

# Loading Data with Schema

**Nikhilesh Vashisht**

<b>Create a schema for the data frame using an object of class StructType</b>	<b>2</b>
<b>Question 1</b>	<b>3</b>
<b>Question 2</b>	<b>5</b>
<b>Question 3</b>	<b>6</b>
<b>Question 4</b>	<b>7</b>
<b>Question 5</b>	<b>8</b>

# Create a schema for the data frame using an object of class StructType

Importing the class:

```
import org.apache.spark.sql.Row;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;
import org.apache.spark.sql.types.StringType;
import org.apache.spark.sql.types.DataTypes._;
```

Specifying schema:

```
val schema = new StructType()
.add("exchange",StringType,true)
.add("symbol",StringType,true)
.add("trade_date",StringType,true)
.add("open_price",DataTypes.DoubleType,true)
.add("high_price",DataTypes.DoubleType,true)
.add("low_price",DataTypes.DoubleType,true)
.add("close_price",DataTypes.DoubleType,true)
.add("volume",IntegerType,true)
.add("adjusted_close",DataTypes.DoubleType,true)
```

```
scala> val schema = new StructType();
schema: org.apache.spark.sql.types.StructType = StructType()

scala> schema = schema.add("check1",DataTypes.DoubleType,false)

scala> :paste
// Entering paste mode (ctrl-D to finish)

val schema = new StructType()
.add("exchange",StringType,true)
.add("symbol",StringType,true)
.add("trade_date",StringType,true)
.add("open_price",DataTypes.DoubleType,true)
.add("high_price",DataTypes.DoubleType,true)
.add("low_price",DataTypes.DoubleType,true)
.add("close_price",DataTypes.DoubleType,true)
.add("volume",IntegerType,true)
.add("adjusted_close",DataTypes.DoubleType,true)

// Exiting paste mode, now interpreting.
```

Loading the stocks data

```
val stocksDf = spark.read.format("csv")
.option("header", "true")
.schema(schema)
.load("hdfs://10.128.0.17/sparkLab2/stocks")
```

```
, StructField(adjusted_close,DoubleType,true))

scala> :paste
// Entering paste mode (ctrl-D to finish)

val stocksDf = spark.read.format("csv")
.option("header", "true")
.schema(schema)
.load("hdfs://10.128.0.17/sparkLab2/stocks")

// Exiting paste mode, now interpreting.

stocksDf: org.apache.spark.sql.DataFrame = [exchange: string, symbol: string ... 7 more fields]

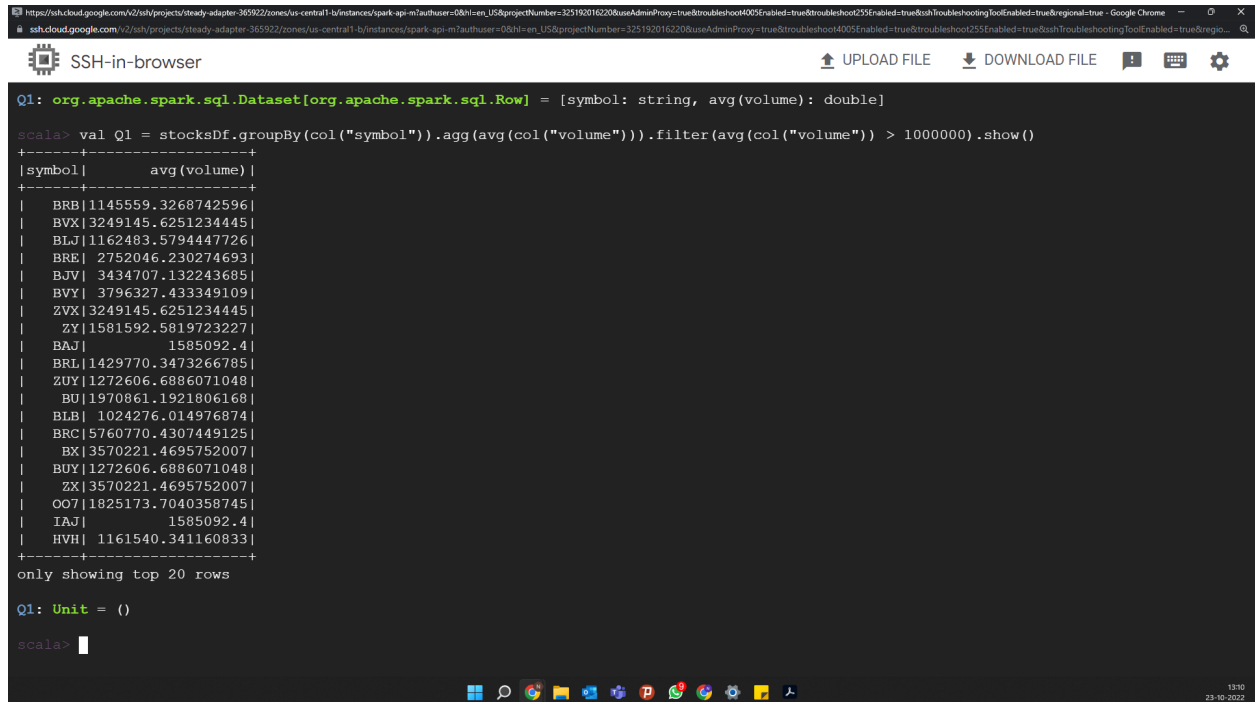
scala> stocksDf.schema
res3: org.apache.spark.sql.types.StructType = StructType(StructField(exchange,StringType,true), StructField(symbol,StringType,true), StructField(trade_date,StringType,true), StructField(open_price,DoubleType,true), StructField(high_price,DoubleType,true), StructField(low_price,DoubleType,true), StructField(close_price,DoubleType,true), StructField(volume,IntegerType,true), StructField(adjusted_close,DoubleType,true))
```

