Lab 2 BDAT 1008
Loading Data with Schema

Nikhilesh Ramabhadraiah Vashisht
200512808

BDAT 1008 – Data Collection and Curation - Lab 2

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

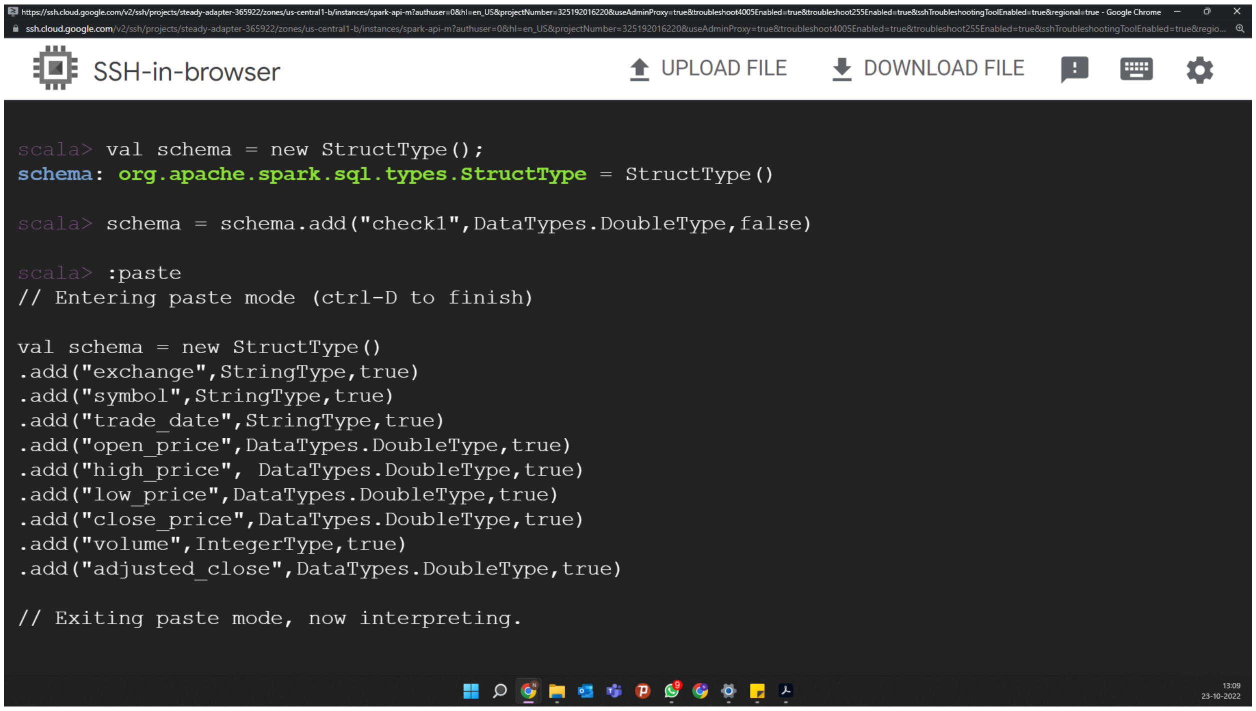
Importing the class:

```
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)
```

