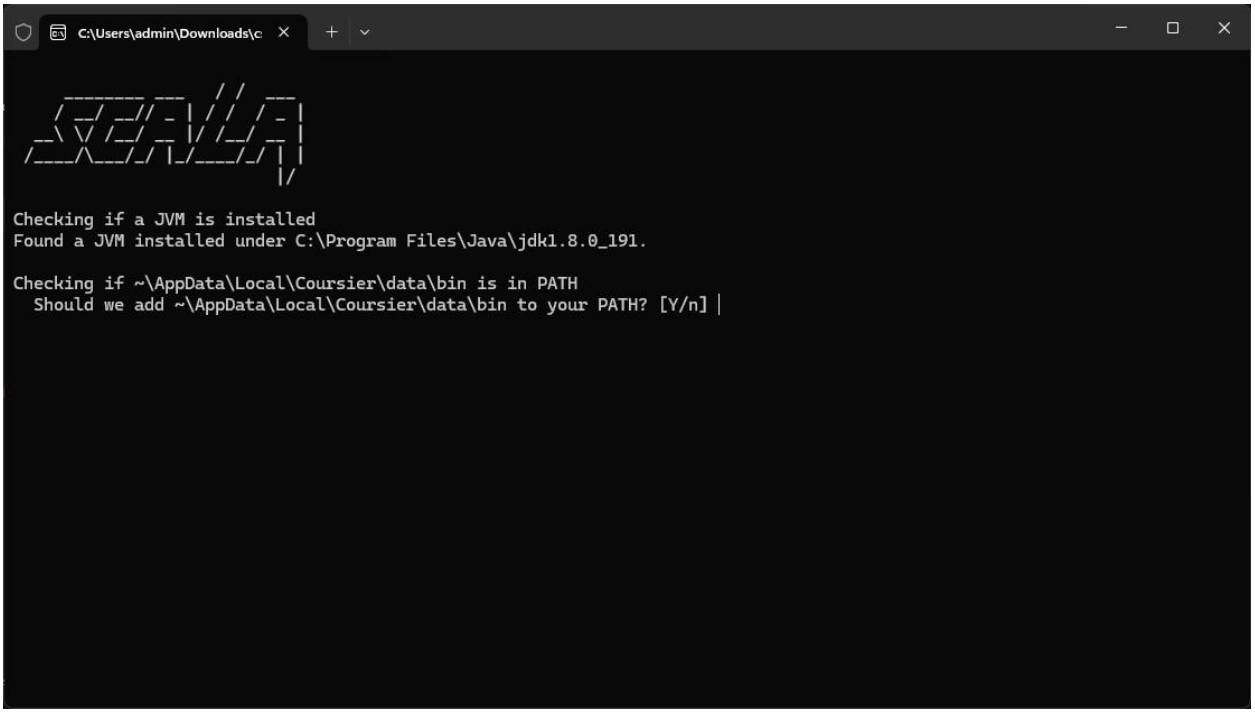
● Extract and replace files
○ Extract and update files
○ Freshen existing files only

Overwrite mode

● Ask before overwrite
○ Overwrite without prompt
○ Skip existing files
○ Rename automatically

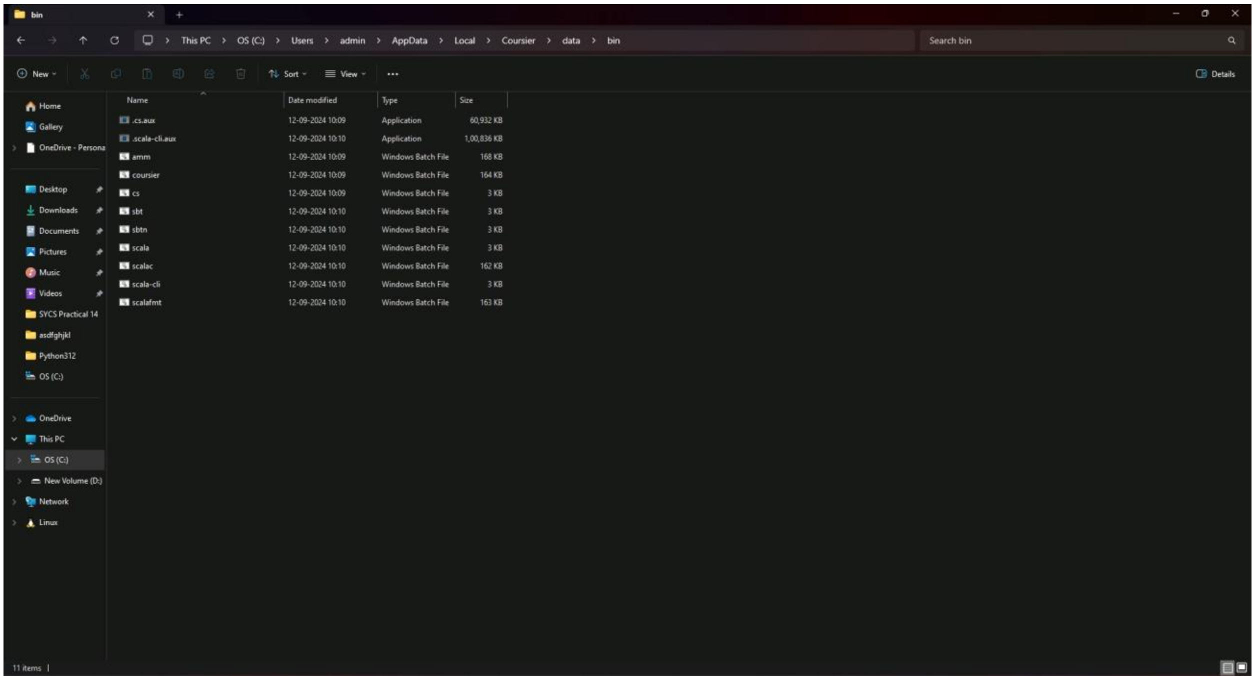
Miscellaneous

□ Keep broken files
□ Display files in Explorer



```
     ________ ___   / /  ___
    / __/ __// _ | / /  / _ |
  __\ \/ /__/ __ |/ /__/ __ |
 /____/\___/_/ |_/____/_/ | |
                          |/

Checking if a JVM is installed
Found a JVM installed under C:\Program Files\Java\jdk1.8.0_191.

Checking if ~\AppData\Local\Coursier\data\bin is in PATH
  Should we add ~\AppData\Local\Coursier\data\bin to your PATH? [Y/n] |
```

3. Go to the following path and then copy it.

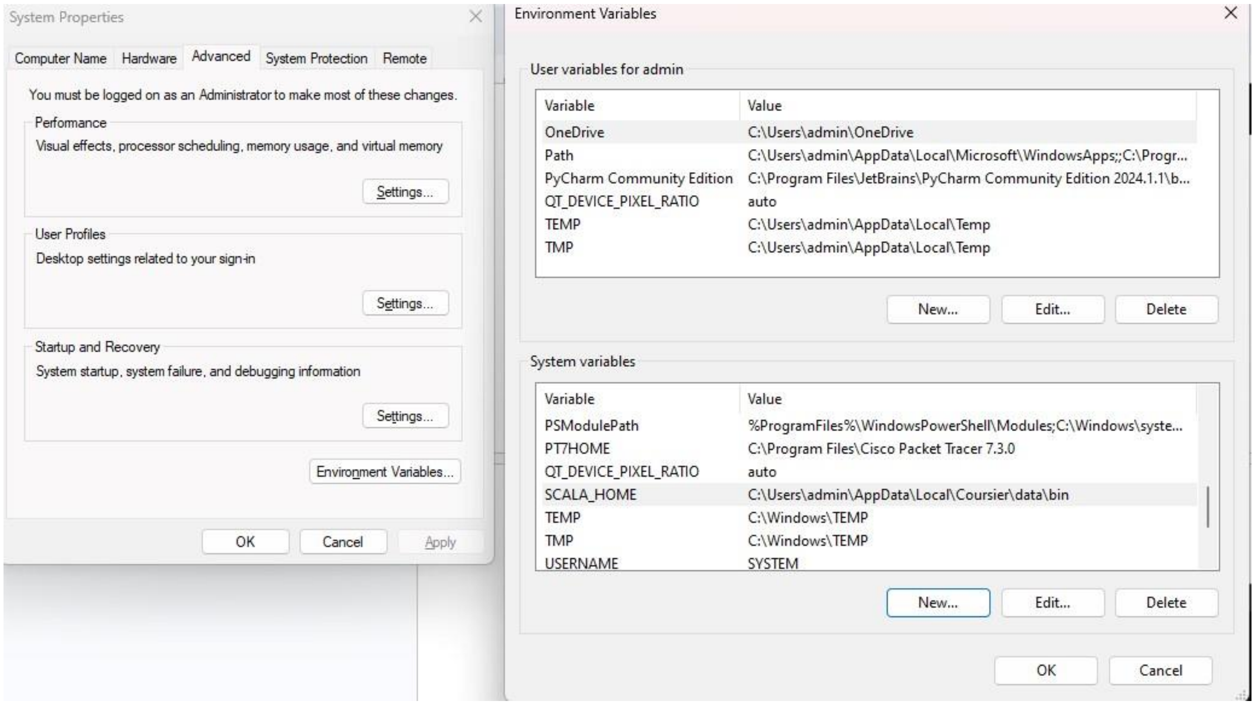


This PC > OS (C:) > Users > admin > AppData > Local > Coursier > data > bin

Name	Date modified	Type	Size
.cs.aux	12-09-2024 10:09	Application	60,932 KB
.scala-cli.aux	12-09-2024 10:10	Application	1,00,836 KB
amm	12-09-2024 10:09	Windows Batch File	168 KB
coursier	12-09-2024 10:09	Windows Batch File	164 KB
cs	12-09-2024 10:09	Windows Batch File	3 KB
sbt	12-09-2024 10:10	Windows Batch File	3 KB
sbtn	12-09-2024 10:10	Windows Batch File	3 KB
scala	12-09-2024 10:10	Windows Batch File	3 KB
scalac	12-09-2024 10:10	Windows Batch File	162 KB
scala-cli	12-09-2024 10:10	Windows Batch File	3 KB
scalafmt	12-09-2024 10:10	Windows Batch File	163 KB

11 items

4. Set the SCALA_HOME environment variable.



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

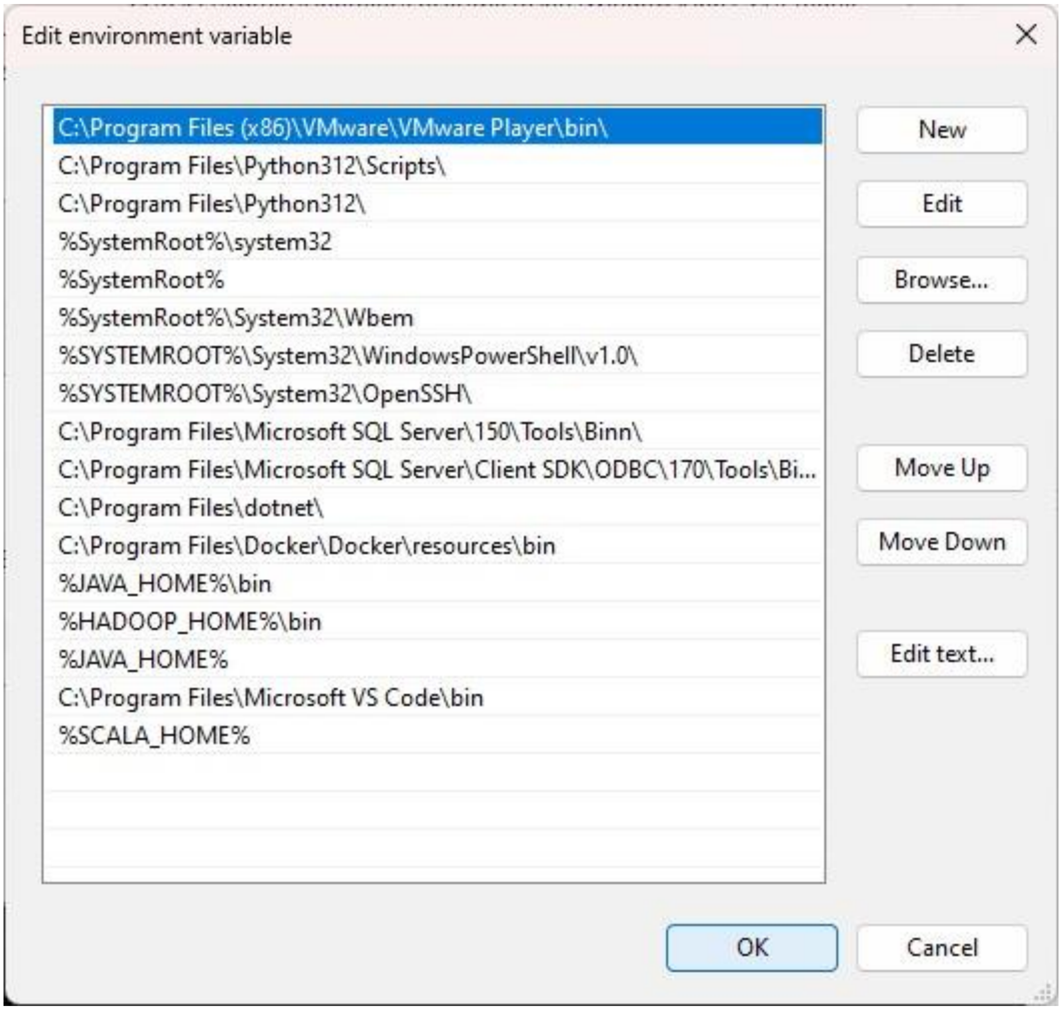
Environment Variables

User variables for admin

Variable	Value
OneDrive	C:\Users\admin\OneDrive
Path	C:\Users\admin\AppData\Local\Microsoft\WindowsApps;;C:\Progr...
PyCharm Community Edition	C:\Program Files\JetBrains\PyCharm Community Edition 2024.1.1\b...
QT_DEVICE_PIXEL_RATIO	auto
TEMP	C:\Users\admin\AppData\Local\Temp
TMP	C:\Users\admin\AppData\Local\Temp

System variables

Variable	Value
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\syste...
PT7HOME	C:\Program Files\Cisco Packet Tracer 7.3.0
QT_DEVICE_PIXEL_RATIO	auto
SCALA_HOME	C:\Users\admin\AppData\Local\Coursier\data\bin
TEMP	C:\Windows\TEMP
TMP	C:\Windows\TEMP
USERNAME	SYSTEM

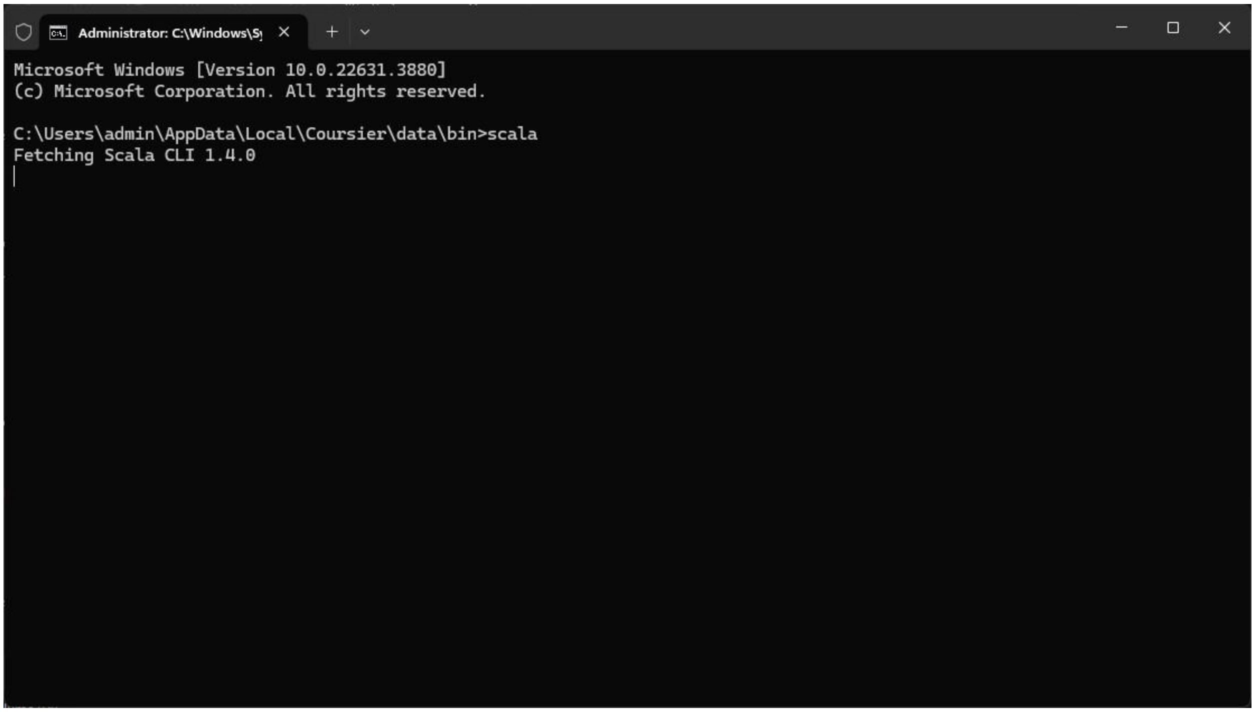


Edit environment variable

```
C:\Program Files (x86)\VMware\VMware Player\bin\
C:\Program Files\Python312\Scripts\
C:\Program Files\Python312\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Bi...
C:\Program Files\dotnet\
C:\Program Files\Docker\Docker\resources\bin
%JAVA_HOME%\bin
%HADOOP_HOME%\bin
%JAVA_HOME%
C:\Program Files\Microsoft VS Code\bin
%SCALA_HOME%
```