# Problem 1

Write a command to find the stocks with average daily volume larger than 1 million shares

val Q1 =

```
stocksDf.select(col("symbol"),col("trade_date"),col("volume")).groupBy(col("symbol")).agg(avg(col("volume"))).filter(avg(col("volume")) > 1000000).show()
```



The screenshot shows a web browser window with an SSH connection to a Google Cloud instance. The terminal displays a Scala command to filter stocks by average daily volume and shows the top 20 results.

```
Q1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [symbol: string, avg(volume): double]

scala> val Q1 = stocksDf.groupBy(col("symbol")).agg(avg(col("volume"))).filter(avg(col("volume")) > 1000000).show()
+-----+-----+
|symbol|      avg(volume)|
+-----+-----+
|BRB|1145559.3268742596|
|BVX|3249145.6251234445|
|BLJ|1162483.5794447726|
|BRE|2752046.230274693|
|BJV|3434707.132243685|
|BVI|3796327.433349109|
|ZVX|3249145.6251234445|
|ZY|1581592.5819723227|
|BAJ|1585092.4|
|BRL|1429770.3473266785|
|ZUY|1272606.6886071048|
|BU|1970861.1921806168|
|BLB|1024276.014976874|
|BRC|5760770.4307449125|
|BX|3570221.4695752007|
|BUY|1272606.6886071048|
|ZX|3570221.4695752007|
|OO7|1825173.7040358745|
|IAJ|1585092.4|
|HVH|1161540.341160833|
+-----+-----+
only showing top 20 rows

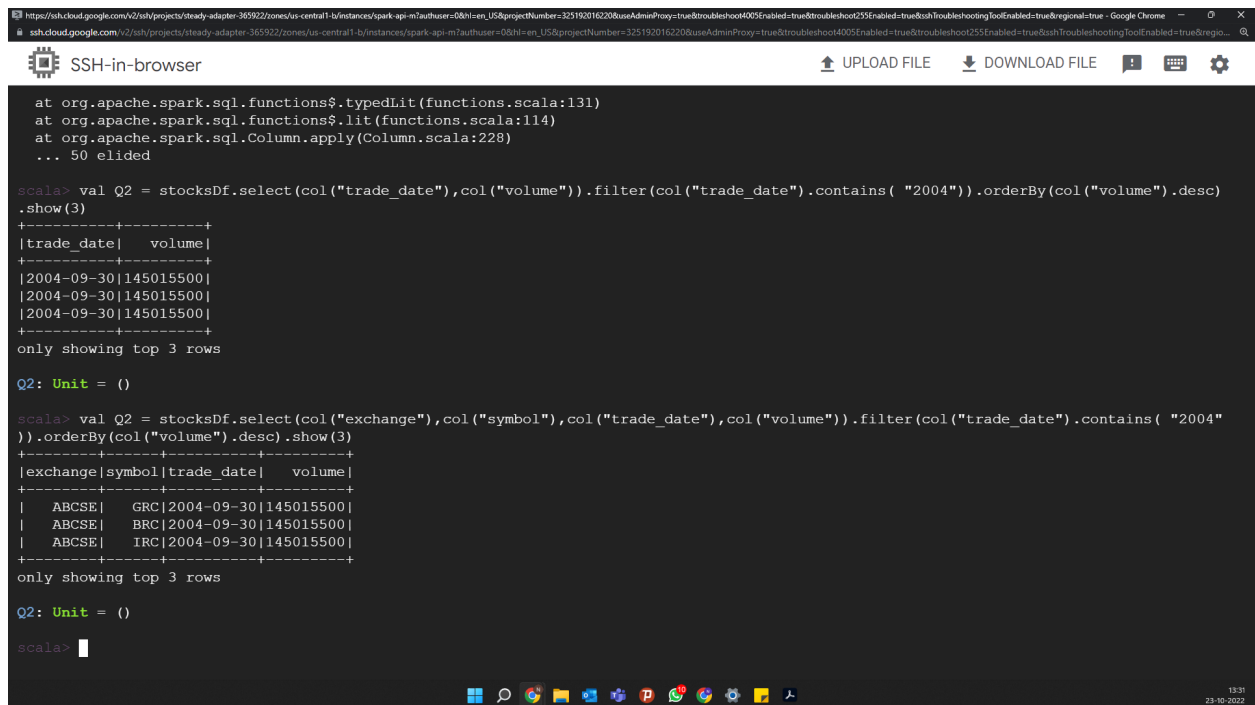
Q1: Unit = ()

scala>
```