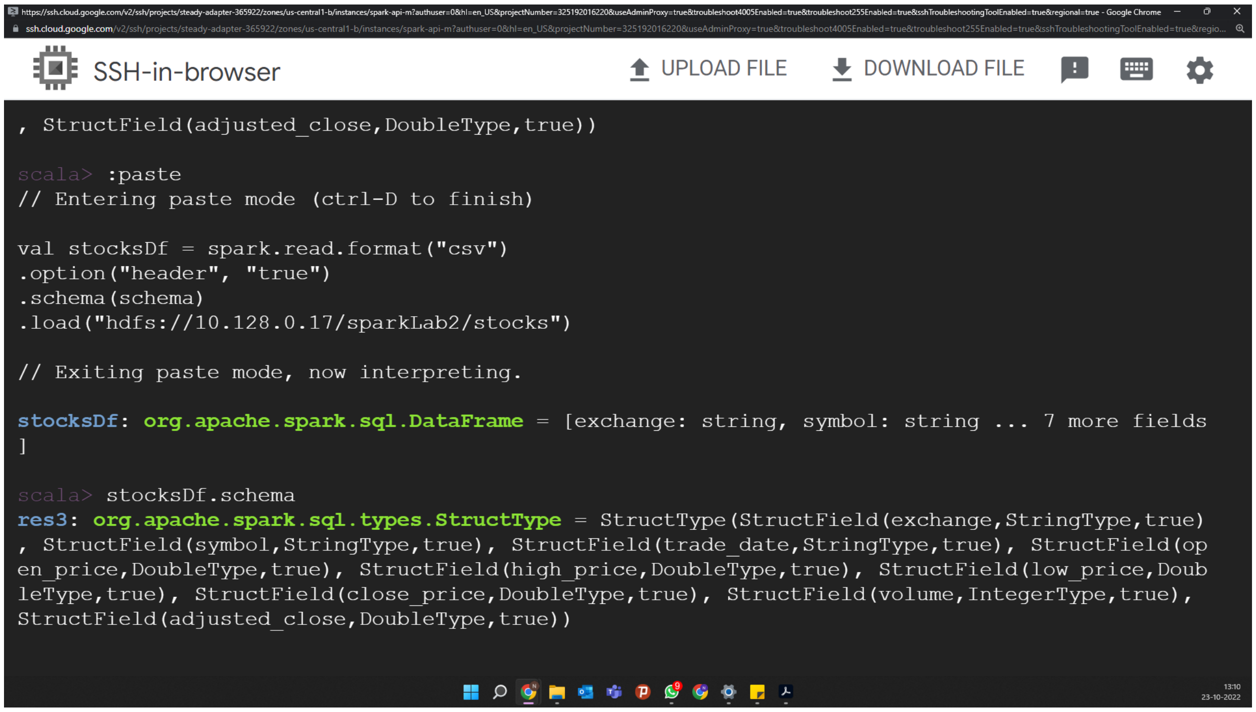
BDAT 1008 – Data Collection and Curation - Lab 2



```
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")

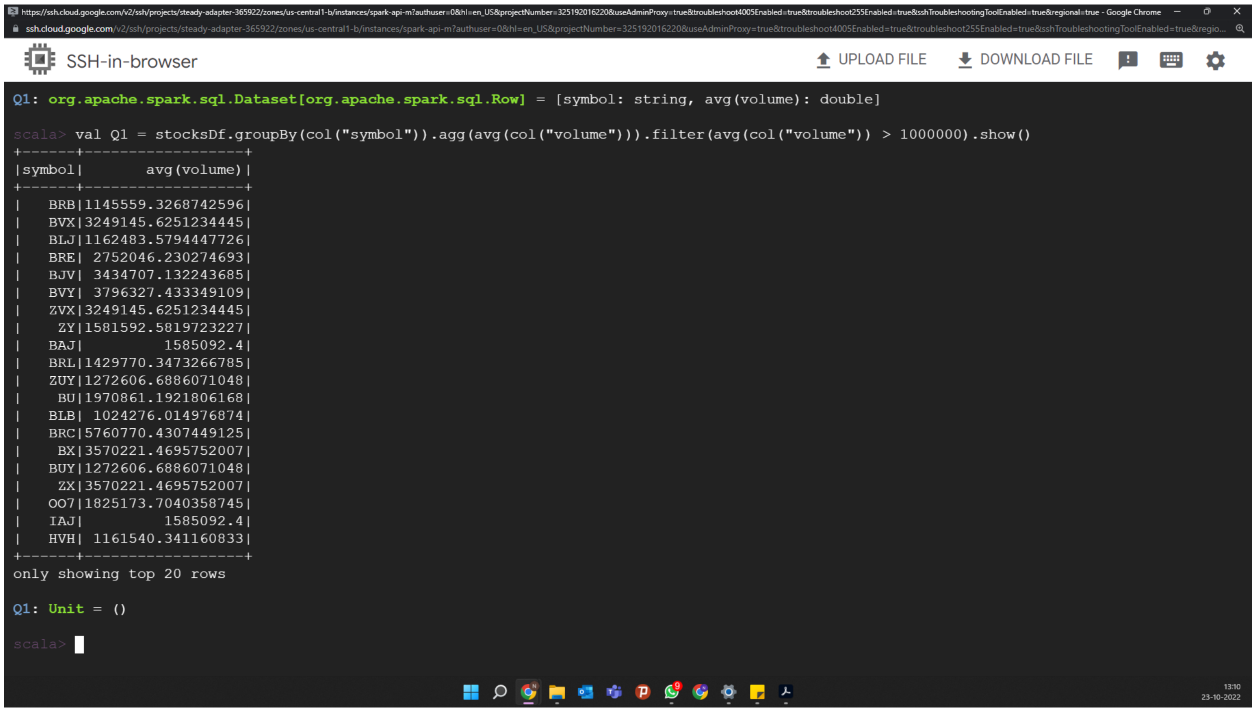