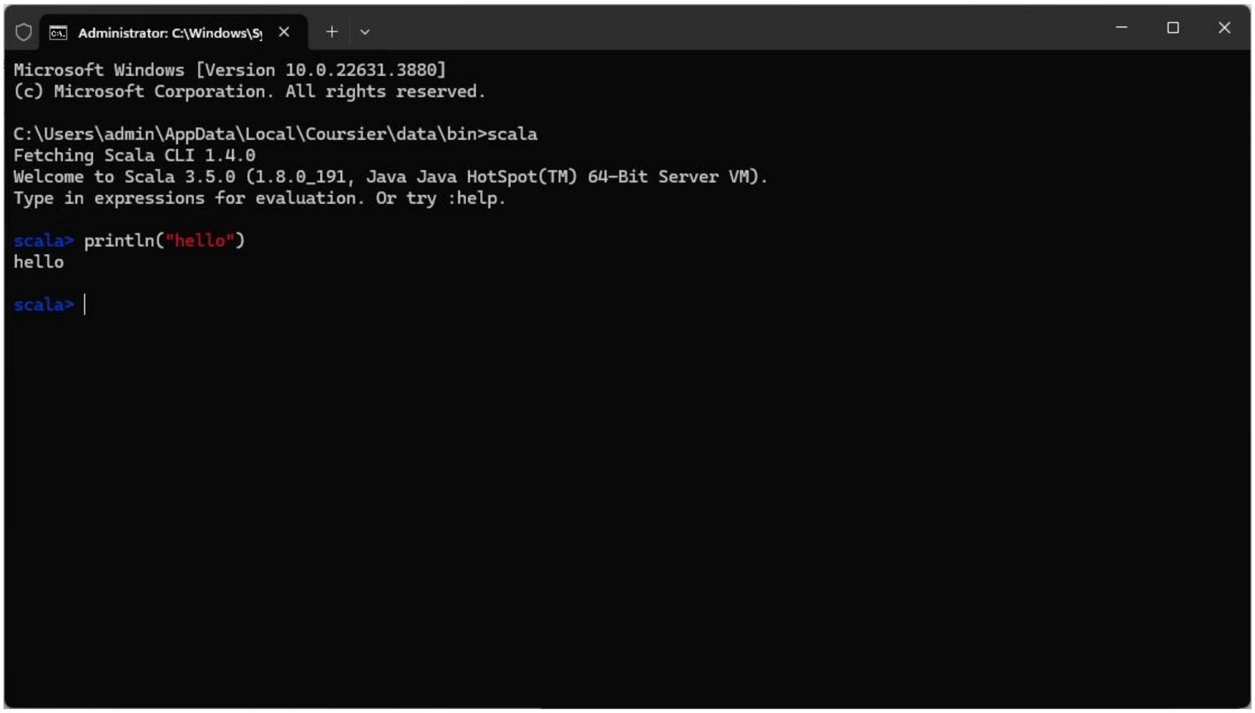
5. Open a command prompt on this path and type “scala”. It will download the Scala CLI.

   C:\Users\admin\AppData\Local\Coursier\data\bin



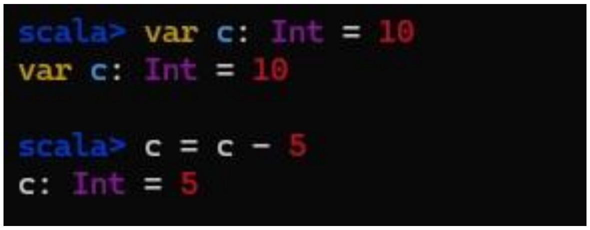
```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\AppData\Local\Coursier\data\bin>scala
Fetching Scala CLI 1.4.0
```

6. Test the working with a command or prompt ‘println(“Hello”)’.

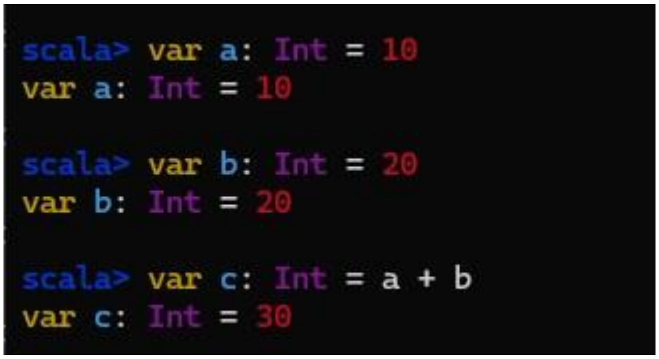


```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\AppData\Local\Coursier\data\bin>scala
Fetching Scala CLI 1.4.0
Welcome to Scala 3.5.0 (1.8.0_191, Java Java HotSpot(TM) 64-Bit Server VM).
Type in expressions for evaluation. Or try :help.

scala> println("hello")
hello

scala>
```

‘Var’ before a variable name is used to make the variable mutable.

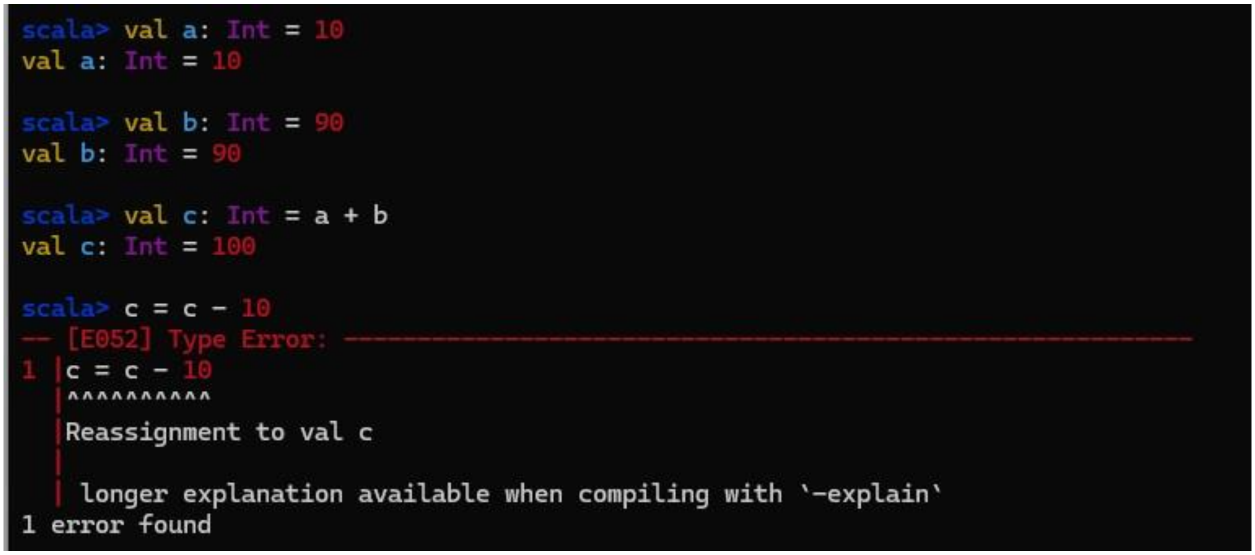


```
scala> var c: Int = 10
var c: Int = 10

scala> c = c - 5
c: Int = 5
```



```
scala> var a: Int = 10
var a: Int = 10

scala> var b: Int = 20
var b: Int = 20

scala> var c: Int = a + b
var c: Int = 30
```

‘Val’ is used to make the variable immutable

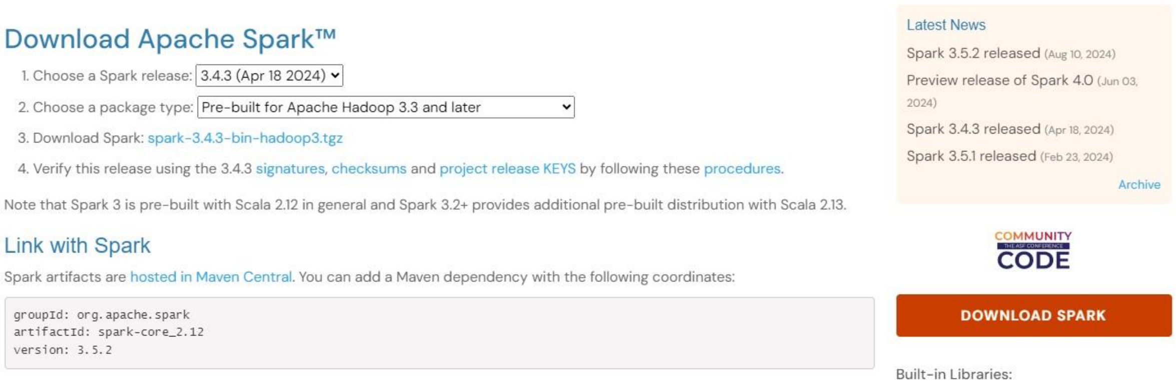


```
scala> val a: Int = 10
val a: Int = 10

scala> val b: Int = 90
val b: Int = 90

scala> val c: Int = a + b
val c: Int = 100

scala> c = c - 10
-- [E052] Type Error: -----------------------------------------------------------
1 |c = c - 10
  |^^^^^^^^^^
  |Reassignment to val c
  |
  | longer explanation available when compiling with `-explain`
1 error found
```

Installing Spark

Steps

1. Download Spark from the following link [Downloads | Apache Spark]



Download Apache Spark™

1. Choose a Spark release: 3.4.3 (Apr 18 2024)
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later
3. Download Spark: spark-3.4.3-bin-hadoop3.tgz
4. Verify this release using the 3.4.3 signatures, checksums and project release KEYS by following these procedures.

Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13.

Link with Spark

Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:

```
groupId: org.apache.spark
artifactId: spark-core_2.12
version: 3.5.2
```

Latest News

Spark 3.5.2 released (Aug 10, 2024)

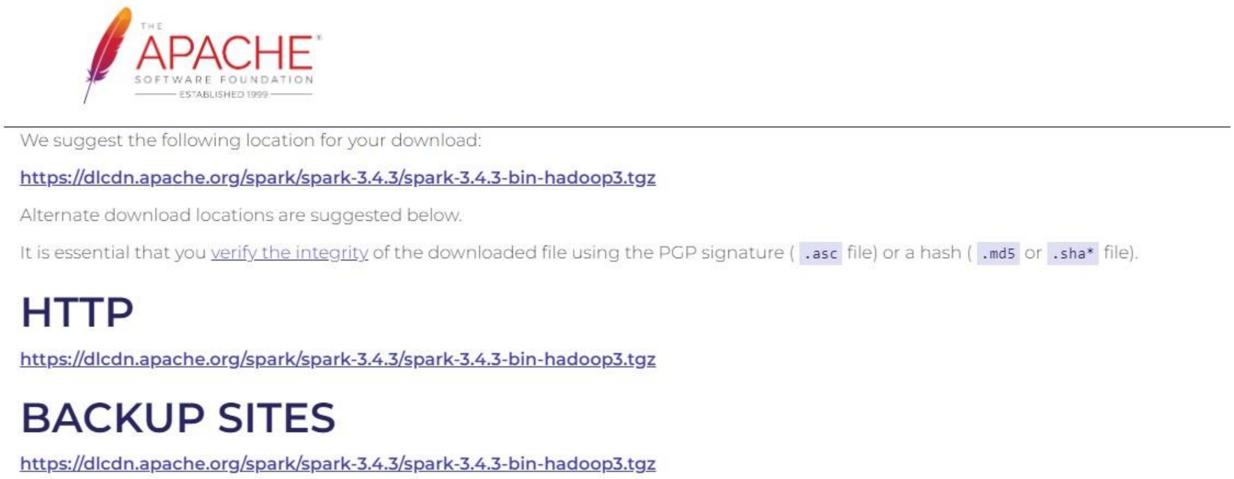
Preview release of Spark 4.0 (Jun 03, 2024)

Spark 3.4.3 released (Apr 18, 2024)

Spark 3.5.1 released (Feb 23, 2024)

Archive

Built-in Libraries:



We suggest the following location for your download:

https://dlcdn.apache.org/spark/spark-3.4.3/spark-3.4.3-bin-hadoop3.tgz

Alternate download locations are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

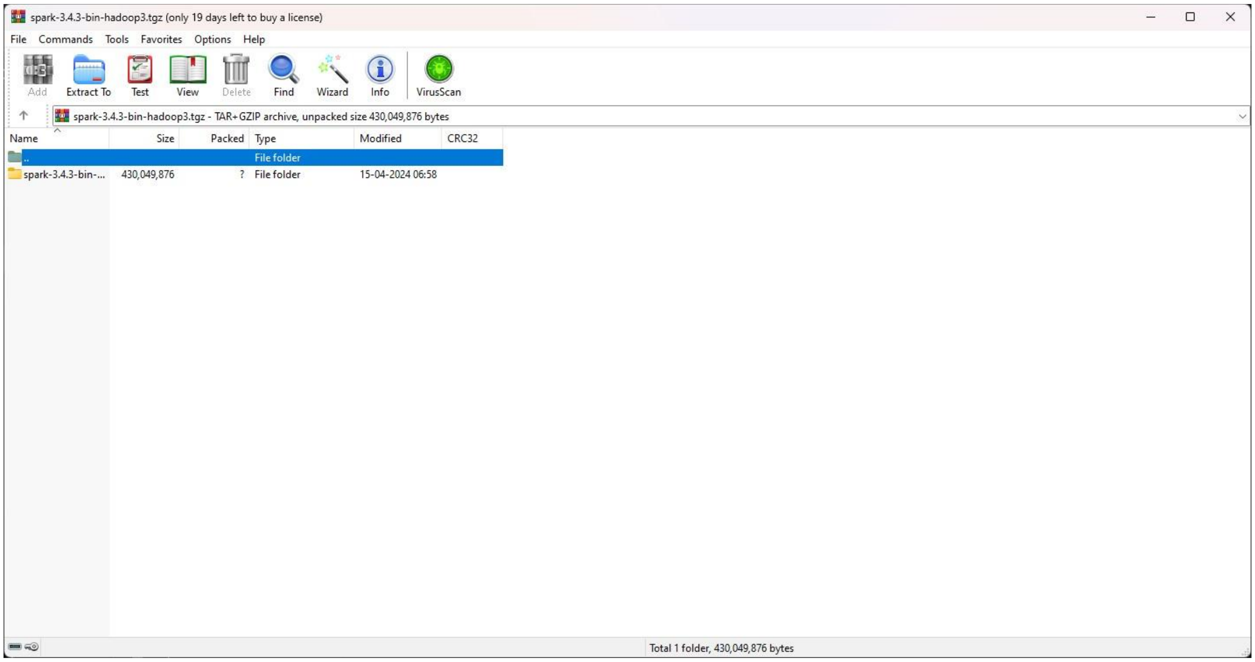
HTTP

https://dlcdn.apache.org/spark/spark-3.4.3/spark-3.4.3-bin-hadoop3.tgz

BACKUP SITES

https://dlcdn.apache.org/spark/spark-3.4.3/spark-3.4.3-bin-hadoop3.tgz

2. Extract the files and set the path for its home directory

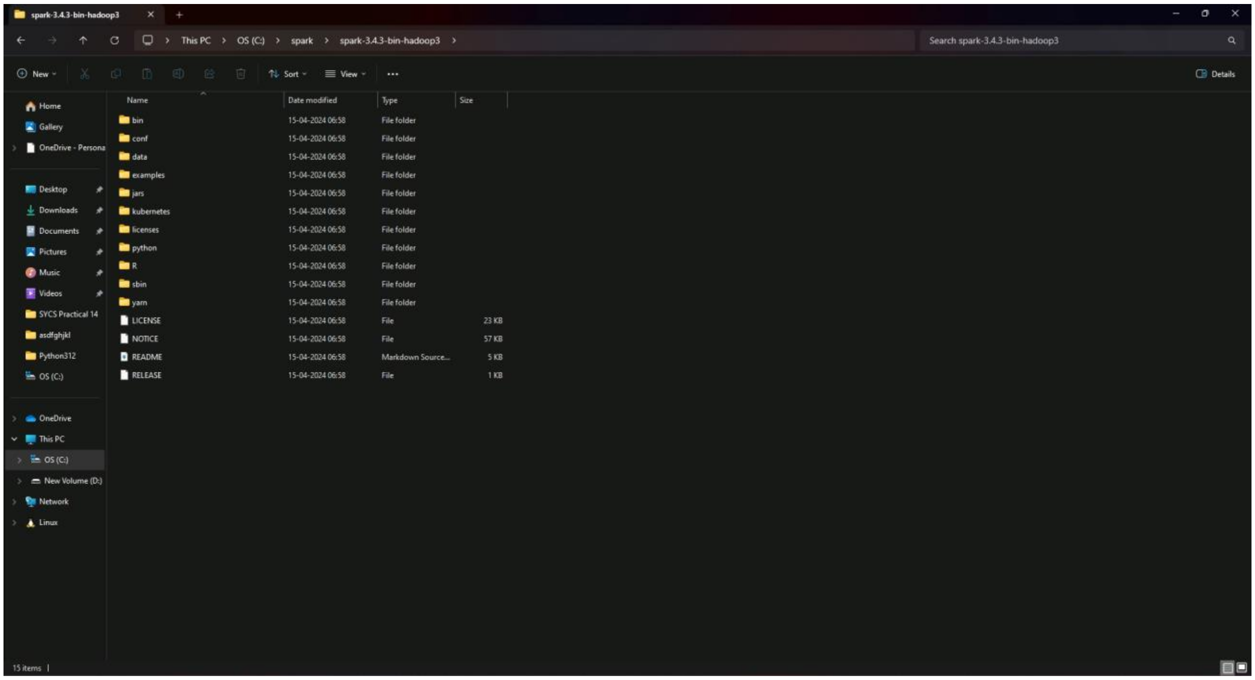


spark-3.4.3-bin-hadoop3.tgz (only 19 days left to buy a license)

