```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
!tar xf spark-3.1.1-bin-hadoop3.2.tgz
!pip install -q findspark
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
!ls
```

```
sample_data  spark-3.1.1-bin-hadoop3.2  spark-3.1.1-bin-hadoop3.2.tgz
```

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
# If we are using sparksession 1st time we have to use sparksession.builder
#sparksession allow all API ,spark context, sql context, hive context, spark stremming
#spark directly allow to use sql . for that create sql
```

```
#you can create multiple spark session with buildet method . but context only create once
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format output
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.1.1
Master
local[*]
AppName
pyspark-shell

```
!ls
```

```
sample_data  spark-3.1.1-bin-hadoop3.2  spark-3.1.1-bin-hadoop3.2.tgz
```

```
spark.stop()
```

```
#Creating spark context-Its like connecting to spark cluster
from pyspark import SparkConf
from pyspark.context import SparkContext
```

```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))
```

```
#Display details of sc
sc
```

SparkContext

Spark UI

Version
v3.1.1
Master
local[*]
AppName
pyspark-shell

```
#Check for prime numbers

def isprime(n):
    """
    check if integer n is a prime
    """
    # make sure n is a positive integer
    n = abs(int(n))
    # 0 and 1 are not primes
    if n < 2:
        return False
    # 2 is the only even prime number
    if n == 2:
        return True
    # all other even numbers are not primes
    if not n & 1:
        return False
    # range starts with 3 and only needs to go up the square root of n
    # for all odd numbers
    for x in range(3, int(n**0.5)+1, 2):
        if n % x == 0:
            return False
    return True


# Create an RDD of numbers from 0 to 1,000,000
nums = sc.parallelize(range(1000000))

# Compute the number of primes in the RDD
print(nums.filter(isprime).count())
```

```
78498
```

```
#create Sparksession
import pyspark
from pyspark.sql import SparkSession
spark1 = SparkSession.builder.master("local[1]").appName("sparksession1").getOrCreate()
#getorcreate() object create sparksession object
#local[1]-- to run satndalone can't be 0 -- the integer is always greater than 1
```

```
spark1
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.1.1
Master
local[1]
AppName
sparksession1

```
#Create new session
Spark2=SparkSession.newSession
Spark2
```

```
<function pyspark.sql.session.SparkSession.newSession>
```

```
#Get existing session
Spark3 =SparkSession.builder.getOrCreate
Spark3
```

```
<bound method SparkSession.Builder.getOrCreate of
<pyspark.sql.session.SparkSession.Builder object at 0x7fc4eac08910>>
```

```
#Example: Create RDD to load the contents of text file and do the text analysis

text = sc.textFile("/content/ex.txt")
text
```

```
/content/ex.txt MapPartitionsRDD[3] at textFile at NativeMethodAccessorImpl.java:0
```

```
from operator import add
def tokenize(text):
    return text.split()
```

```
words = text.flatMap(tokenize)
print(words)
```

```
PythonRDD[4] at RDD at PythonRDD.scala:53
```

```
wc = words.map(lambda x: (x,1))
wc
```

```
PythonRDD[5] at RDD at PythonRDD.scala:53
```

```
print(wc.toDebugString())
```

```
from operator import add
counts = wc.reduceByKey(add)
counts
counts.saveAsTextFile("wcc")
```

```
!ls wcc/
```

```
!head wcc/part-00000
```

```
#Date=12-07-22
from pyspark.sql import SparkSession, SQLContext
```

```
SparkSess = SparkSession.builder.master("local[1]").appName("mtApp").getOrCreate()
```

```
print(SparkSess.sparkContext)
print(SparkSess.sparkContext.appName)
```

```
<SparkContext master=local[1] appName=mtApp>
mtApp
```

```
SparkSess.sparkContext.stop()
```

```
#create a list of roll number and name
st =[(25,'Sarita'),(10,'Akshata'),(20,'Kajal'),(30,'Priti')]
```

```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))
sts = sc.parallelize(st)
```

```
print(sts)
```

```
ParallelCollectionRDD[10] at readRDDFromFile at PythonRDD.scala:274
```

```
columns =['Roll_No','Name']
```

```
df = SparkSess.createDataFrame(data=st,schema=columns)
```

```
df.show()
```

```
+-------+-------+
|Roll_No|   Name|
+-------+-------+
|     25| Sarita|
|     10|Akshata|
|     20|  Kajal|
|     30|  Priti|
```

```
+-------+-------+
```

```
df.collect()
```

```
[Row(Roll_No=25, Name='Sarita'),
 Row(Roll_No=10, Name='Akshata'),
 Row(Roll_No=20, Name='Kajal'),
 Row(Roll_No=30, Name='Priti')]
```

```
df.count()
```

```
4
```

```
df.first()
```

```
Row(Roll_No=25, Name='Sarita')
```

```
st1 =[{'Name':'Sarita','Roll_no':25},{'Name':'Akshata','Roll_no':10},{'Name':'Kajal','Roll
```

```
st1
```

```
[{'Name': 'Sarita', 'Roll_no': 25},
 {'Name': 'Akshata', 'Roll_no': 10},
 {'Name': 'Kajal', 'Roll_no': 20},
 {'Name': 'Priti', 'Roll_no': 30}]
```

```
df = SparkSess.createDataFrame(st1)
```

```
df.show()
```

```
+-------+-------+
|   Name|Roll_no|
+-------+-------+
| Sarita|     25|
|Akshata|     10|
|  Kajal|     20|
|  Priti|     30|
+-------+-------+
```

```
dfcsv = SparkSess.read.csv("/content/st1.csv")
```

```
dfcsv
```

```
DataFrame[_c0: string, _c1: string]
```

```
dfcsv.show()
```

```
+-------+-------+
|    _c0|    _c1|
+-------+-------+
|Roll_No|   Name|
|     25| Sarita|
|     10|Akshata|
|     20|  Kajal|
|     30|  Priti|
+-------+-------+
```

we can try this format-csv,avro,parquet,tsv,xml,text,json