```
, StructField(adjusted_close,DoubleType,true))

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(op
en_price,DoubleType,true), StructField(high_price,DoubleType,true), StructField(low_price,Doub
leType,true), StructField(close_price,DoubleType,true), StructField(volume,IntegerType,true),
StructField(adjusted_close,DoubleType,true))
```

Question 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()

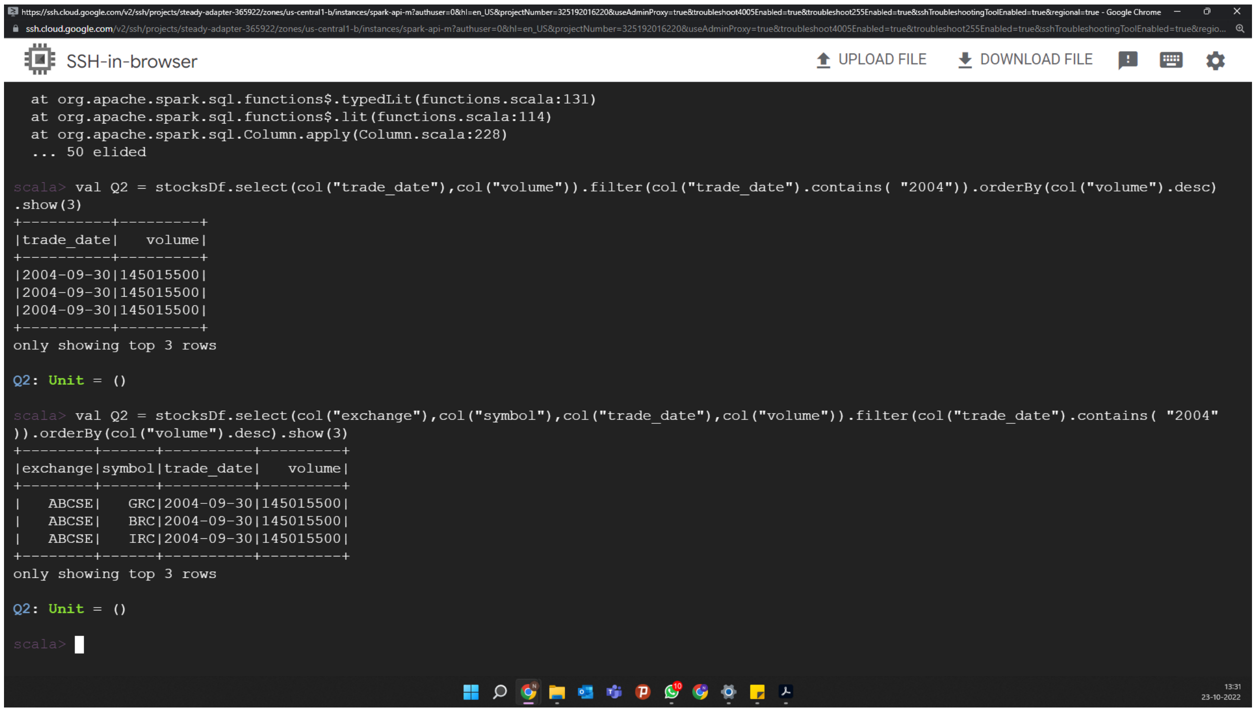