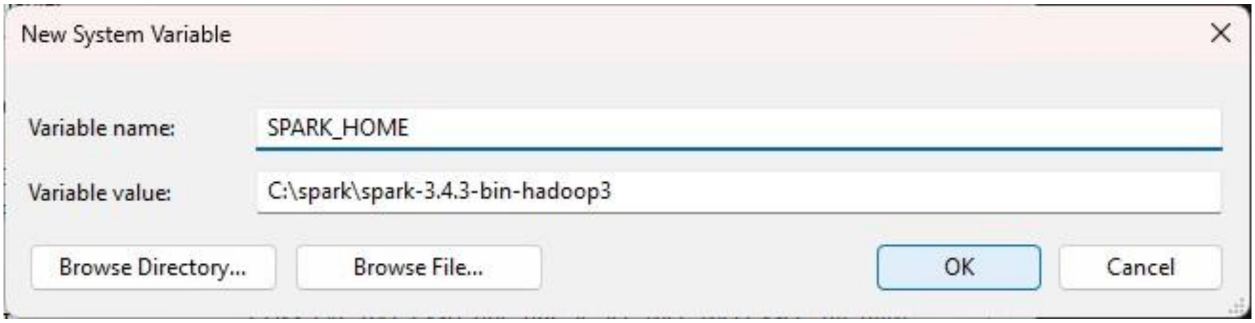
spark-3.4.3-bin-hadoop3.tgz - TAR+GZIP archive, unpacked size 430,049,876 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
spark-3.4.3-bin-...	430,049,876	?	File folder	15-04-2024 06:58	

Total 1 folder, 430,049,876 bytes



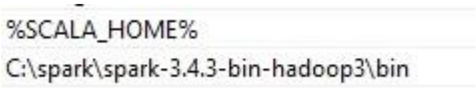
Name	Date modified	Type	Size
bin	15-04-2024 06:58	File folder	
conf	15-04-2024 06:58	File folder	
data	15-04-2024 06:58	File folder	
examples	15-04-2024 06:58	File folder	
jars	15-04-2024 06:58	File folder	
kubernetes	15-04-2024 06:58	File folder	
licenses	15-04-2024 06:58	File folder	
python	15-04-2024 06:58	File folder	
R	15-04-2024 06:58	File folder	
sbin	15-04-2024 06:58	File folder	
yarn	15-04-2024 06:58	File folder	
LICENSE	15-04-2024 06:58	File	23 KB
NOTICE	15-04-2024 06:58	File	57 KB
README	15-04-2024 06:58	Markdown Source...	5 KB
RELEASE	15-04-2024 06:58	File	1 KB



New System Variable

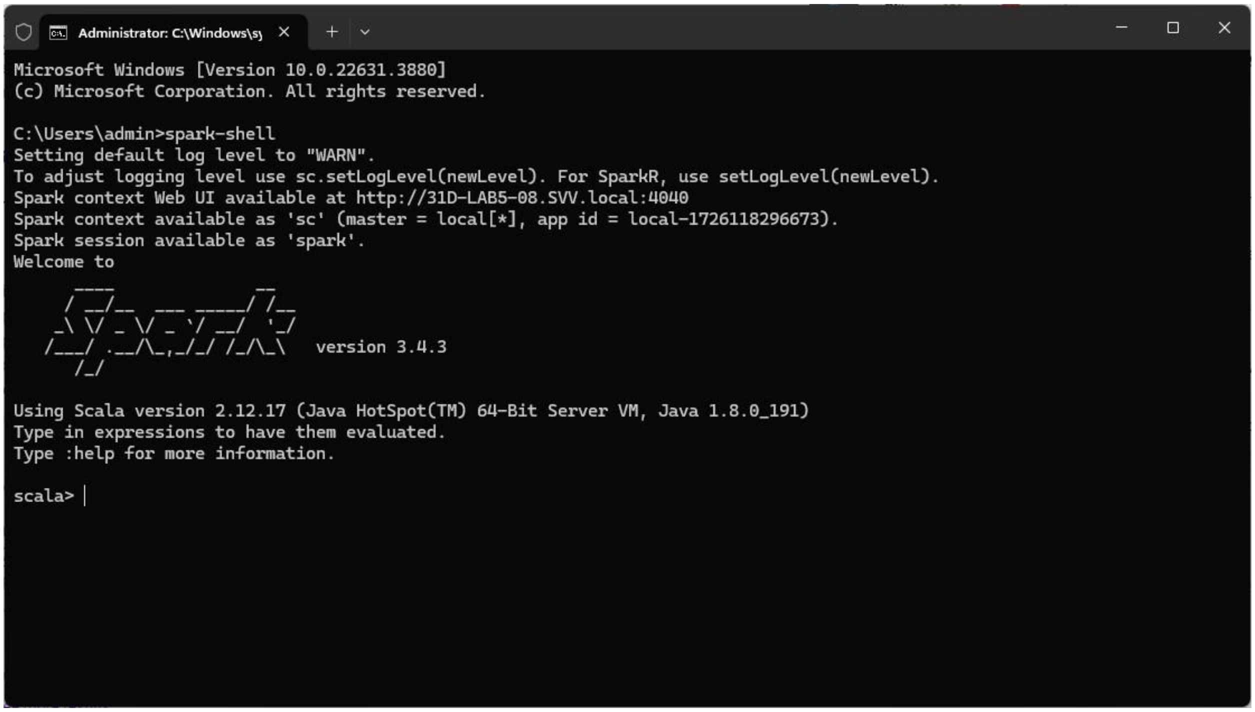
Variable name: SPARK_HOME

Variable value: C:\spark\spark-3.4.3-bin-hadoop3



%SCALA_HOME%
C:\spark\spark-3.4.3-bin-hadoop3\bin

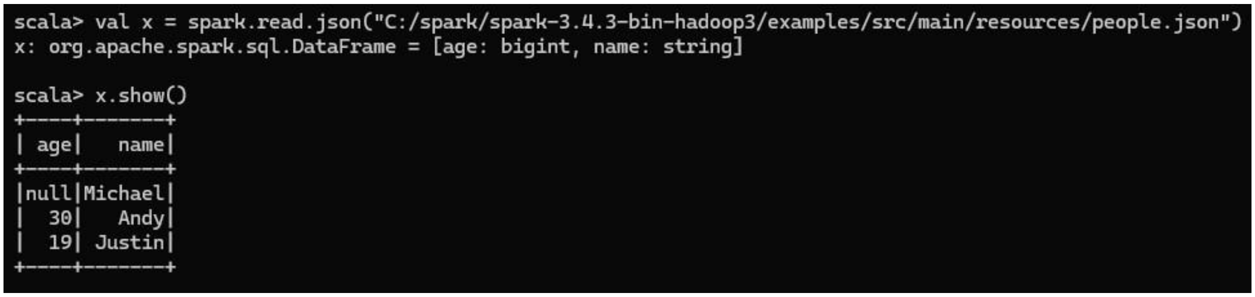
3. Check the installation with ‘spark-shell’



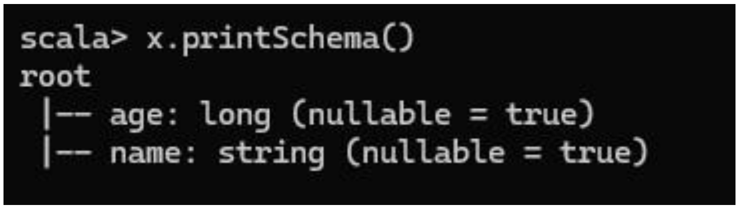
```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://31D-LAB5-08.SVV.local:4040
Spark context available as 'sc' (master = local[*], app id = local-1726118296673).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.3
      /_/

Using Scala version 2.12.17 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_191)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Lets try to show a sample data from this path

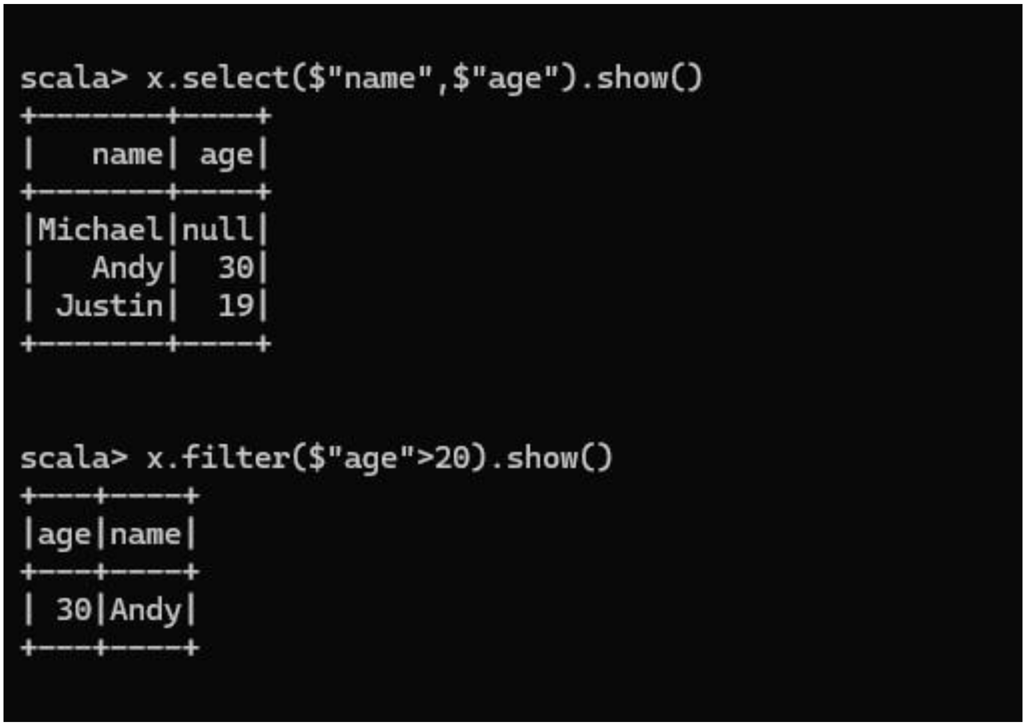
C:\spark\spark-3.4.3-bin-hadoop3\examples\src\main\resources\people.json



```
scala> val x = spark.read.json("C:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.json")
x: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> x.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

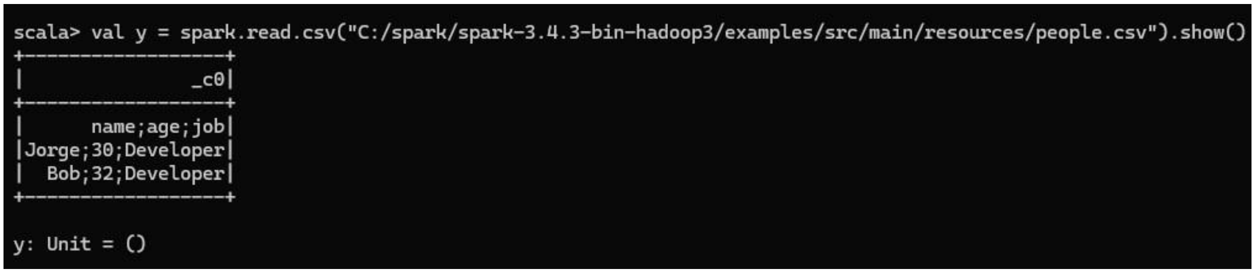


```
scala> x.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```



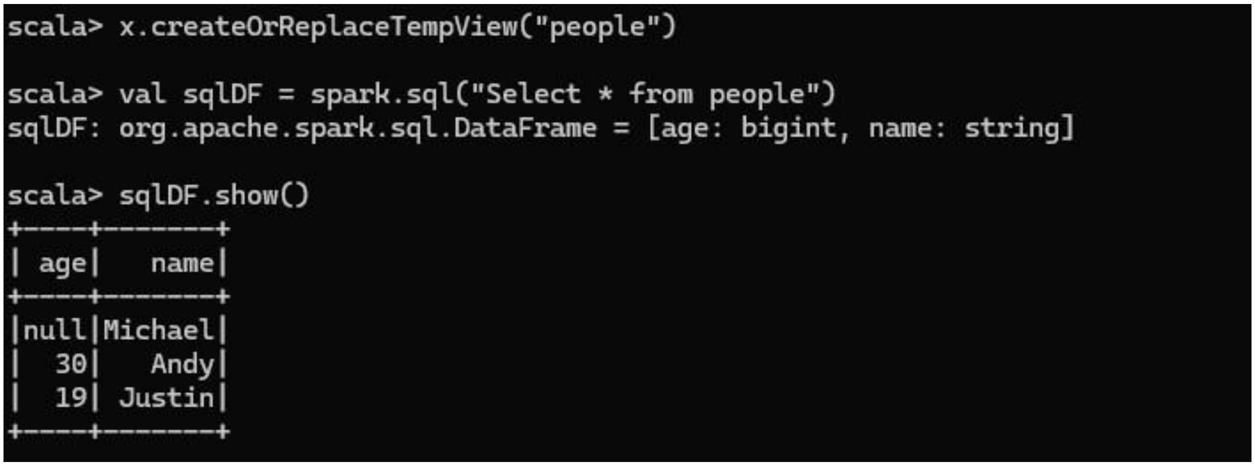
```
scala> x.select($"name",$"age").show()
+-------+----+
|   name| age|
+-------+----+
|Michael|null|
|   Andy|  30|
| Justin|  19|
+-------+----+


scala> x.filter($"age">20).show()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```

Reading CSV/Excel File



```
scala> val y = spark.read.csv("C:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.csv").show()
+------------------+
|               _c0|
+------------------+
|      name;age;job|
|Jorge;30;Developer|
|  Bob;32;Developer|
+------------------+

y: Unit = ()
```

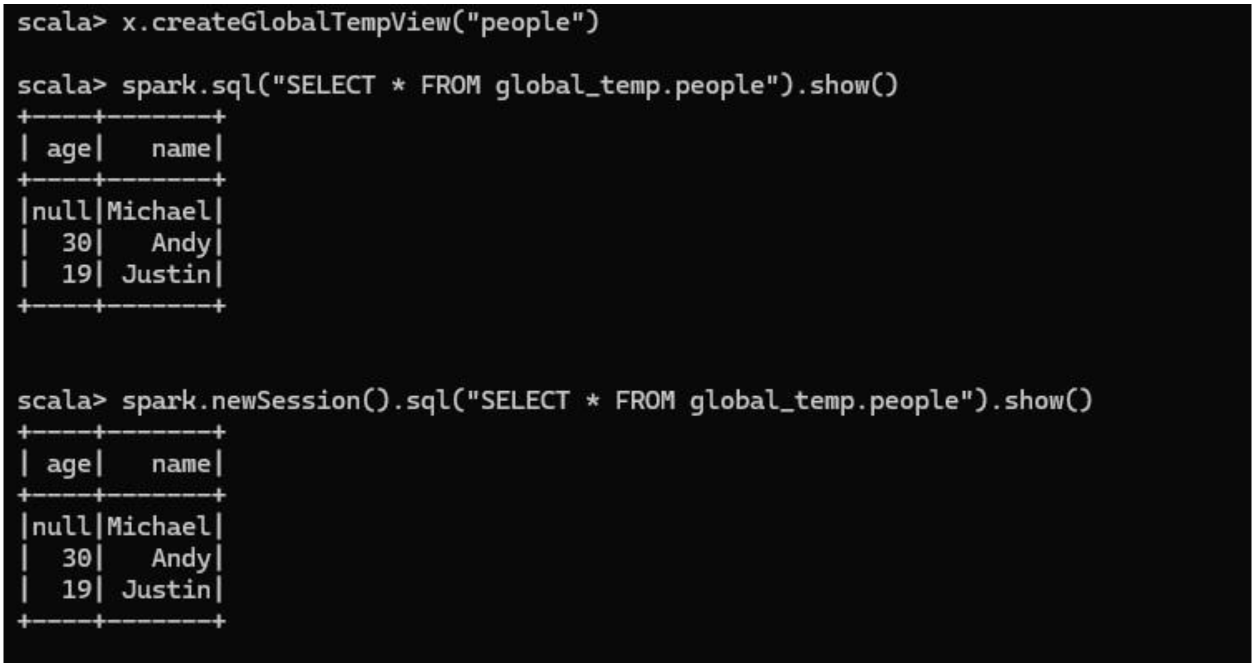
Creating an SQL Tempory View



```
scala> x.createOrReplaceTempView("people")

scala> val sqlDF = spark.sql("Select * from people")
sqlDF: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> sqlDF.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

Temporary views in Spark SQL are session-scoped and will disappear if the session that creates it terminates. If you want to have a temporary view that is shared among all sessions and keep alive until the Spark application terminates, you can create a global temporary view.

Global temporary view is tied to a system preserved database global_temp, and we must use the qualified name to refer it, e.g. SELECT * FROM global_temp.view.

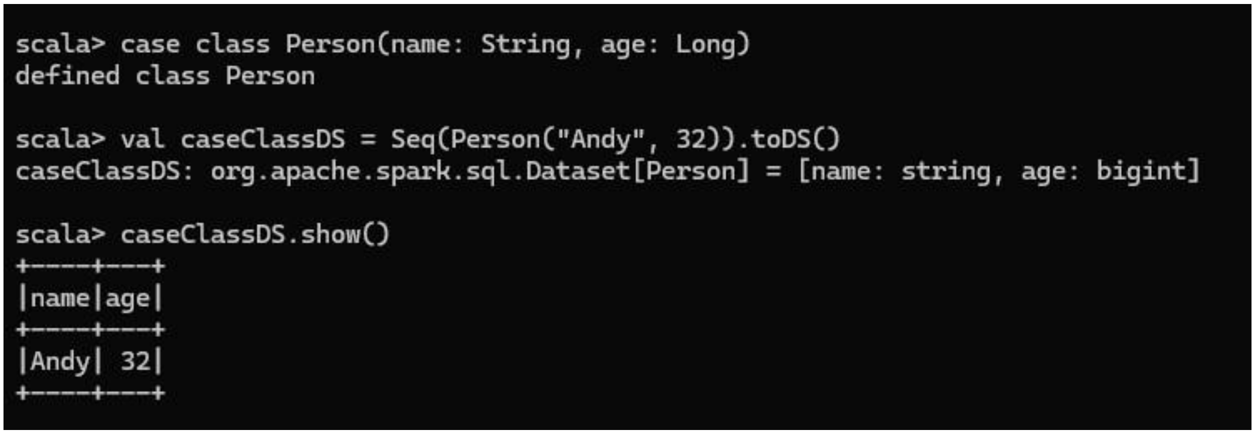