# Problem 2

Write a Scala DataFrame query to find the top 3 stocks by volume for the year 2004.

```
val Q2 =  
stocksDf.select(col("exchange"),col("symbol"),col("trade_date"),col("volume")).filter(col("trade_date").contains("2004")).orderBy(col("volume").desc).show(3)
```



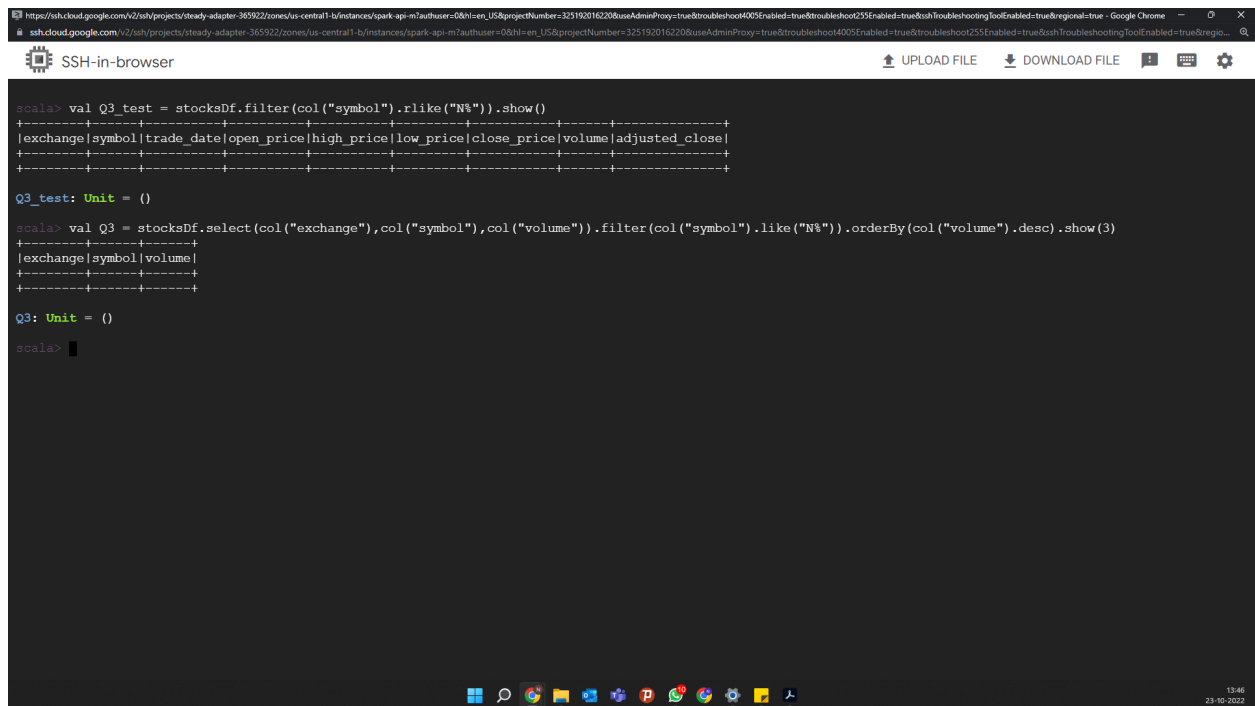
The screenshot shows a web browser window with the address bar displaying a Google Cloud SSH-in-browser URL. The browser window has a title bar with "SSH-in-browser" and buttons for "UPLOAD FILE" and "DOWNLOAD FILE". The main content area shows a Scala REPL session. The first query filters for the year 2004 and orders by volume, showing the top 3 rows. The second query filters for the year 2004 and orders by volume, showing the top 3 rows. The output of the second query is a table with 4 columns: exchange, symbol, trade\_date, and volume. The table shows three rows of data for the year 2004, all with a volume of 145015500.

```
at org.apache.spark.sql.functions$.typedLit(functions.scala:131)  
at org.apache.spark.sql.functions$.lit(functions.scala:114)  
at org.apache.spark.sql.Column.apply(Column.scala:228)  
... 50 elided  
  
scala> val Q2 = stocksDf.select(col("trade_date"),col("volume")).filter(col("trade_date").contains("2004")).orderBy(col("volume").desc).show(3)  
.show(3)  
+-----+-----+  
|trade_date| volume|  
+-----+-----+  
|2004-09-30|145015500|  
|2004-09-30|145015500|  
|2004-09-30|145015500|  
+-----+-----+  
only showing top 3 rows  
  
Q2: Unit = ()  
  
scala> val Q2 = stocksDf.select(col("exchange"),col("symbol"),col("trade_date"),col("volume")).filter(col("trade_date").contains("2004")).orderBy(col("volume").desc).show(3)  
.show(3)  
+-----+-----+-----+-----+  
|exchange|symbol|trade_date| volume|  
+-----+-----+-----+-----+  
| ABCSE | GRC |2004-09-30|145015500|  
| ABCSE | BRC |2004-09-30|145015500|  
| ABCSE | IRC |2004-09-30|145015500|  
+-----+-----+-----+-----+  
only showing top 3 rows  
  
Q2: Unit = ()  
  
scala> 
```

