```
#AIM: CREATING SPARK SESSION WITH DATA FRAME
#https://spark.apache.org/docs/latest/api/python/getting_started/quickstart_df.html
```

```
!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#for system use file we use os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
```

+ Code + Text

```
spark = SparkSession.builder.getOrCreate()
```

```
from datetime import datetime, date
import pandas as pd
from pyspark.sql import Row

df = spark.createDataFrame([
    Row(a=1, b=2., c='string1', d=date(2000, 1, 1), e=datetime(2000, 1, 1, 12, 0)),
    Row(a=2, b=3., c='string2', d=date(2000, 2, 1), e=datetime(2000, 1, 2, 12, 0)),
    Row(a=4, b=5., c='string3', d=date(2000, 3, 1), e=datetime(2000, 1, 3, 12, 0))
])
df
```

```
DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]
```

```
df = spark.createDataFrame([
    (1, 2., 'string1', date(2000, 1, 1), datetime(2000, 1, 1, 12, 0)),
    (2, 3., 'string2', date(2000, 2, 1), datetime(2000, 1, 2, 12, 0)),
    (3, 4., 'string3', date(2000, 3, 1), datetime(2000, 1, 3, 12, 0))
], schema='a long, b double, c string, d date, e timestamp')
df
```

```
DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]
```

```
pandas_df = pd.DataFrame({
    'a': [1, 2, 3],
    'b': [2., 3., 4.],
```

```
    'c': ['string1', 'string2', 'string3'],
    'd': [date(2000, 1, 1), date(2000, 2, 1), date(2000, 3, 1)],
    'e': [datetime(2000, 1, 1, 12, 0), datetime(2000, 1, 2, 12, 0), datetime(2000, 1, 3, 1
})
df = spark.createDataFrame(pandas_df)
df
```

```
DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]
```

```
rdd = spark.sparkContext.parallelize([
    (1, 2., 'string1', date(2000, 1, 1), datetime(2000, 1, 1, 12, 0)),
    (2, 3., 'string2', date(2000, 2, 1), datetime(2000, 1, 2, 12, 0)),
    (3, 4., 'string3', date(2000, 3, 1), datetime(2000, 1, 3, 12, 0))
])
df = spark.createDataFrame(rdd, schema=['a', 'b', 'c', 'd', 'e'])
df
```

```
DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]
```

```
# All DataFrames above result same.
df.show()
```

```
+---+---+-------+----------+-------------------+
|  a|  b|      c|         d|                  e|
+---+---+-------+----------+-------------------+
|  1|2.0|string1|2000-01-01|2000-01-01 12:00:00|
|  2|3.0|string2|2000-02-01|2000-01-02 12:00:00|
|  3|4.0|string3|2000-03-01|2000-01-03 12:00:00|
+---+---+-------+----------+-------------------+
```

```
df.printSchema()
```

```
root
 |-- a: long (nullable = true)
 |-- b: double (nullable = true)
 |-- c: string (nullable = true)
 |-- d: date (nullable = true)
 |-- e: timestamp (nullable = true)
```

```
df.show(1)
```

```
+---+---+-------+----------+-------------------+
|  a|  b|      c|         d|                  e|
+---+---+-------+----------+-------------------+
|  1|2.0|string1|2000-01-01|2000-01-01 12:00:00|
+---+---+-------+----------+-------------------+
only showing top 1 row
```

```
spark.conf.set('spark.sql.repl.eagerEval.enabled', True)
```

```
df
```