```
scala> x.createGlobalTempView("people")

scala> spark.sql("SELECT * FROM global_temp.people").show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+


scala> spark.newSession().sql("SELECT * FROM global_temp.people").show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

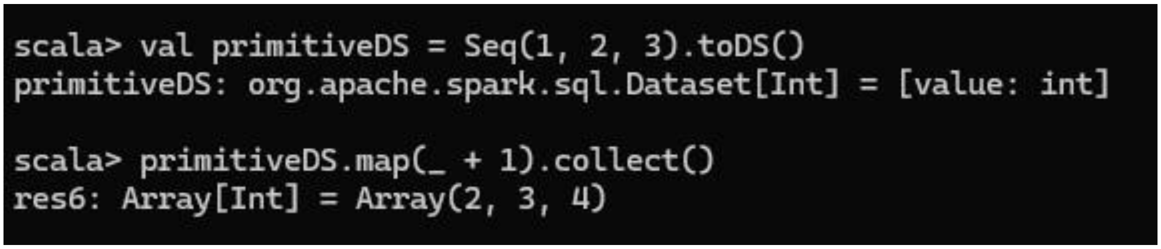
Creating Datasets

Datasets are similar to RDDs, however, instead of using Java serialization or Kryo they use a specialized Encoder to serialize the objects for processing or transmitting over the network. While both encoders and standard serialization are responsible for turning an object into bytes, encoders are code generated dynamically and use a format that allows Spark to perform many operations like filtering, sorting and hashing without deserializing the bytes back into an object.

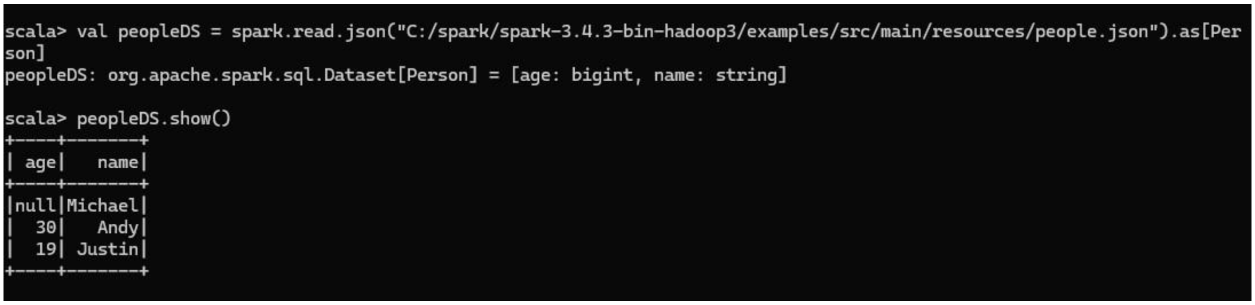
```
case class Person(name: String, age: Long)

// Encoders are created for case classes val
caseClassDS = Seq(Person("Andy", 32)).toDS()
caseClassDS.show()

// +----+---+

// |name|age|

// +----+---+

// |Andy| 32|
// +----+---+

// Encoders for most common types are automatically provided by importing spark.implicits._ val
primitiveDS = Seq(1, 2, 3).toDS() primitiveDS.map(_ + 1).collect() // Returns: Array(2, 3, 4)

// DataFrames can be converted to a Dataset by providing a class. Mapping will be done by name

val path = "examples/src/main/resources/people.json" val
peopleDS = spark.read.json(path).as[Person]
peopleDS.show()

// +----+-------+

// | age|   name|

// +----+-------+

// |null|Michael|

// |  30|   Andy|

// |  19| Justin|

// +----+-------+
```



```
scala> case class Person(name: String, age: Long)
defined class Person

scala> val caseClassDS = Seq(Person("Andy", 32)).toDS()
caseClassDS: org.apache.spark.sql.Dataset[Person] = [name: string, age: bigint]

scala> caseClassDS.show()
+----+---+
|name|age|
+----+---+
|Andy| 32|
+----+---+
```



```
scala> val primitiveDS = Seq(1, 2, 3).toDS()
primitiveDS: org.apache.spark.sql.Dataset[Int] = [value: int]

scala> primitiveDS.map(_ + 1).collect()
res6: Array[Int] = Array(2, 3, 4)
```



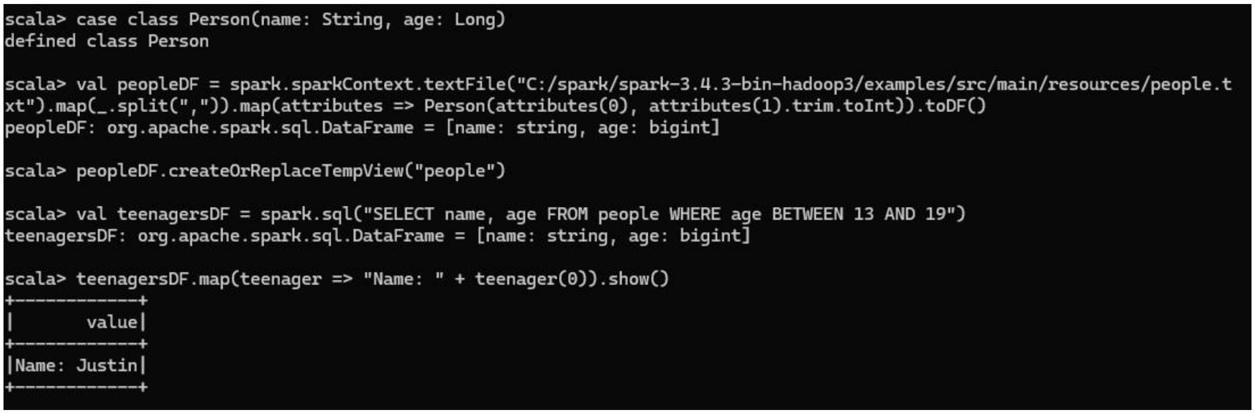
```
scala> val peopleDS = spark.read.json("C:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.json").as[Per
son]
peopleDS: org.apache.spark.sql.Dataset[Person] = [age: bigint, name: string]

scala> peopleDS.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

Inferring the Schema Using Reflection

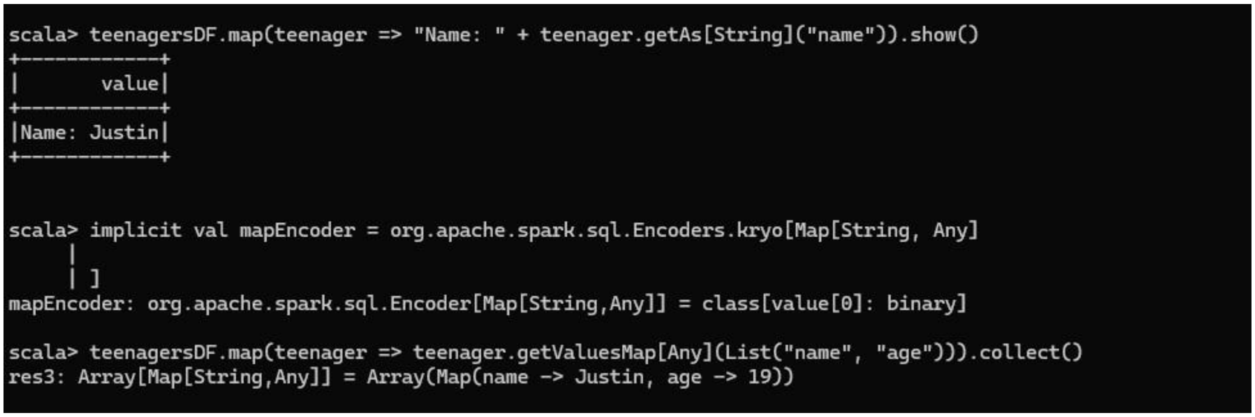
The Scala interface for Spark SQL supports automatically converting an RDD containing case classes to a DataFrame. The case class defines the schema of the table. The names of the arguments to the case class are read using reflection and become the names of the columns.

Case classes can also be nested or contain complex types such as Seqs or Arrays. This RDD can be implicitly converted to a DataFrame and then be registered as a table. Tables can be used in subsequent SQL statements.

```
// For implicit conversions from RDDs to DataFrames import
spark.implicits._

// Create an RDD of Person objects from a text file, convert it to a Dataframe val
peopleDF = spark.sparkContext

  .textFile("examples/src/main/resources/people.txt")
  .map(_.split(","))
  .map(attributes => Person(attributes(0), attributes(1).trim.toInt)) .toDF()

// Register the DataFrame as a temporary view peopleDF.createOrReplaceTempView("people")

// SQL statements can be run by using the sql methods provided by Spark

val teenagersDF = spark.sql("SELECT name, age FROM people WHERE age BETWEEN 13 AND 19")

// The columns of a row in the result can be accessed by field index teenagersDF.map(teenager =>
"Name: " + teenager(0)).show()

// +------------+

// |       value|

// +------------+

// |Name: Justin|

// +------------+

// or by field name teenagersDF.map(teenager => "Name: " +
teenager.getAs[String]("name")).show()

// +------------+

// |       value|

// +------------+

// |Name: Justin|

// +------------+

// No pre-defined encoders for Dataset[Map[K,V]], define explicitly

implicit val mapEncoder = org.apache.spark.sql.Encoders.kryo[Map[String, Any]]

// Primitive types and case classes can be also defined as

// implicit val stringIntMapEncoder: Encoder[Map[String, Any]] = ExpressionEncoder()

// row.getValuesMap[T] retrieves multiple columns at once into a Map[String, T]

teenagersDF.map(teenager => teenager.getValuesMap[Any](List("name", "age"))).collect()

// Array(Map("name" -> "Justin", "age" -> 19))
```



```
scala> case class Person(name: String, age: Long)
defined class Person

scala> val peopleDF = spark.sparkContext.textFile("C:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.t
xt").map(_.split(",")).map(attributes => Person(attributes(0), attributes(1).trim.toInt)).toDF()
peopleDF: org.apache.spark.sql.DataFrame = [name: string, age: bigint]

scala> peopleDF.createOrReplaceTempView("people")

scala> val teenagersDF = spark.sql("SELECT name, age FROM people WHERE age BETWEEN 13 AND 19")
teenagersDF: org.apache.spark.sql.DataFrame = [name: string, age: bigint]

scala> teenagersDF.map(teenager => "Name: " + teenager(0)).show()
+------------+
|       value|
+------------+
|Name: Justin|
+------------+
```



```
scala> teenagersDF.map(teenager => "Name: " + teenager.getAs[String]("name")).show()
+------------+
|       value|
+------------+
|Name: Justin|
+------------+


scala> implicit val mapEncoder = org.apache.spark.sql.Encoders.kryo[Map[String, Any]
     |
     | ]
mapEncoder: org.apache.spark.sql.Encoder[Map[String,Any]] = class[value[0]: binary]

scala> teenagersDF.map(teenager => teenager.getValuesMap[Any](List("name", "age"))).collect()
res3: Array[Map[String,Any]] = Array(Map(name -> Justin, age -> 19))
```

Programmatically Specifying the Schema

When case classes cannot be defined ahead of time (for example, the structure of records is encoded in a string, or a text dataset will be parsed and fields will be projected differently for different users), a DataFrame can be created programmatically with three steps.

Create an RDD of Rows from the original RDD;

Create the schema represented by a StructType matching the structure of Rows in the RDD created in Step 1.

Apply the schema to the RDD of Rows via createDataFrame method provided by SparkSession.

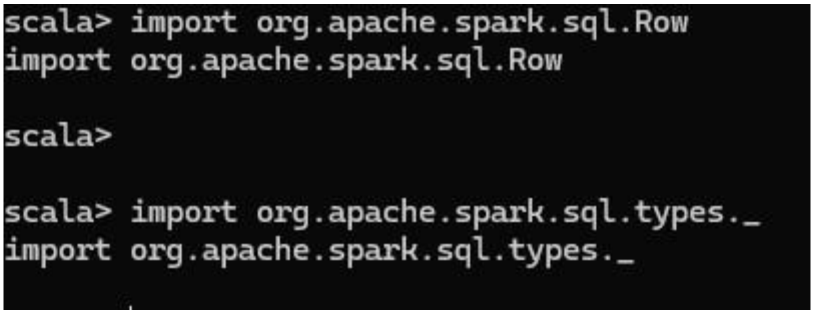
import org.apache.spark.sql.Row import org.apache.spark.sql.types._

// Create an RDD

val peopleRDD = spark.sparkContext.textFile("examples/src/main/resources/people.txt")

// The schema is encoded in a string val
schemaString = "name age"

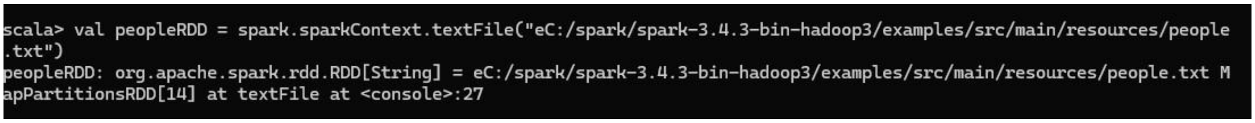
```
// Generate the schema based on the string of schema val
fields = schemaString.split(" ")

  .map(fieldName => StructField(fieldName, StringType, nullable = true))



schema = StructType(fields)

// Convert records of the RDD (people) to Rows val
rowRDD = peopleRDD

  .map(_.split(","))

  .map(attributes => Row(attributes(0), attributes(1).trim))

// Apply the schema to the RDD val peopleDF =
spark.createDataFrame(rowRDD, schema)

// Creates a temporary view using the DataFrame peopleDF.createOrReplaceTempView("people")

// SQL can be run over a temporary view created using DataFrames val
results = spark.sql("SELECT name FROM people")

// The results of SQL queries are DataFrames and support all the normal RDD operations // The
columns of a row in the result can be accessed by field index or by field name
results.map(attributes => "Name: " + attributes(0)).show()