```
Q1: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [symbol: string, avg(volume): double]

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|
|   BVY| 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> 
```

Question 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)



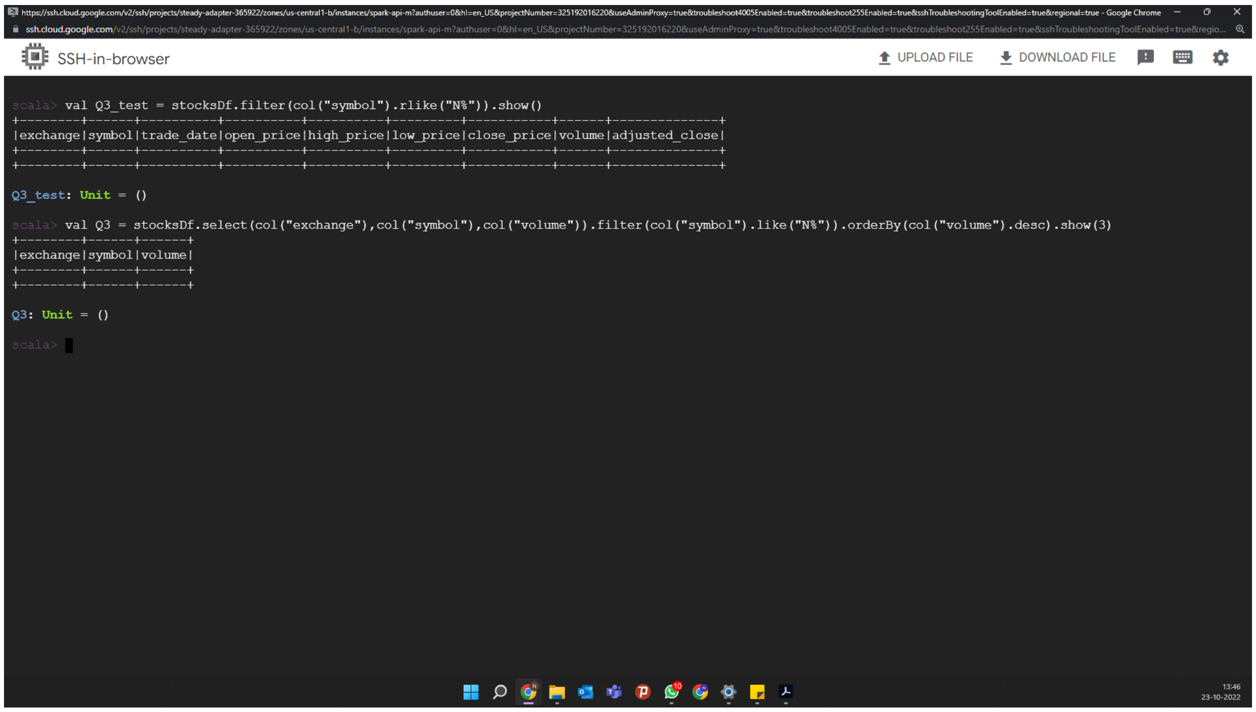
```
https://ssh.cloud.google.com/v2/ssh/projects/steady-adapter-365922/zones/us-central1-b/instances/spark-api-m?authuser=0&hl=en_US&projectNumber=325192016220&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&sshTroubleshootingToolEnabled=true&regional=true - Google Chrome
ssh.cloud.google.com/v2/ssh/projects/steady-adapter-365922/zones/us-central1-b/instances/spark-api-m?authuser=0&hl=en_US&projectNumber=325192016220&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&sshTroubleshootingToolEnabled=true&regio...
```