a	b	c	d	e
1	2.0	string1	2000-01-01	2000-01-01 12:00:00
2	3.0	string2	2000-02-01	2000-01-02 12:00:00
3	4.0	string3	2000-03-01	2000-01-03 12:00:00

```
df.show(1, vertical=True)
```

```
-RECORD 0------------------
 a   | 1
 b   | 2.0
 c   | string1
 d   | 2000-01-01
 e   | 2000-01-01 12:00:00
only showing top 1 row
```

```
df.columns
```

```
['a', 'b', 'c', 'd', 'e']
```

```
df.printSchema()
```

```
root
 |-- a: long (nullable = true)
 |-- b: double (nullable = true)
 |-- c: string (nullable = true)
 |-- d: date (nullable = true)
 |-- e: timestamp (nullable = true)
```

```
df.select("a", "b", "c").describe().show()#when we are using session so the data will be s
```

```
+-------+---+---+-------+
|summary|  a|  b|      c|
+-------+---+---+-------+
|  count|  3|  3|      3|
|   mean|2.0|3.0|   null|
| stddev|1.0|1.0|   null|
|    min|  1|2.0|string1|
|    max|  3|4.0|string3|
+-------+---+---+-------+
```

```
df.collect()
```

```
[Row(a=1, b=2.0, c='string1', d=datetime.date(2000, 1, 1), e=datetime.datetime(2000,
1, 1, 12, 0)),
 Row(a=2, b=3.0, c='string2', d=datetime.date(2000, 2, 1), e=datetime.datetime(2000,
1, 2, 12, 0)),
 Row(a=3, b=4.0, c='string3', d=datetime.date(2000, 3, 1), e=datetime.datetime(2000,
1, 3, 12, 0))]
```

```
df.take(2)
```

```
[Row(a=1, b=2.0, c='string1', d=datetime.date(2000, 1, 1), e=datetime.datetime(2000,
1, 1, 12, 0)),
 Row(a=2, b=3.0, c='string2', d=datetime.date(2000, 2, 1), e=datetime.datetime(2000,
1, 2, 12, 0))]
```

```
n=df.toPandas()
```

```
n
```

	a	b	c	d	e
0	1	2.0	string1	2000-01-01	2000-01-01 12:00:00
1	2	3.0	string2	2000-02-01	2000-01-02 12:00:00
2	3	4.0	string3	2000-03-01	2000-01-03 12:00:00

```
#Selecting and Accessing Data¶
```

```
df.a
```

```
Column<'a'>
```

```
# checking column value types

from pyspark.sql import Column
from pyspark.sql.functions import upper

type(df.a) == type(upper(df.a)) == type(df.a.isNull())
```

```
True
```

```
type(df.a)
```

```
pyspark.sql.column.Column
```

```
type(upper(df.a))
```

```
pyspark.sql.column.Column
```

```
type(df.a.isNull())
```

```
pyspark.sql.column.Column
```

```
df.select(df.b).show()
```

```
+---+
```

```
|  b|
+---+
|2.0|
|3.0|
|4.0|
+---+
```

```
df.withColumn('upper_c', upper(df.c)).show()#add new column in data frame
```

```
+---+---+-------+----------+-------------------+-------+
|  a|  b|      c|         d|                  e|upper_c|
+---+---+-------+----------+-------------------+-------+
|  1|2.0|string1|2000-01-01|2000-01-01 12:00:00|STRING1|
|  2|3.0|string2|2000-02-01|2000-01-02 12:00:00|STRING2|
|  3|4.0|string3|2000-03-01|2000-01-03 12:00:00|STRING3|
+---+---+-------+----------+-------------------+-------+
```

```
df.filter(df.a == 3).show()
```

```
+---+---+-------+----------+-------------------+
|  a|  b|      c|         d|                  e|
+---+---+-------+----------+-------------------+
|  3|4.0|string3|2000-03-01|2000-01-03 12:00:00|
+---+---+-------+----------+-------------------+
```

```
df.filter(df.c == 'string2').show()
```

```
+---+---+-------+----------+-------------------+
|  a|  b|      c|         d|                  e|
+---+---+-------+----------+-------------------+
|  2|3.0|string2|2000-02-01|2000-01-02 12:00:00|
+---+---+-------+----------+-------------------+
```

```
#Applying a Function:
```

```
import pandas as pd
from pyspark.sql.functions import pandas_udf

@pandas_udf('long')# user define functions in spark session for applying function.
def pandas_plus_one(series: pd.Series,Series1:pd.Series) -> pd.Series:
    # Simply plus one by using pandas Series.
    return series+Series1

df.select(pandas_plus_one(df.a,df.b)).show()
```

```
+--------------------+
|pandas_plus_one(a, b)|
+--------------------+
|                   3|
|                   5|
```

```
|                    7|
+---------------------+
```