// +-------------+

// |        value|

// +-------------+

// |Name: Michael|

// |   Name: Andy|

// | Name: Justin|

// +-------------+

import org.apache.spark.sql.Row
Import org.apache.spark.sql.types._
```



```
scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala>

scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._
```

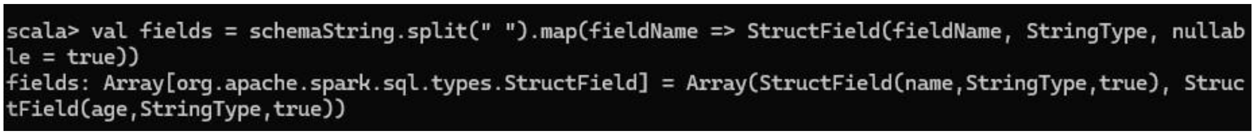
val peopleRDD =

spark.sparkContext.textFile("C:/spark/spark-3.4.3-bin/hadoop/examples/src/main/resources/people.txt")



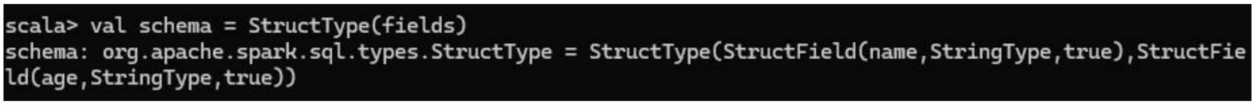
```
scala> val peopleRDD = spark.sparkContext.textFile("eC:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people
.txt")
peopleRDD: org.apache.spark.rdd.RDD[String] = eC:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.txt M
apPartitionsRDD[14] at textFile at <console>:27
```

fields = schemaString.split(" ").map(fieldName => StructField(fieldName, StringType, nullable = true))



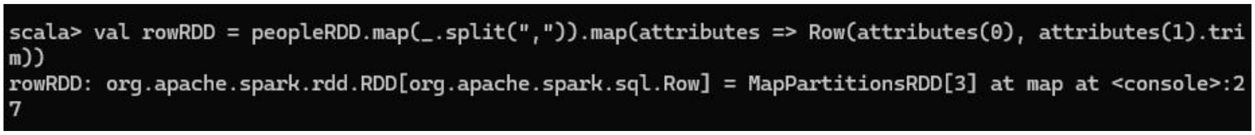
```
scala> val fields = schemaString.split(" ").map(fieldName => StructField(fieldName, StringType, nullab
le = true))
fields: Array[org.apache.spark.sql.types.StructField] = Array(StructField(name,StringType,true), Struc
tField(age,StringType,true))
```

val schema = StructType(fields)



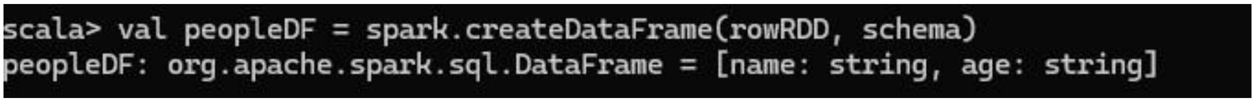
```
scala> val schema = StructType(fields)
schema: org.apache.spark.sql.types.StructType = StructType(StructField(name,StringType,true),StructFie
ld(age,StringType,true))
```

val rowRDD = peopleRDD.map(_.split(",")).map(attributed => Row(attributes(0), attributes(1).trim))



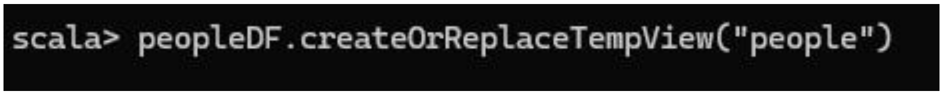
```
scala> val rowRDD = peopleRDD.map(_.split(",")).map(attributes => Row(attributes(0), attributes(1).tri
m))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[3] at map at <console>:2
7
```

val peopleDF = spark.createDataFrame(rowRDD, schema)



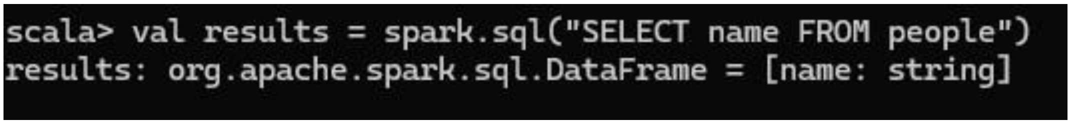
```
scala> val peopleDF = spark.createDataFrame(rowRDD, schema)
peopleDF: org.apache.spark.sql.DataFrame = [name: string, age: string]
```

peopleDF.createOrReplaceTempView("people")



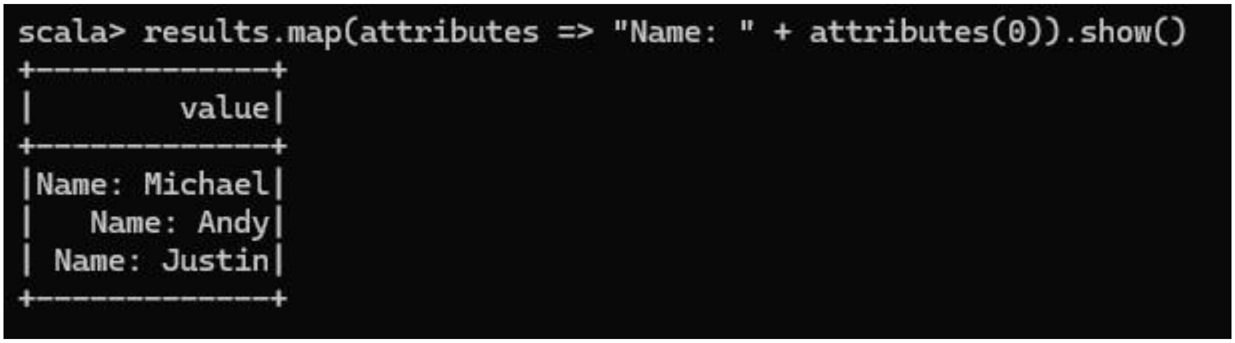
```
scala> peopleDF.createOrReplaceTempView("people")
```

val results = spark.sql("SELECT name FROM people")



```
scala> val results = spark.sql("SELECT name FROM people")
results: org.apache.spark.sql.DataFrame = [name: string]
```

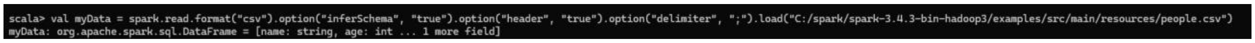
results.map(attributes => “Name: “ + attributes(0)).show()



```
scala> results.map(attributes => "Name: " + attributes(0)).show()
+-------------+
|        value|
+-------------+
|Name: Michael|
|   Name: Andy|
| Name: Justin|
+-------------+
```

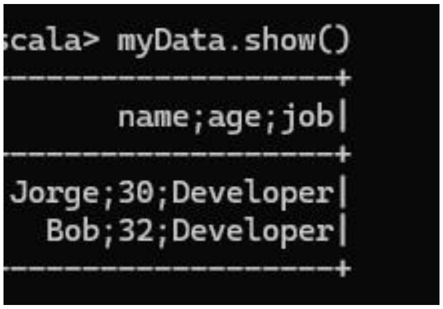
Basic Operations with csv file

myData = spark.read.format(“csv”).option(“inferSchema”,”true”).option(“header”,”true”).option(“delimeter”,”:
”).load(“C:/spark/spark-3.4.3-hadoop3/examples/src/main/resources/people.csv”)



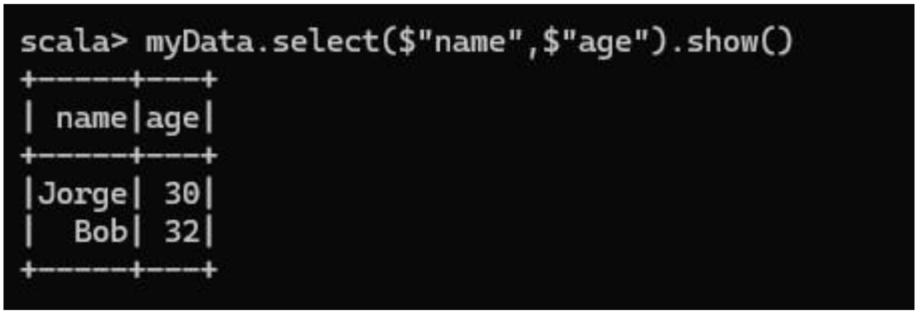
```
scala> val myData = spark.read.format("csv").option("inferSchema", "true").option("header", "true").option("delimiter", ";").load("C:/spark/spark-3.4.3-bin-hadoop3/examples/src/main/resources/people.csv")
myData: org.apache.spark.sql.DataFrame = [name: string, age: int ... 1 more field]
```

myData.show()



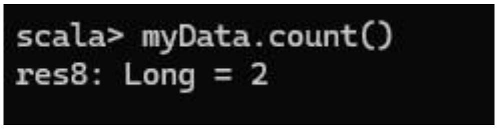
```
scala> myData.show()
------------------+
      name;age;job|
------------------+
Jorge;30;Developer|
  Bob;32;Developer|
------------------+
```

myData.select($”name”,”$age”).show()



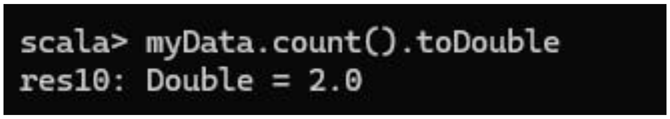
```
scala> myData.select($"name",$"age").show()
+-----+---+
| name|age|
+-----+---+
|Jorge| 30|
|  Bob| 32|
+-----+---+
```

myData.count()



```
scala> myData.count()
res8: Long = 2
```

myData.count().toDouble

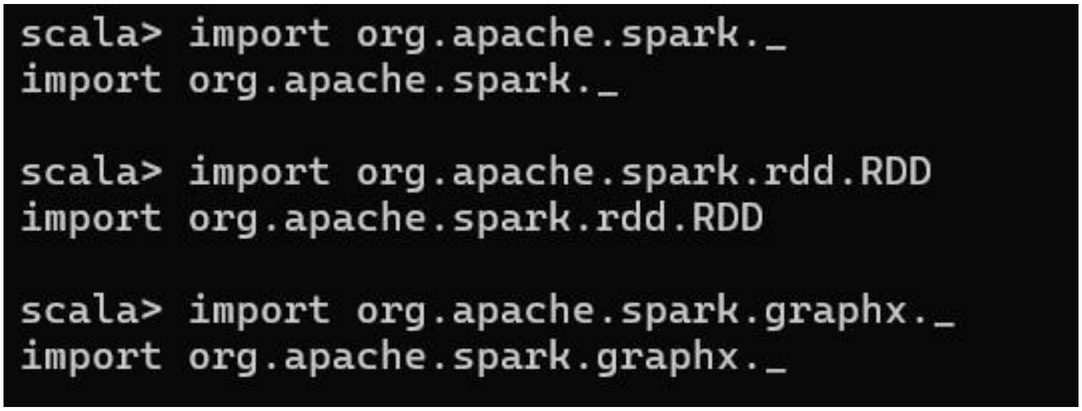


```
scala> myData.count().toDouble
res10: Double = 2.0
```

Practical 03

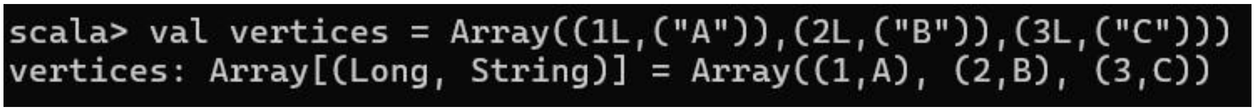
Aim : GraphX

import
org.apache.spark._
import
org.apache.spark.rdd.RDD
import
org.apache.spark.graphx.
_



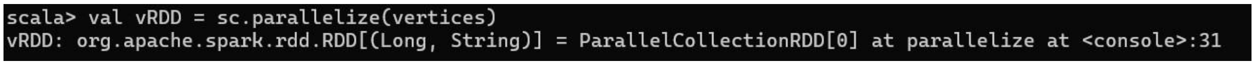
```
scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD

scala> import org.apache.spark.graphx._
import org.apache.spark.graphx._
```

val vertices = Array((1L,("A")),(2L,("B")),(3L,("C")))



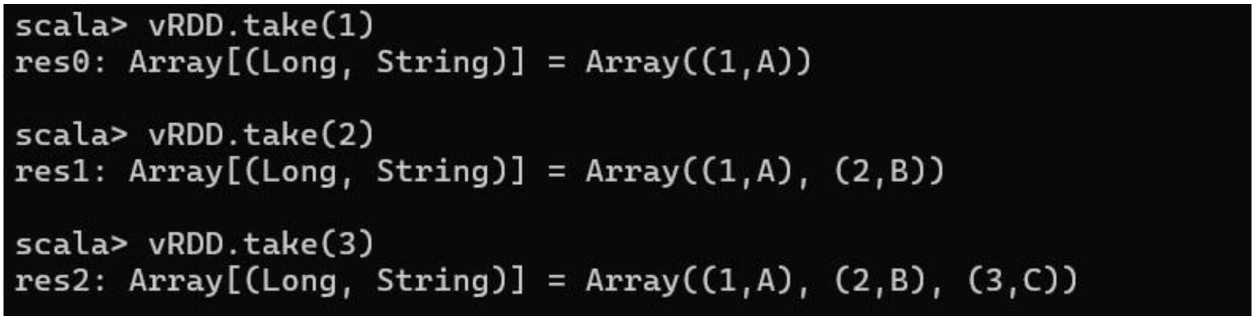
```
scala> val vertices = Array((1L,("A")),(2L,("B")),(3L,("C")))
vertices: Array[(Long, String)] = Array((1,A), (2,B), (3,C))
```

val vRDD = sc.parallelize(vertices)



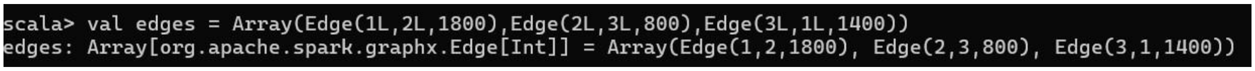
```
scala> val vRDD = sc.parallelize(vertices)
vRDD: org.apache.spark.rdd.RDD[(Long, String)] = ParallelCollectionRDD[0] at parallelize at <console>:31
```

vRDD.take(1)
vRDD.take(2)
vRDD.take(3)



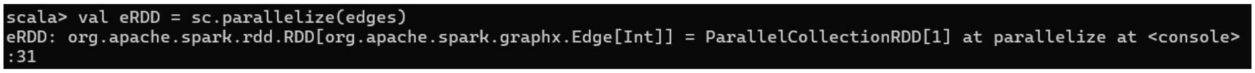
```
scala> vRDD.take(1)
res0: Array[(Long, String)] = Array((1,A))

scala> vRDD.take(2)
res1: Array[(Long, String)] = Array((1,A), (2,B))

scala> vRDD.take(3)
res2: Array[(Long, String)] = Array((1,A), (2,B), (3,C))
```

val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,1L,1400))



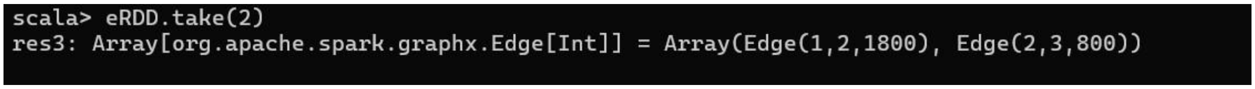
```
scala> val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,1L,1400))
edges: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,2,1800), Edge(2,3,800), Edge(3,1,1400))
```

val eRDD = sc.parallelize(edges)



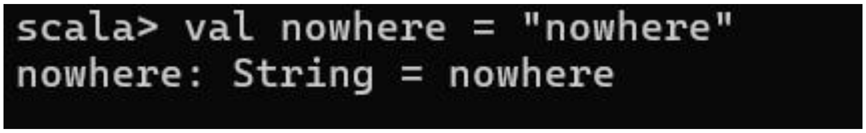
```
scala> val eRDD = sc.parallelize(edges)
eRDD: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = ParallelCollectionRDD[1] at parallelize at <console>
:31
```

eRDD.take(2)



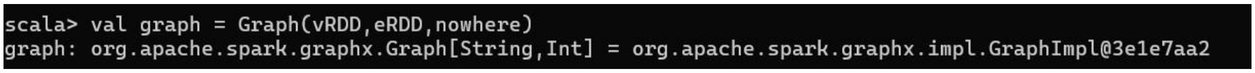
```
scala> eRDD.take(2)
res3: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,2,1800), Edge(2,3,800))
```

val nowhere = "nowhere"



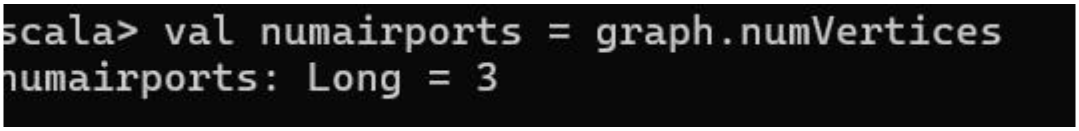
```
scala> val nowhere = "nowhere"
nowhere: String = nowhere
```

val graph = Graph(vRDD,eRDD,nowhere)



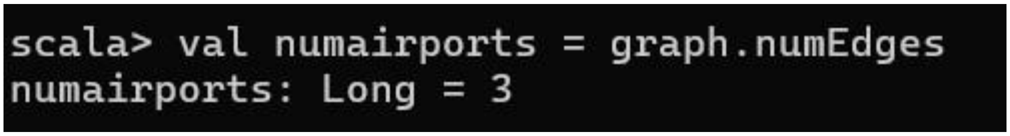
```
scala> val graph = Graph(vRDD,eRDD,nowhere)
graph: org.apache.spark.graphx.Graph[String,Int] = org.apache.spark.graphx.impl.GraphImpl@3e1e7aa2
```

#To check number of Airports
val numairports =
graph.numVertices



```
scala> val numairports = graph.numVertices
numairports: Long = 3
```

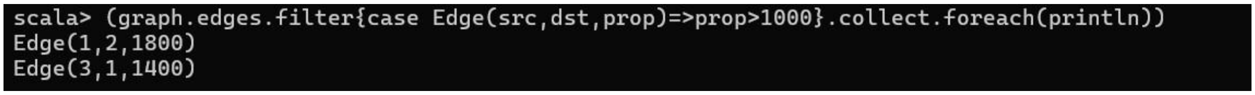
#To check routes val
numairports =
graph.numEdges