```
  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)
+----------+---------+
|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)
+--------+------+----------+---------+
|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> 
```

Question 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%")).orderBy(col("volume").desc).show(3)


https://ssh.cloud.google.com/v2/ssh/projects/steady-adapter-365922/zones/us-central1-b/instances/spark-api-m?authuser=0&hl=en_US&projectNumber=325192016220&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true&sshTroubleshootingToolEnabled=true&regional=true - Google Chrome
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
13:46
23-10-2022

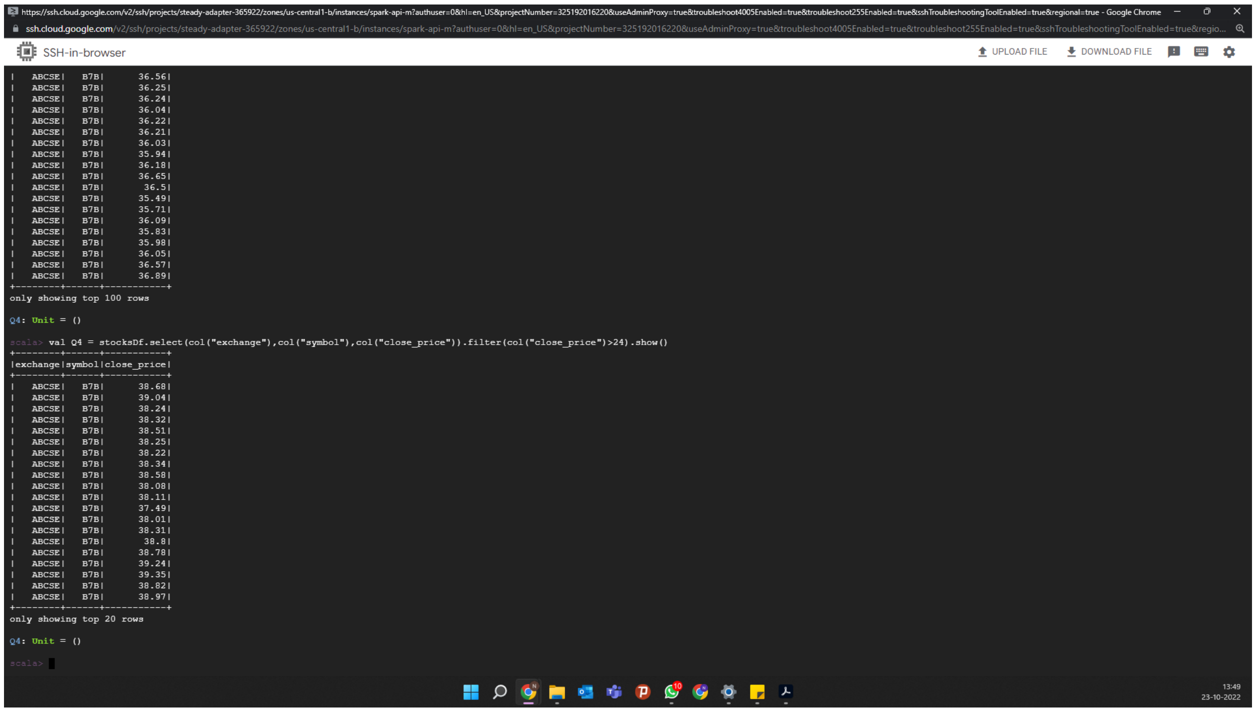
```
scala> val Q3_test = stocksDf.filter(col("symbol").rlike("N%")).show()
+--------+------+----------+----------+----------+---------+-----------+------+--------------+
|exchange|symbol|trade_date|open_price|high_price|low_price|close_price|volume|adjusted_close|
+--------+------+----------+----------+----------+---------+-----------+------+--------------+
+--------+------+----------+----------+----------+---------+-----------+------+--------------+

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)

Question 4

Write a Scala DataFrame to find all the stocks symbols whose closing price is larger than your age.

val Q4 = stocksDf.select(col("exchange"),col("symbol"),col("close_price")).filter(col("close_price")>24).show()