## Problem 3

Write a Scala DataFrame query to find the top 3 stocks by volume and whose symbol start with the first letter of your name (example for Saber, it is symbols starting with “S”).

```
val Q3 =  
stocksDf.select(col("exchange"),col("symbol"),col("volume")).filter(col("symbol").like("N%")).orde  
rBy(col("volume").desc).show(3)
```



```
scala> val Q3_test = stocksDf.filter(col("symbol").rlike("N%")).show()  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|exchange|symbol|trade_date|open_price|high_price|low_price|close_price|volume|adjusted_close|  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
Q3_test: Unit = ()  
  
scala> val Q3 = stocksDf.select(col("exchange"),col("symbol"),col("volume")).filter(col("symbol").like("N%")).orderBy(col("volume").desc).show(3)  
+-----+-----+-----+  
|exchange|symbol|volume|  
+-----+-----+-----+  
Q3: Unit = ()  
  
scala>
```

There were no symbols starting with the first letter of my name (N)

```
https://ssh.cloud.google.com/v2/ssh/projects/steady-adaptor-365922/zones/us-central1-b/instances/park-api-m?authuser=0&hl=en_US#projectNumber=32519010200&useAdminProxy=true&troubleshoot4005enabled=true&troubleshoot4005rated=true&troubleshootingtoolEnabled=true&xgafid=true - Google Chrome
```

```
SSH-in-browser
```

```
| ANICSE| B7B|    36.561  
| ANICSE| B7B|    36.251  
| ANICSE| B7B|    36.241  
| ANICSE| B7B|    36.041  
| ANICSE| B7B|    36.221  
| ANICSE| B7B|    36.211  
| ANICSE| B7B|    36.021  
| ANICSE| B7B|    35.941  
| ANICSE| B7B|    36.181  
| ANICSE| B7B|    36.651  
| ANICSE| B7B|    36.511  
| ANICSE| B7B|    35.491  
| ANICSE| B7B|    35.711  
| ANICSE| B7B|    36.091  
| ANICSE| B7B|    35.831  
| ANICSE| B7B|    35.981  
| ANICSE| B7B|    36.051  
| ANICSE| B7B|    36.571  
| ANICSE| B7B|    36.891  
=====  
only showing top 100 rows  
  
Q4: Unit = ()  
  
In [4]: val Q4 = stocksDf.select(col("exchange"), col("symbol"), col("close_price")).filter(col("close_price")>24).show()  
-----  
| exchange| symbol| close_price|  
-----+-----+-----  
| ANICSE| B7B|      36.681  
| ANICSE| B7B|      39.041  
| ANICSE| B7B|      38.241  
| ANICSE| B7B|      38.321  
| ANICSE| B7B|      38.511  
| ANICSE| B7B|      38.251  
| ANICSE| B7B|      38.221  
| ANICSE| B7B|      38.341  
| ANICSE| B7B|      38.581  
| ANICSE| B7B|      38.081  
| ANICSE| B7B|      38.111  
| ANICSE| B7B|      37.491  
| ANICSE| B7B|      38.011  
| ANICSE| B7B|      38.311  
| ANICSE| B7B|      38.611  
| ANICSE| B7B|      38.781  
| ANICSE| B7B|      39.241  
| ANICSE| B7B|      39.351  
| ANICSE| B7B|      38.821  
| ANICSE| B7B|      38.971  
=====  
only showing top 20 rows  
  
Q4: Unit = ()  
  
In [4]:
```