```
scala> val numairports = graph.numEdges
numairports: Long = 3
```

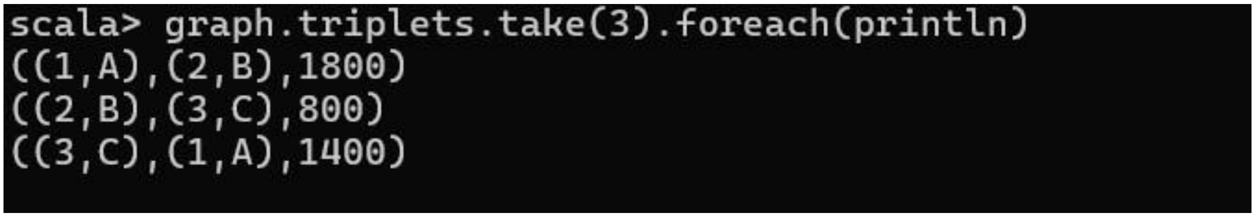
#Route having distance > 1000

(graph.edges.filter{case Edge(src,dst,prop)=>prop>1000}.collect.foreach(println))



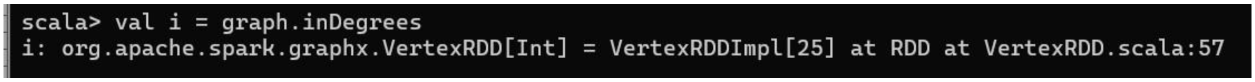
```
scala> (graph.edges.filter{case Edge(src,dst,prop)=>prop>1000}.collect.foreach(println))
Edge(1,2,1800)
Edge(3,1,1400)
```

#Triplet Information
graph.triplets.take(3).foreach(println)



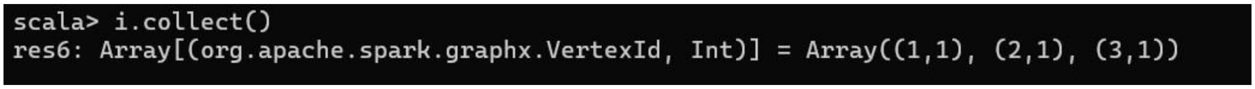
```
scala> graph.triplets.take(3).foreach(println)
((1,A),(2,B),1800)
((2,B),(3,C),800)
((3,C),(1,A),1400)
```

#Indegree val i =
graph.inDegrees



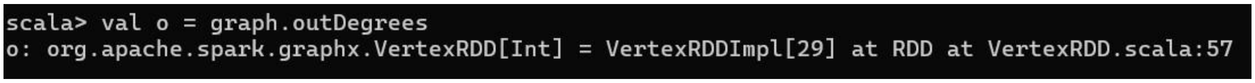
```
scala> val i = graph.inDegrees
i: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[25] at RDD at VertexRDD.scala:57
```

i.collect()



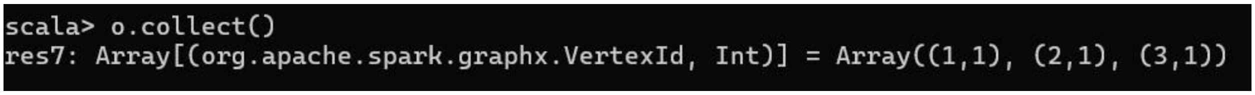
```
scala> i.collect()
res6: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,1), (2,1), (3,1))
```

#Outdegrees val o =
graph.outDegrees



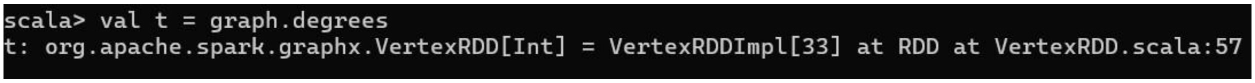
```
scala> val o = graph.outDegrees
o: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[29] at RDD at VertexRDD.scala:57
```

o.collect()



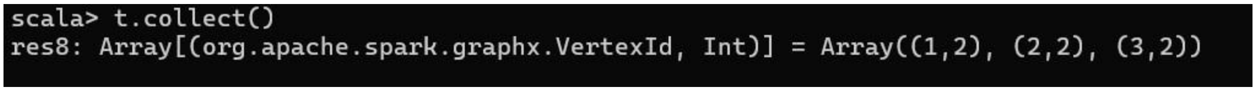
```
scala> o.collect()
res7: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,1), (2,1), (3,1))
```

#Total Degree val
t = graph.degrees



```
scala> val t = graph.degrees
t: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[33] at RDD at VertexRDD.scala:57
```

t.collect()



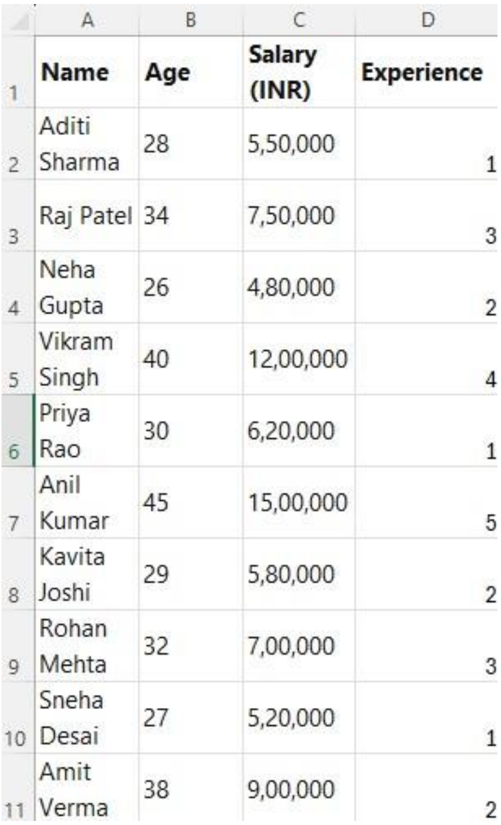
```
scala> t.collect()
res8: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,2), (2,2), (3,2))
```

Practical 04

Aim : PySpark

Steps :

1. Make a CSV file with data related to Name Age Salary and Experience.



	A	B	C	D
1	Name	Age	Salary (INR)	Experience
2	Aditi Sharma	28	5,50,000	1
3	Raj Patel	34	7,50,000	3
4	Neha Gupta	26	4,80,000	2
5	Vikram Singh	40	12,00,000	4
6	Priya Rao	30	6,20,000	1
7	Anil Kumar	45	15,00,000	5
8	Kavita Joshi	29	5,80,000	2
9	Rohan Mehta	32	7,00,000	3
10	Sneha Desai	27	5,20,000	1
11	Amit Verma	38	9,00,000	2

2. Code

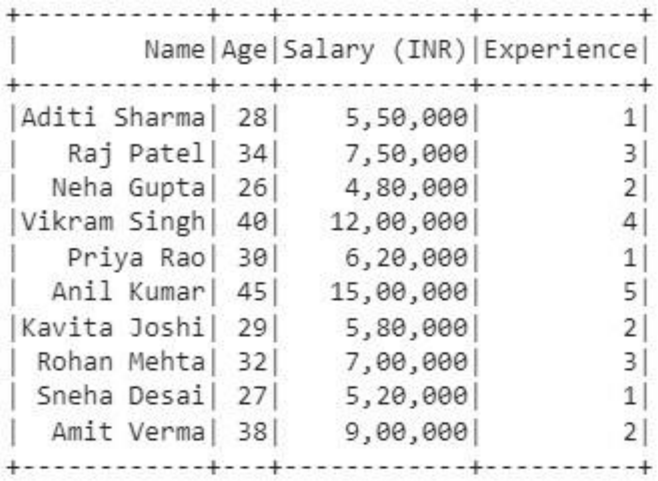
```
from pyspark.sql import SparkSession

# Create a Spark session spark =
SparkSession.builder.appName("Read CSV").getOrCreate()

# Path to your CSV file
csv_file_path = "Student.csv"
```

# Read the CSV file into a DataFrame. inferSchema tries to determine the datatype of values in the fields.

```
df = spark.read.csv(csv_file_path, header=True,
                                    inferSchema=True)
```

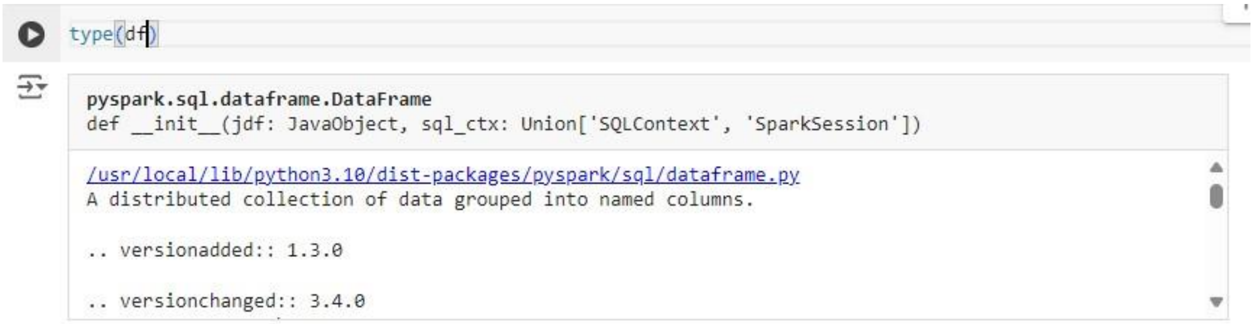
# Display the DataFrame
df.show()

Output



```
+------------+---+-----------+----------+
|        Name|Age|Salary (INR)|Experience|
+------------+---+-----------+----------+
|Aditi Sharma| 28|   5,50,000|         1|
|   Raj Patel| 34|   7,50,000|         3|
|  Neha Gupta| 26|   4,80,000|         2|
|Vikram Singh| 40|  12,00,000|         4|
|   Priya Rao| 30|   6,20,000|         1|
|  Anil Kumar| 45|  15,00,000|         5|
|Kavita Joshi| 29|   5,80,000|         2|
| Rohan Mehta| 32|   7,00,000|         3|
| Sneha Desai| 27|   5,20,000|         1|
|  Amit Verma| 38|   9,00,000|         2|
+------------+---+-----------+----------+
```

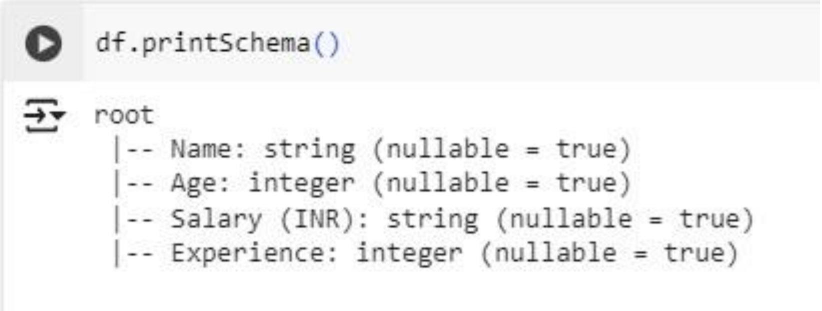
type(df)



```
type(df)
```

```
pyspark.sql.dataframe.DataFrame
def __init__(jdf: JavaObject, sql_ctx: Union['SQLContext', 'SparkSession'])

/usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py
A distributed collection of data grouped into named columns.

.. versionadded:: 1.3.0

.. versionchanged:: 3.4.0
```

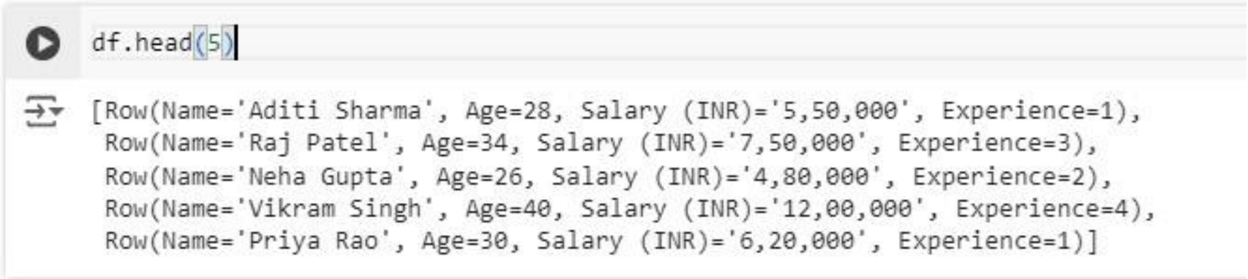
df.printSchema()



```
df.printSchema()
```

```
root
 |-- Name: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Salary (INR): string (nullable = true)
 |-- Experience: integer (nullable = true)
```

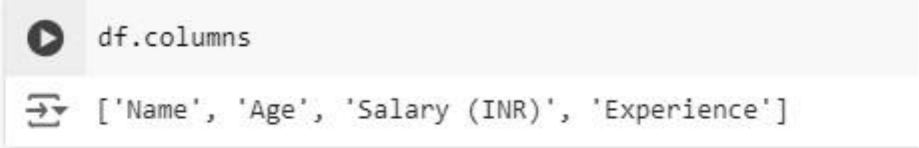
df.head(5) or df.show(5)



```
df.head(5)
```

```
[Row(Name='Aditi Sharma', Age=28, Salary (INR)='5,50,000', Experience=1),
 Row(Name='Raj Patel', Age=34, Salary (INR)='7,50,000', Experience=3),
 Row(Name='Neha Gupta', Age=26, Salary (INR)='4,80,000', Experience=2),
 Row(Name='Vikram Singh', Age=40, Salary (INR)='12,00,000', Experience=4),
 Row(Name='Priya Rao', Age=30, Salary (INR)='6,20,000', Experience=1)]
```

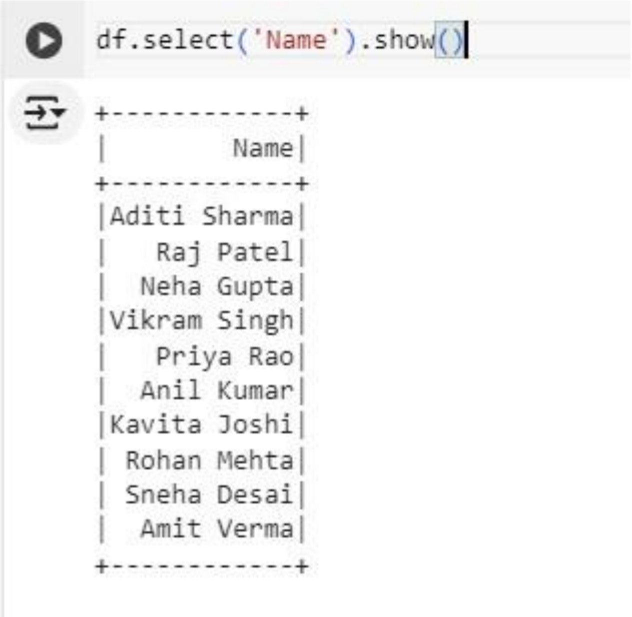
df.columns



```
df.columns
```

```
['Name', 'Age', 'Salary (INR)', 'Experience']
```

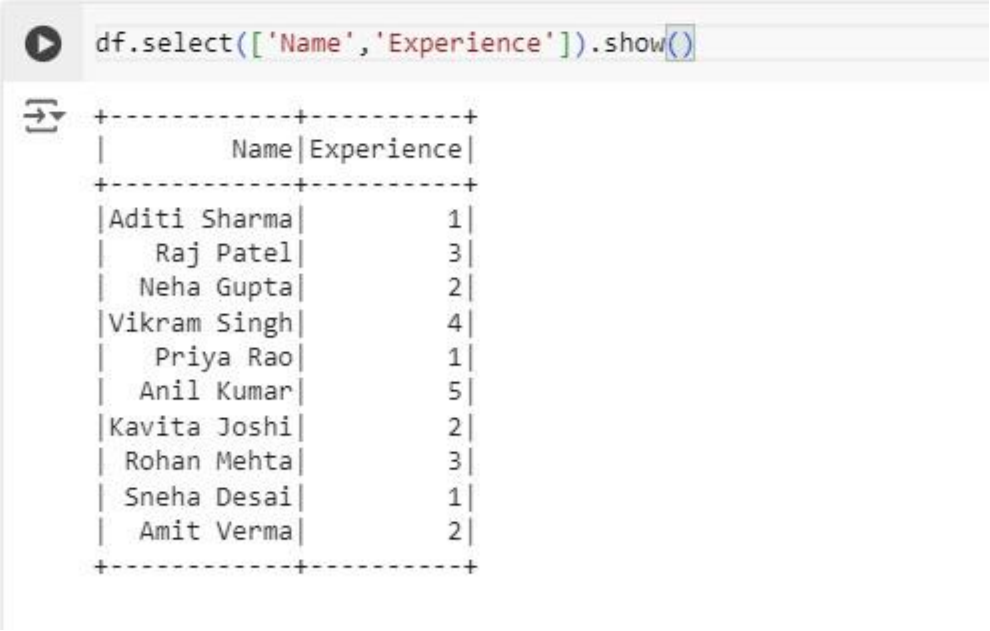
df.select('Name').show()



```
df.select('Name').show()
```

```
+------------+
|        Name|
+------------+
|Aditi Sharma|
|   Raj Patel|
|  Neha Gupta|
|Vikram Singh|
|   Priya Rao|
|  Anil Kumar|
|Kavita Joshi|
| Rohan Mehta|
| Sneha Desai|
|  Amit Verma|
+------------+
```

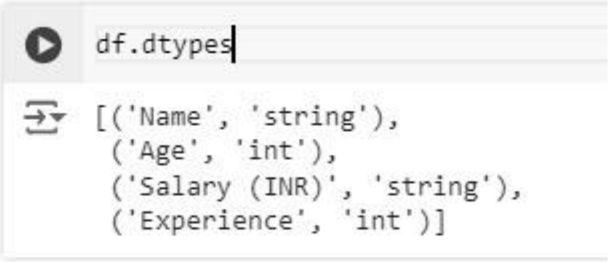
df.select(['Name','Experience']).show()