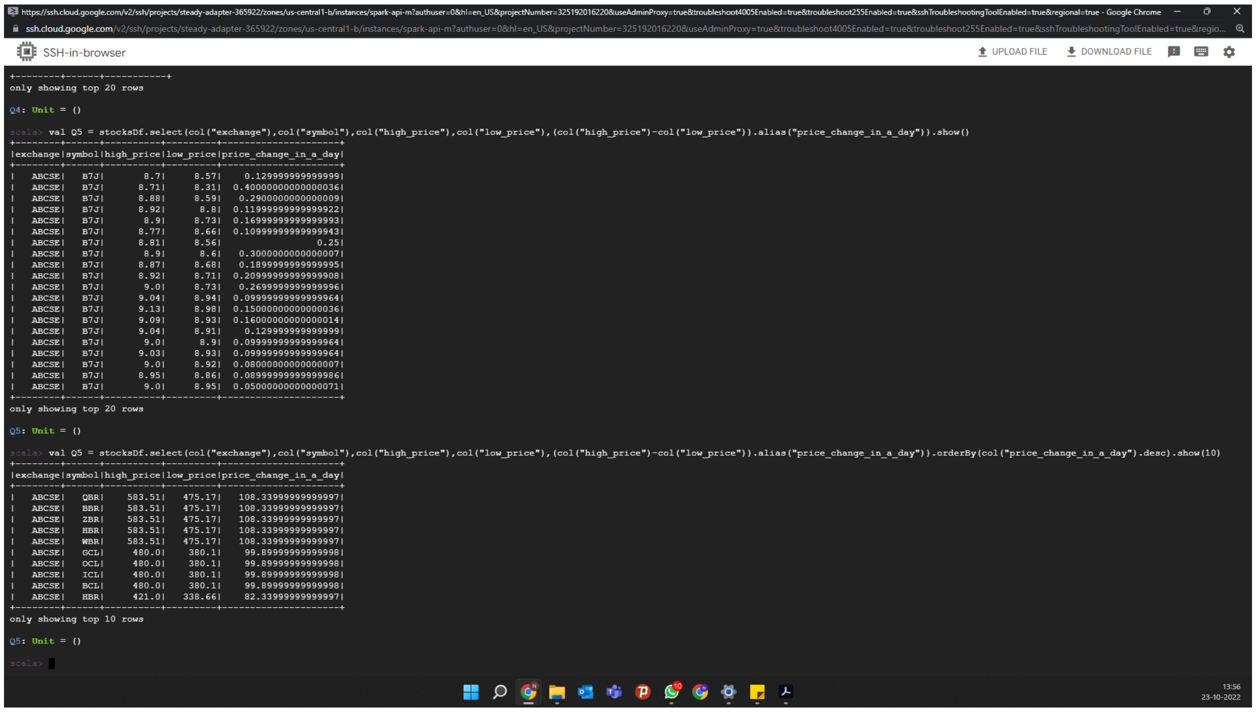


```
|   ABCSE|   B7B|      36.56|
|   ABCSE|   B7B|      36.25|
|   ABCSE|   B7B|      36.24|
|   ABCSE|   B7B|      36.04|
|   ABCSE|   B7B|      36.22|
|   ABCSE|   B7B|      36.21|
|   ABCSE|   B7B|      36.03|
|   ABCSE|   B7B|      35.94|
|   ABCSE|   B7B|      36.18|
|   ABCSE|   B7B|      36.65|
|   ABCSE|   B7B|       36.5|
|   ABCSE|   B7B|      35.49|
|   ABCSE|   B7B|      35.71|
|   ABCSE|   B7B|      36.09|
|   ABCSE|   B7B|      35.83|
|   ABCSE|   B7B|      35.98|
|   ABCSE|   B7B|      36.05|
|   ABCSE|   B7B|      36.57|
|   ABCSE|   B7B|      36.89|
+--------+------+-----------+
only showing top 100 rows

Q4: Unit = ()

scala> val Q4 = stocksDf.select(col("exchange"),col("symbol"),col("close_price")).filter(col("close_price")>24).show()
+--------+------+-----------+
|exchange|symbol|close_price|
+--------+------+-----------+
|   ABCSE|   B7B|      38.68|
|   ABCSE|   B7B|      39.04|
|   ABCSE|   B7B|      38.24|
|   ABCSE|   B7B|      38.32|
|   ABCSE|   B7B|      38.51|
|   ABCSE|   B7B|      38.25|
|   ABCSE|   B7B|      38.22|
|   ABCSE|   B7B|      38.34|
|   ABCSE|   B7B|      38.58|
|   ABCSE|   B7B|      38.08|
|   ABCSE|   B7B|      38.11|
|   ABCSE|   B7B|      37.49|
|   ABCSE|   B7B|      38.01|
|   ABCSE|   B7B|      38.31|
|   ABCSE|   B7B|       38.8|
|   ABCSE|   B7B|      38.78|
|   ABCSE|   B7B|      39.24|
|   ABCSE|   B7B|      39.35|
|   ABCSE|   B7B|      38.82|
|   ABCSE|   B7B|      38.97|
+--------+------+-----------+
only showing top 20 rows

Q4: Unit = ()

scala> 
```