**Write a Scala DataFrame to find the top 10 stocks with largest intraday price change (difference between high and low price during a trading day) and also display the amount of the change.**

sshcloud.google.com/v2/ssh/projects/steady-adaptor-365922/zone/us-central1/instances/spark-api-m7/authuser=59Ht=USCloudProjectNumber=32519216229UserAdminProject=trueTroubleshoot4005Enabled=trueTroubleshootingToolEnabled=trueEngo...

SSH-in-browser

UPLOAD FILE

DOWNLOAD FILE



only showing top 20 rows

Q4: Unit = ()

```
scala> val Q5 = stockDf.select(col("exchange"),col("symbol"),col("high_price"),col("low_price"),(col("high_price")-col("low_price")).alias("price_change_in_a_day")).show()

[exchange|symbol|high_price|low_price|price_change_in_a_day|
|-----|-----|-----|-----|-----|
| ABCSE| B7J| 8.71| 8.57| 0.12999999999999999|
| ABCSE| B7J| 8.71| 8.31| 0.40000000000000006|
| ABCSE| B7J| 8.88| 8.59| 0.29000000000000009|
| ABCSE| B7J| 8.92| 8.8| 0.11999999999999992|
| ABCSE| B7J| 8.9| 8.73| 0.16999999999999993|
| ABCSE| B7J| 8.77| 8.66| 0.10999999999999993|
| ABCSE| B7J| 8.81| 8.56| 0.25|
| ABCSE| B7J| 8.9| 8.61| 0.30000000000000007|
| ABCSE| B7J| 8.87| 8.68| 0.18999999999999995|
| ABCSE| B7J| 8.92| 8.71| 0.20999999999999998|
| ABCSE| B7J| 9.01| 8.73| 0.26999999999999996|
| ABCSE| B7J| 9.04| 8.94| 0.09999999999999994|
| ABCSE| B7J| 9.13| 8.98| 0.15000000000000003|
| ABCSE| B7J| 9.09| 8.93| 0.16000000000000004|
| ABCSE| B7J| 9.04| 8.93| 0.12999999999999999|
| ABCSE| B7J| 9.01| 8.91| 0.09999999999999994|
| ABCSE| B7J| 9.03| 8.93| 0.09999999999999994|
| ABCSE| B7J| 9.01| 8.92| 0.08000000000000007|
| ABCSE| B7J| 8.95| 8.86| 0.08999999999999996|
| ABCSE| B7J| 9.01| 8.95| 0.05000000000000071|
```

only showing top 20 rows

Q5: Unit = ()

```
scala> val Q5 = stockDf.select(col("exchange"),col("symbol"),col("high_price"),col("low_price"),(col("high_price")-col("low_price")).alias("price_change_in_a_day")).orderBy(col("price_change_in_a_day").desc).show(10)

[exchange|symbol|high_price|low_price|price_change_in_a_day|
|-----|-----|-----|-----|-----|
| ABCSE| GBR| 583.51| 475.17| 108.33999999999997|
| ABCSE| HBR| 583.51| 475.17| 108.33999999999997|
| ABCSE| ZBR| 583.51| 475.17| 108.33999999999997|
| ABCSE| HBR| 583.51| 475.17| 108.33999999999997|
| ABCSE| HBR| 583.51| 475.17| 108.33999999999997|
| ABCSE| GCL| 480.01| 380.11| 99.899999999999998|
| ABCSE| OCL| 480.01| 380.11| 99.899999999999998|
| ABCSE| ICL| 480.01| 380.11| 99.899999999999998|
| ABCSE| BCL| 480.01| 380.11| 99.899999999999998|
| ABCSE| HBR| 421.01| 338.66| 82.339999999999997|
```

only showing top 10 rows

Q5: Unit = ()

```
scala>
```

1556

23-10-2022