```
df.select(['Name','Experience']).show()
```

```
+-----------+----------+
|       Name|Experience|
+-----------+----------+
|Aditi Sharma|         1|
|   Raj Patel|         3|
|  Neha Gupta|         2|
|Vikram Singh|         4|
|   Priya Rao|         1|
|  Anil Kumar|         5|
|Kavita Joshi|         2|
| Rohan Mehta|         3|
| Sneha Desai|         1|
|  Amit Verma|         2|
+-----------+----------+
```

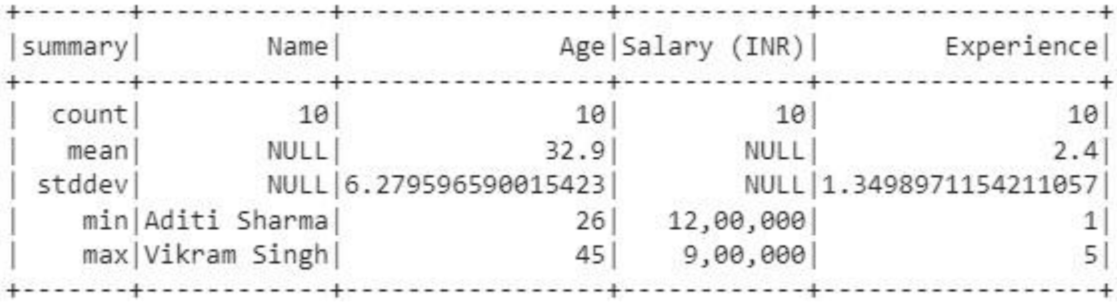
df.dtypes



```
df.dtypes
```

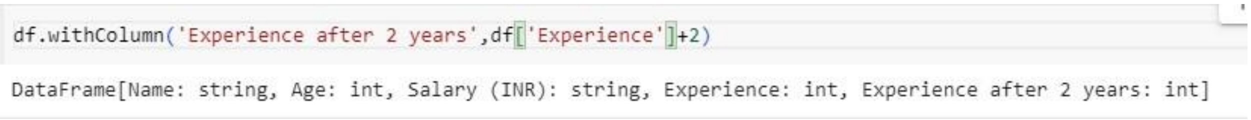
```
[('Name', 'string'),
 ('Age', 'int'),
 ('Salary (INR)', 'string'),
 ('Experience', 'int')]
```

df.describe().show()



```
+-------+-----------+-----------------+------------+------------------+
|summary|       Name|              Age|Salary (INR)|        Experience|
+-------+-----------+-----------------+------------+------------------+
|  count|         10|               10|          10|                10|
|   mean|       NULL|             32.9|        NULL|               2.4|
| stddev|       NULL|6.279596590015423|        NULL|1.3498971154211057|
|    min|Aditi Sharma|              26|   12,00,000|                 1|
|    max|Vikram Singh|              45|    9,00,000|                 5|
+-------+-----------+-----------------+------------+------------------+
```

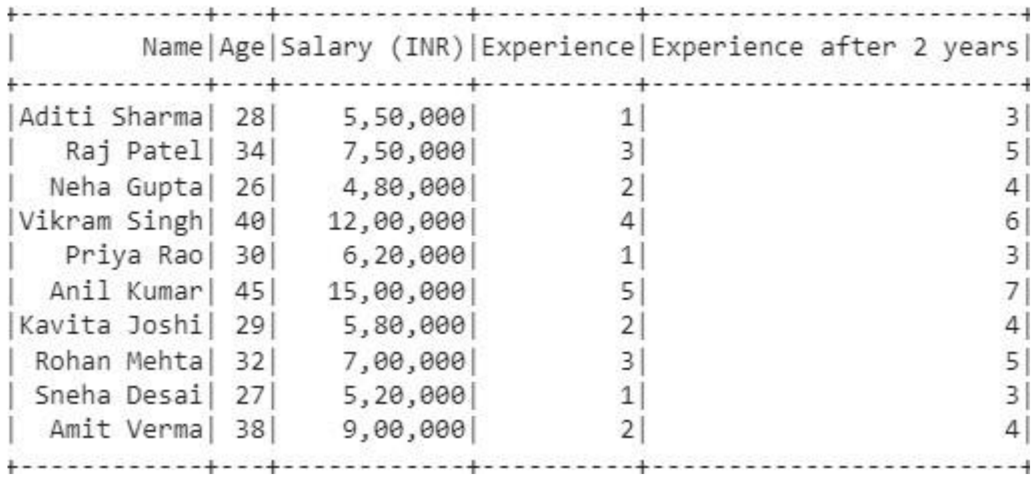
Adding columns to the dataframe df = df.withColumn('Experience after 2 years',df['Experience']+2)



```
df.withColumn('Experience after 2 years',df['Experience']+2)
```

```
DataFrame[Name: string, Age: int, Salary (INR): string, Experience: int, Experience after 2 years: int]
```

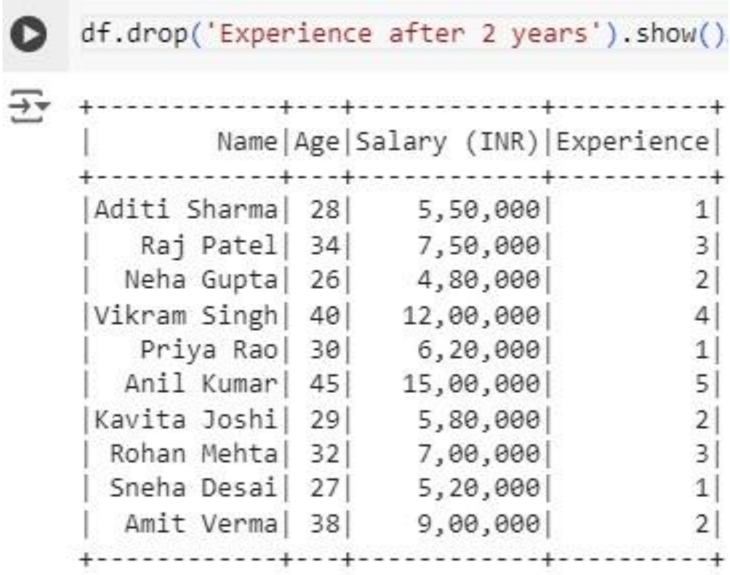
df.show()



```
+------------+---+------------+----------+------------------------+
|        Name|Age|Salary (INR)|Experience|Experience after 2 years|
+------------+---+------------+----------+------------------------+
|Aditi Sharma| 28|    5,50,000|         1|                       3|
|   Raj Patel| 34|    7,50,000|         3|                       5|
|  Neha Gupta| 26|    4,80,000|         2|                       4|
|Vikram Singh| 40|   12,00,000|         4|                       6|
|   Priya Rao| 30|    6,20,000|         1|                       3|
|  Anil Kumar| 45|   15,00,000|         5|                       7|
|Kavita Joshi| 29|    5,80,000|         2|                       4|
| Rohan Mehta| 32|    7,00,000|         3|                       5|
| Sneha Desai| 27|    5,20,000|         1|                       3|
|  Amit Verma| 38|    9,00,000|         2|                       4|
+------------+---+------------+----------+------------------------+
```

Dropping columns from dataframe

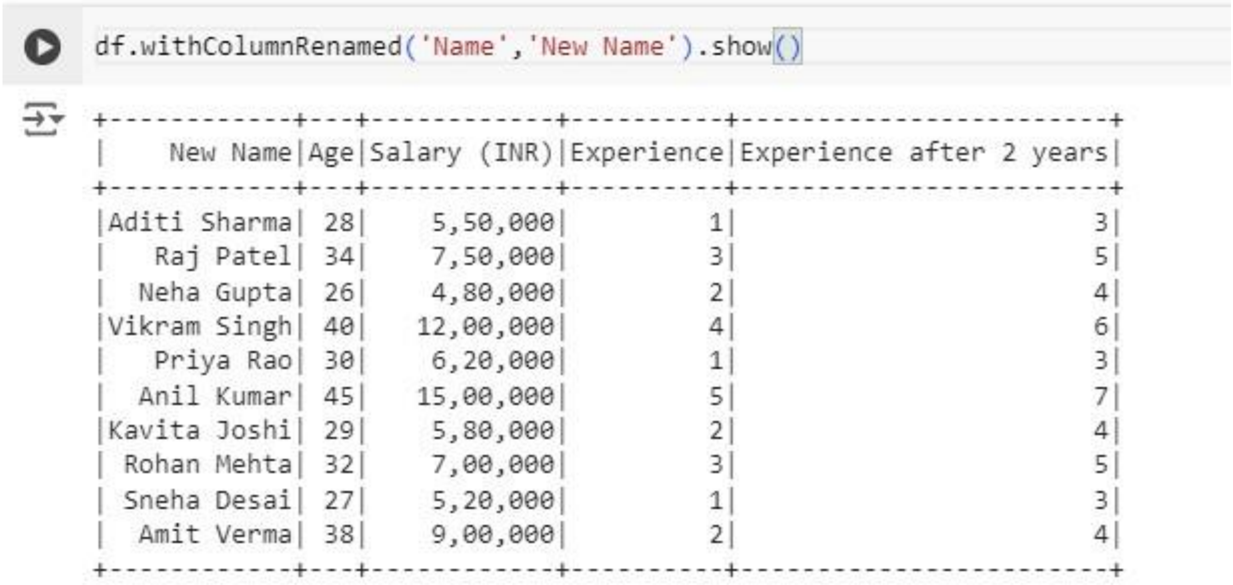
df.drop('Experience after 2 years').show()



```
df.drop('Experience after 2 years').show()
```

```
+------------+---+------------+----------+
|        Name|Age|Salary (INR)|Experience|
+------------+---+------------+----------+
|Aditi Sharma| 28|    5,50,000|         1|
|   Raj Patel| 34|    7,50,000|         3|
|  Neha Gupta| 26|    4,80,000|         2|
|Vikram Singh| 40|   12,00,000|         4|
|   Priya Rao| 30|    6,20,000|         1|
|  Anil Kumar| 45|   15,00,000|         5|
|Kavita Joshi| 29|    5,80,000|         2|
| Rohan Mehta| 32|    7,00,000|         3|
| Sneha Desai| 27|    5,20,000|         1|
|  Amit Verma| 38|    9,00,000|         2|
+------------+---+------------+----------+
```

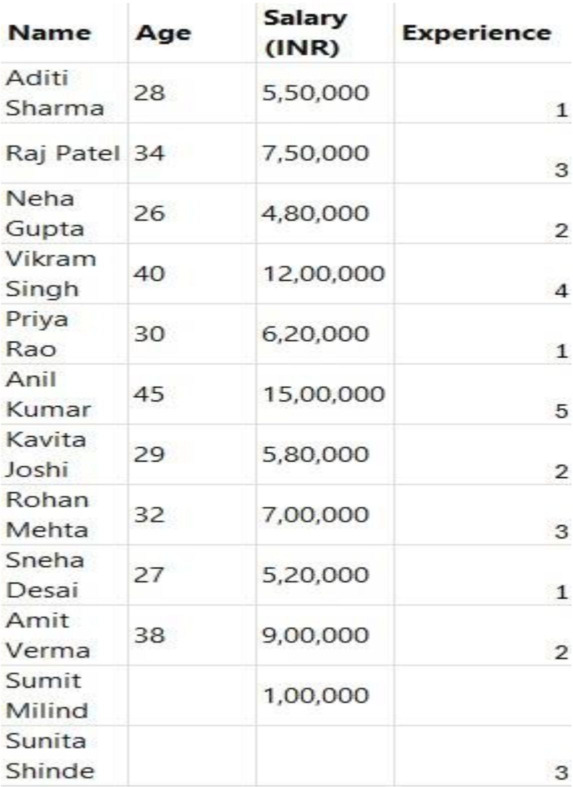
df.withColumnRenamed('Name','New Name').show()



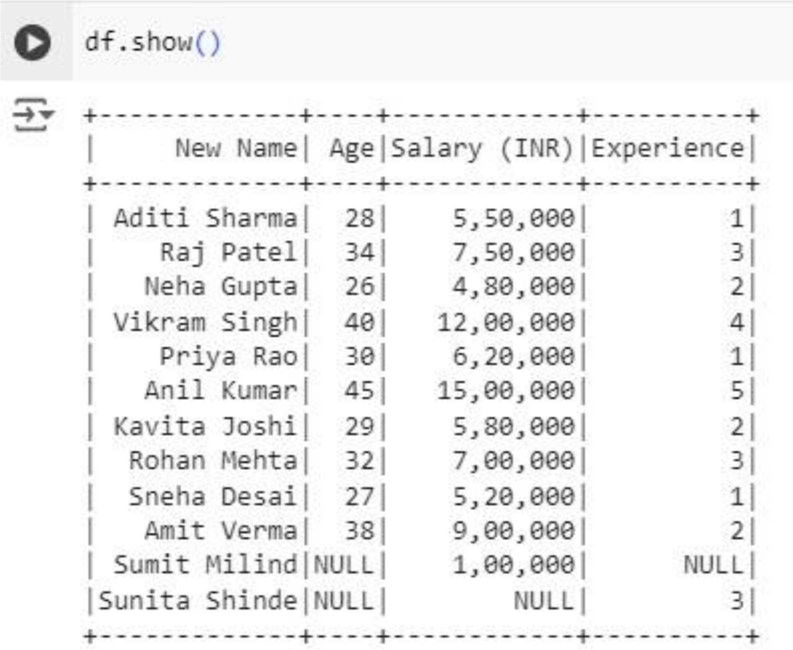
```
df.withColumnRenamed('Name','New Name').show()
```

```
+------------+---+------------+----------+-------------------------+
|    New Name|Age|Salary (INR)|Experience|Experience after 2 years|
+------------+---+------------+----------+-------------------------+
|Aditi Sharma| 28|    5,50,000|         1|                        3|
|   Raj Patel| 34|    7,50,000|         3|                        5|
|  Neha Gupta| 26|    4,80,000|         2|                        4|
|Vikram Singh| 40|   12,00,000|         4|                        6|
|   Priya Rao| 30|    6,20,000|         1|                        3|
|  Anil Kumar| 45|   15,00,000|         5|                        7|
|Kavita Joshi| 29|    5,80,000|         2|                        4|
| Rohan Mehta| 32|    7,00,000|         3|                        5|
| Sneha Desai| 27|    5,20,000|         1|                        3|
|  Amit Verma| 38|    9,00,000|         2|                        4|
+------------+---+------------+----------+-------------------------+
```

With new data that has null values



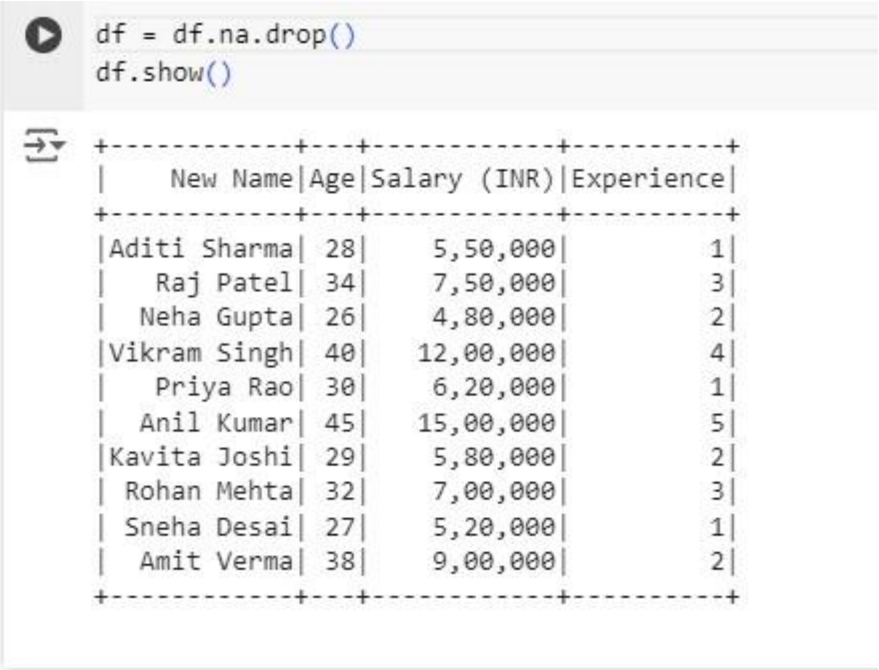
Name	Age	Salary (INR)	Experience
Aditi Sharma	28	5,50,000	1
Raj Patel	34	7,50,000	3
Neha Gupta	26	4,80,000	2
Vikram Singh	40	12,00,000	4
Priya Rao	30	6,20,000	1
Anil Kumar	45	15,00,000	5
Kavita Joshi	29	5,80,000	2
Rohan Mehta	32	7,00,000	3
Sneha Desai	27	5,20,000	1
Amit Verma	38	9,00,000	2
Sumit Milind		1,00,000	
Sunita Shinde			3