Question 5

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.

val Q5 = stocksDf.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)



```
+--------+------+-----------+
only showing top 20 rows

Q4: Unit = ()

scala> val Q5 = stocksDf.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|       8.7|     8.57|   0.129999999999999|
|   ABCSE|   B7J|      8.71|     8.31|  0.40000000000000036|
|   ABCSE|   B7J|      8.88|     8.59|   0.2900000000000009|
|   ABCSE|   B7J|      8.92|      8.8|  0.11999999999999922|
|   ABCSE|   B7J|       8.9|     8.73|  0.16999999999999993|
|   ABCSE|   B7J|      8.77|     8.66|  0.10999999999999943|
|   ABCSE|   B7J|      8.81|     8.56|                 0.25|
|   ABCSE|   B7J|       8.9|      8.6|   0.3000000000000007|
|   ABCSE|   B7J|      8.87|     8.68|   0.1899999999999995|
|   ABCSE|   B7J|      8.92|     8.71|  0.20999999999999908|
|   ABCSE|   B7J|       9.0|     8.73|   0.2699999999999996|
|   ABCSE|   B7J|      9.04|     8.94|  0.09999999999999964|
|   ABCSE|   B7J|      9.13|     8.98|  0.15000000000000036|
|   ABCSE|   B7J|      9.09|     8.93|  0.16000000000000014|
|   ABCSE|   B7J|      9.04|     8.91|   0.129999999999999|
|   ABCSE|   B7J|       9.0|      8.9|  0.09999999999999964|
|   ABCSE|   B7J|      9.03|     8.93|  0.09999999999999964|
|   ABCSE|   B7J|       9.0|     8.92|  0.08000000000000007|
|   ABCSE|   B7J|      8.95|     8.86|  0.08999999999999986|
|   ABCSE|   B7J|       9.0|     8.95|  0.05000000000000071|
+--------+------+----------+---------+--------------------+
only showing top 20 rows

Q5: Unit = ()

scala> val Q5 = stocksDf.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|   QBR|    583.51|   475.17|   108.33999999999997|
|   ABCSE|   BBR|    583.51|   475.17|   108.33999999999997|
|   ABCSE|   ZBR|    583.51|   475.17|   108.33999999999997|
|   ABCSE|   HBR|    583.51|   475.17|   108.33999999999997|
|   ABCSE|   WBR|    583.51|   475.17|   108.33999999999997|
|   ABCSE|   GCL|     480.0|    380.1|    99.89999999999998|
|   ABCSE|   OCL|     480.0|    380.1|    99.89999999999998|
|   ABCSE|   ICL|     480.0|    380.1|    99.89999999999998|
|   ABCSE|   BCL|     480.0|    380.1|    99.89999999999998|
|   ABCSE|   HBR|     421.0|   338.66|    82.33999999999997|
+--------+------+----------+---------+--------------------+
only showing top 10 rows

Q5: Unit = ()

scala> 
```