```
df.show()
```

```
+------------+----+-----------+----------+
|    New Name| Age|Salary (INR)|Experience|
+------------+----+-----------+----------+
| Aditi Sharma|  28|   5,50,000|         1|
|    Raj Patel|  34|   7,50,000|         3|
|   Neha Gupta|  26|   4,80,000|         2|
| Vikram Singh|  40|  12,00,000|         4|
|    Priya Rao|  30|   6,20,000|         1|
|   Anil Kumar|  45|  15,00,000|         5|
| Kavita Joshi|  29|   5,80,000|         2|
|  Rohan Mehta|  32|   7,00,000|         3|
|  Sneha Desai|  27|   5,20,000|         1|
|   Amit Verma|  38|   9,00,000|         2|
| Sumit Milind|NULL|   1,00,000|      NULL|
|Sunita Shinde|NULL|       NULL|         3|
+------------+----+-----------+----------+
```

df = df.na.drop() df.show()

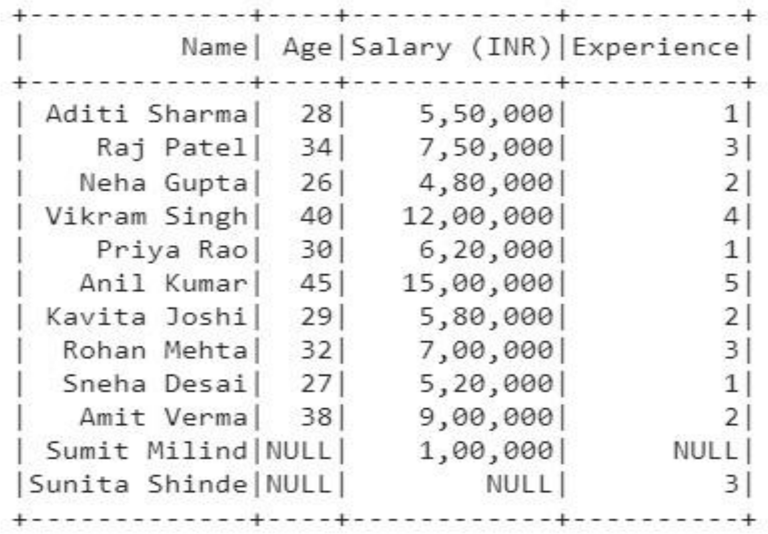


```
df = df.na.drop()
df.show()
```

```
+------------+---+------------+----------+
|    New Name|Age|Salary (INR)|Experience|
+------------+---+------------+----------+
|Aditi Sharma| 28|    5,50,000|         1|
|   Raj Patel| 34|    7,50,000|         3|
|  Neha Gupta| 26|    4,80,000|         2|
|Vikram Singh| 40|   12,00,000|         4|
|   Priya Rao| 30|    6,20,000|         1|
|  Anil Kumar| 45|   15,00,000|         5|
|Kavita Joshi| 29|    5,80,000|         2|
| Rohan Mehta| 32|    7,00,000|         3|
| Sneha Desai| 27|    5,20,000|         1|
|  Amit Verma| 38|    9,00,000|         2|
+------------+---+------------+----------+
```

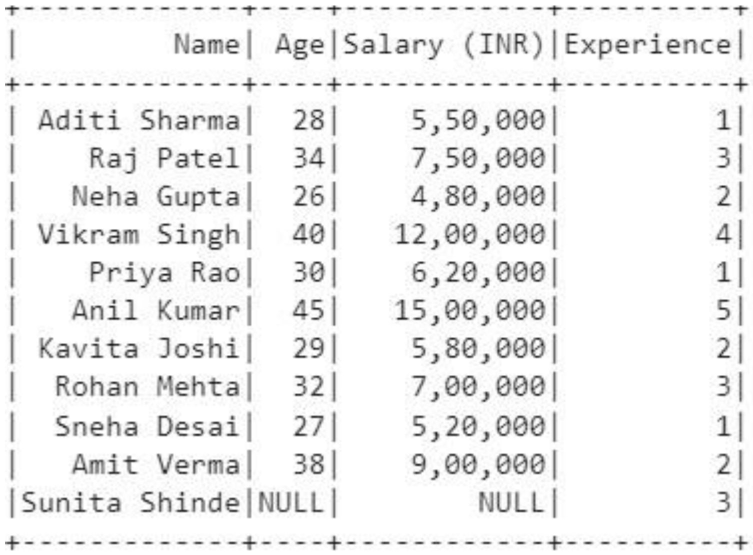
#Drops entries that have all columns as null df = df.na.drop(how="all") df.show()

Here none will be dropped

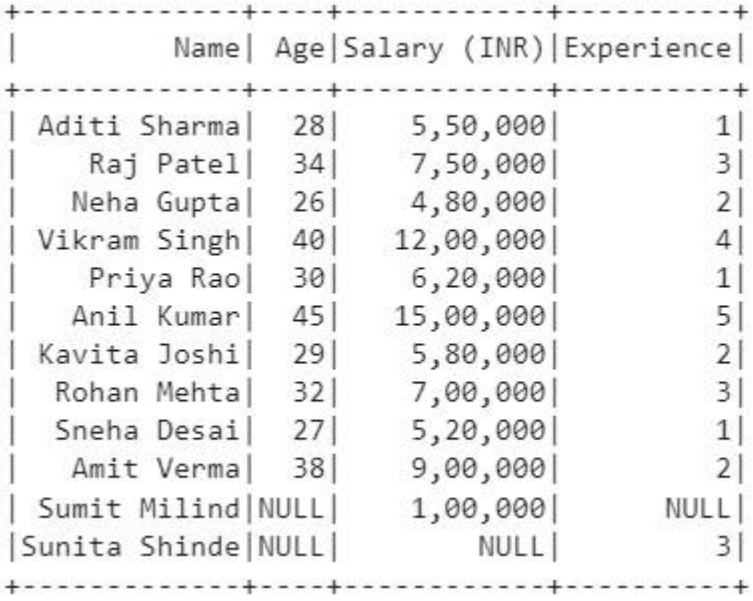


```
+-------------+----+------------+----------+
|         Name| Age|Salary (INR)|Experience|
+-------------+----+------------+----------+
| Aditi Sharma|  28|    5,50,000|         1|
|    Raj Patel|  34|    7,50,000|         3|
|   Neha Gupta|  26|    4,80,000|         2|
| Vikram Singh|  40|   12,00,000|         4|
|    Priya Rao|  30|    6,20,000|         1|
|   Anil Kumar|  45|   15,00,000|         5|
| Kavita Joshi|  29|    5,80,000|         2|
|  Rohan Mehta|  32|    7,00,000|         3|
|  Sneha Desai|  27|    5,20,000|         1|
|   Amit Verma|  38|    9,00,000|         2|
| Sumit Milind|NULL|    1,00,000|      NULL|
|Sunita Shinde|NULL|        NULL|         3|
+-------------+----+------------+----------+
```

#Only drop the rows that have given threshold number of NULL COLUMNS df = df.na.drop(how="any", thresh = 2) df.show()

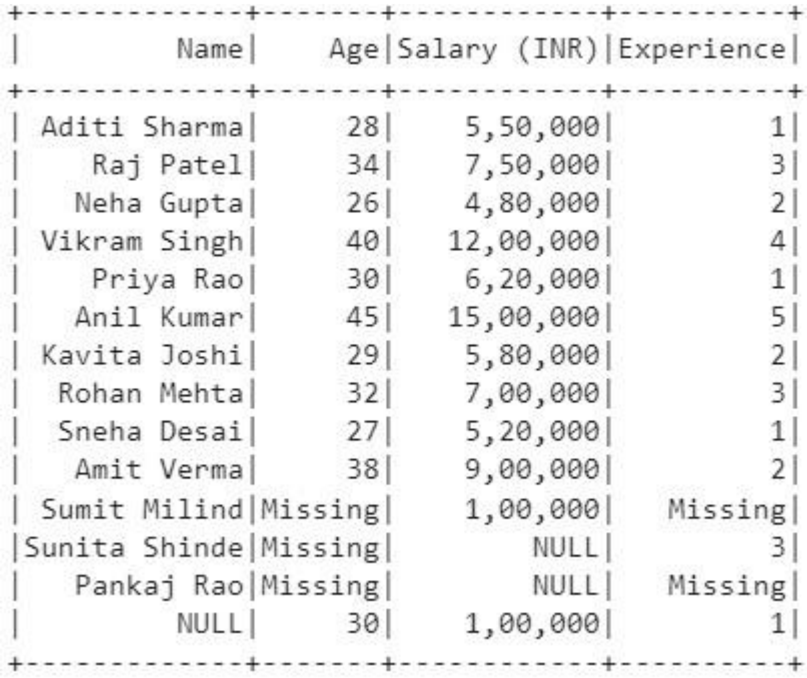


```
+------------+----+------------+----------+
|        Name| Age|Salary (INR)|Experience|
+------------+----+------------+----------+
|Aditi Sharma|  28|    5,50,000|         1|
|   Raj Patel|  34|    7,50,000|         3|
|  Neha Gupta|  26|    4,80,000|         2|
|Vikram Singh|  40|   12,00,000|         4|
|   Priya Rao|  30|    6,20,000|         1|
|  Anil Kumar|  45|   15,00,000|         5|
|Kavita Joshi|  29|    5,80,000|         2|
| Rohan Mehta|  32|    7,00,000|         3|
| Sneha Desai|  27|    5,20,000|         1|
|  Amit Verma|  38|    9,00,000|         2|
|Sunita Shinde|NULL|        NULL|         3|
+------------+----+------------+----------+
```



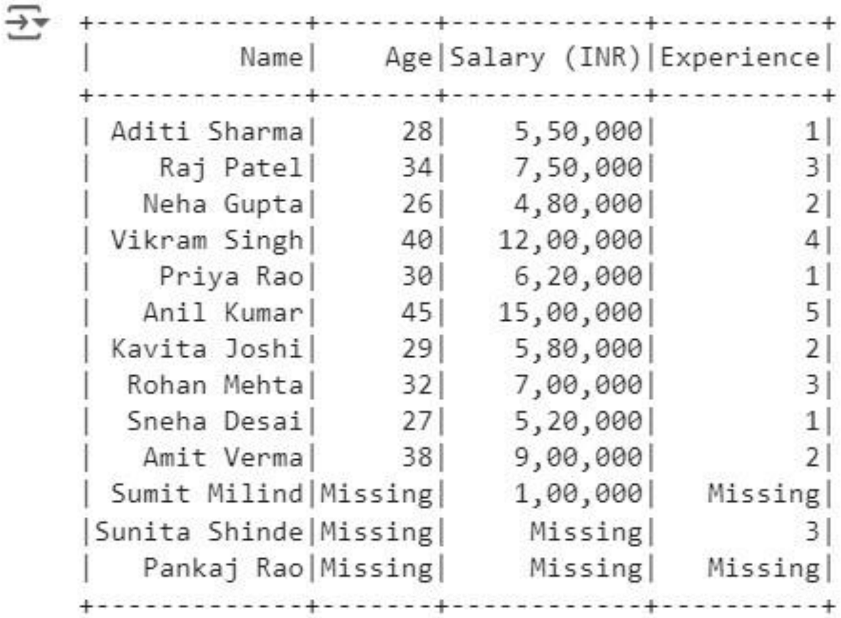
```
+------------+----+------------+----------+
|        Name| Age|Salary (INR)|Experience|
+------------+----+------------+----------+
|Aditi Sharma|  28|    5,50,000|         1|
|   Raj Patel|  34|    7,50,000|         3|
|  Neha Gupta|  26|    4,80,000|         2|
|Vikram Singh|  40|   12,00,000|         4|
|   Priya Rao|  30|    6,20,000|         1|
|  Anil Kumar|  45|   15,00,000|         5|
|Kavita Joshi|  29|    5,80,000|         2|
| Rohan Mehta|  32|    7,00,000|         3|
| Sneha Desai|  27|    5,20,000|         1|
|  Amit Verma|  38|    9,00,000|         2|
|Sumit Milind|NULL|    1,00,000|      NULL|
|Sunita Shinde|NULL|        NULL|         3|
+------------+----+------------+----------+
```

```
# Drops rows that has Experience as NULL df =
df.na.drop(how = "any",subset = ['Experience'])
df.show()

df = df.na.fill('Missing',['Age','Experience']).show()
```



```
+------------+-------+-----------+----------+
|        Name|    Age|Salary (INR)|Experience|
+------------+-------+-----------+----------+
| Aditi Sharma|     28|   5,50,000|         1|
|    Raj Patel|     34|   7,50,000|         3|
|   Neha Gupta|     26|   4,80,000|         2|
| Vikram Singh|     40|  12,00,000|         4|
|    Priya Rao|     30|   6,20,000|         1|
|   Anil Kumar|     45|  15,00,000|         5|
| Kavita Joshi|     29|   5,80,000|         2|
|  Rohan Mehta|     32|   7,00,000|         3|
|  Sneha Desai|     27|   5,20,000|         1|
|   Amit Verma|     38|   9,00,000|         2|
| Sumit Milind|Missing|   1,00,000|   Missing|
|Sunita Shinde|Missing|       NULL|         3|
|   Pankaj Rao|Missing|       NULL|   Missing|
|         NULL|     30|   1,00,000|         1|
+------------+-------+-----------+----------+
```

```
#Fills in the said text in the string columns that are NULL df =
df.na.fill("Missing") df.show()
```

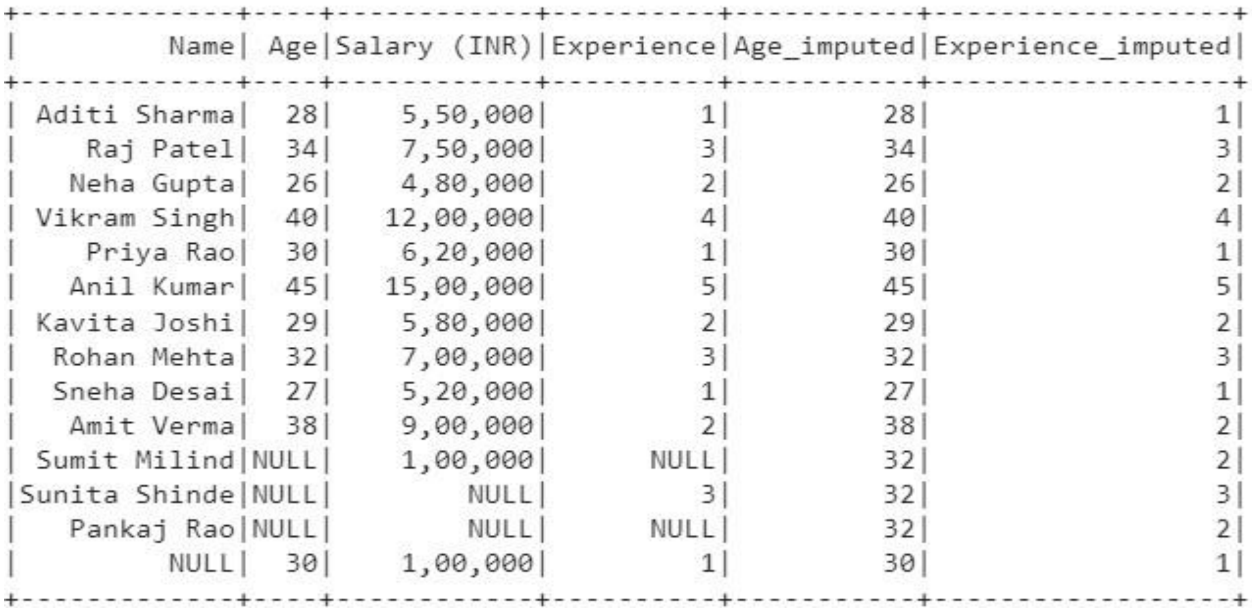


```
+------------+-------+-----------+----------+
|        Name|    Age|Salary (INR)|Experience|
+------------+-------+-----------+----------+
| Aditi Sharma|     28|   5,50,000|         1|
|    Raj Patel|     34|   7,50,000|         3|
|   Neha Gupta|     26|   4,80,000|         2|
| Vikram Singh|     40|  12,00,000|         4|
|    Priya Rao|     30|   6,20,000|         1|
|   Anil Kumar|     45|  15,00,000|         5|
| Kavita Joshi|     29|   5,80,000|         2|
|  Rohan Mehta|     32|   7,00,000|         3|
|  Sneha Desai|     27|   5,20,000|         1|
|   Amit Verma|     38|   9,00,000|         2|
| Sumit Milind|Missing|   1,00,000|   Missing|
|Sunita Shinde|Missing|    Missing|         3|
|   Pankaj Rao|Missing|    Missing|   Missing|
+------------+-------+-----------+----------+
```

```
from pyspark.ml.feature import Imputer imputer =
Imputer(

   inputCols = ['Age','Experience'], outputCols = ["{}_imputed".format(c) for c in
```

['Age','Experience']]).setStrategy("mean")

imputer.fit(df).transform(df).show()



```
+-------------+----+-----------+----------+-----------+------------------+
|         Name| Age|Salary (INR)|Experience|Age_imputed|Experience_imputed|
+-------------+----+-----------+----------+-----------+------------------+
| Aditi Sharma|  28|   5,50,000|         1|         28|                 1|
|    Raj Patel|  34|   7,50,000|         3|         34|                 3|
|   Neha Gupta|  26|   4,80,000|         2|         26|                 2|
| Vikram Singh|  40|  12,00,000|         4|         40|                 4|
|    Priya Rao|  30|   6,20,000|         1|         30|                 1|
|   Anil Kumar|  45|  15,00,000|         5|         45|                 5|
| Kavita Joshi|  29|   5,80,000|         2|         29|                 2|
|  Rohan Mehta|  32|   7,00,000|         3|         32|                 3|
|  Sneha Desai|  27|   5,20,000|         1|         27|                 1|
|   Amit Verma|  38|   9,00,000|         2|         38|                 2|
| Sumit Milind|NULL|   1,00,000|      NULL|         32|                 2|
|Sunita Shinde|NULL|       NULL|         3|         32|                 3|
|   Pankaj Rao|NULL|       NULL|      NULL|         32|                 2|
|         NULL|  30|   1,00,000|         1|         30|                 1|
+-------------+----+-----------+----------+-----------+------------------+
```