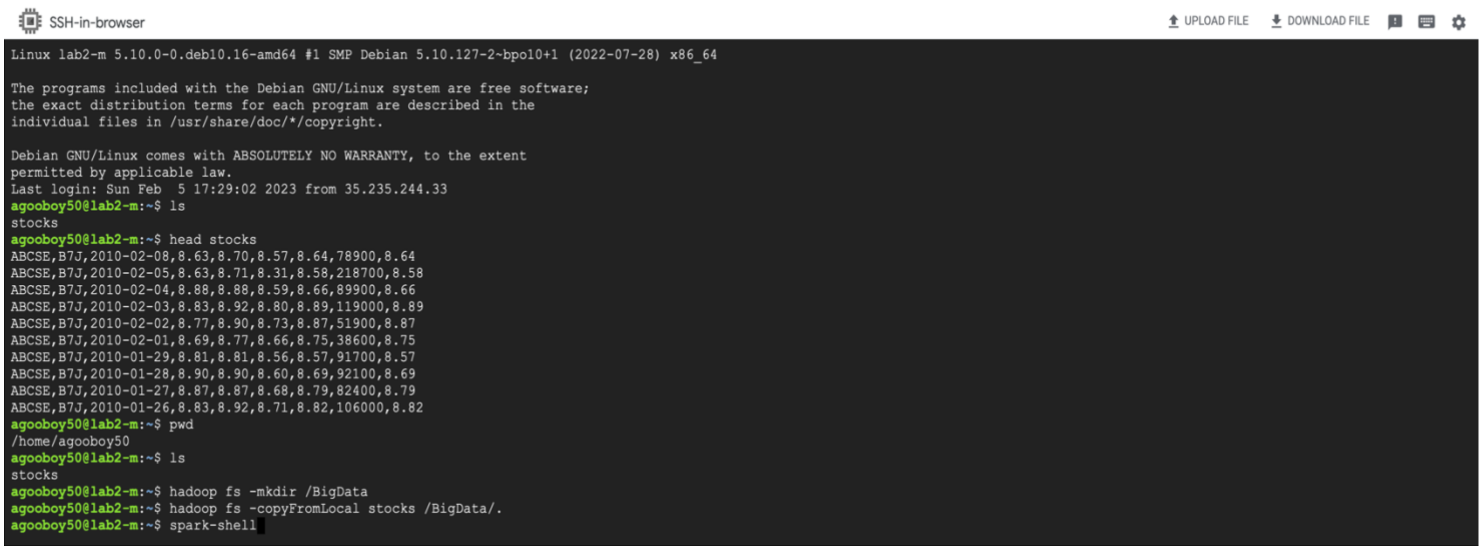
In-class Lab – Loading Data with Schema

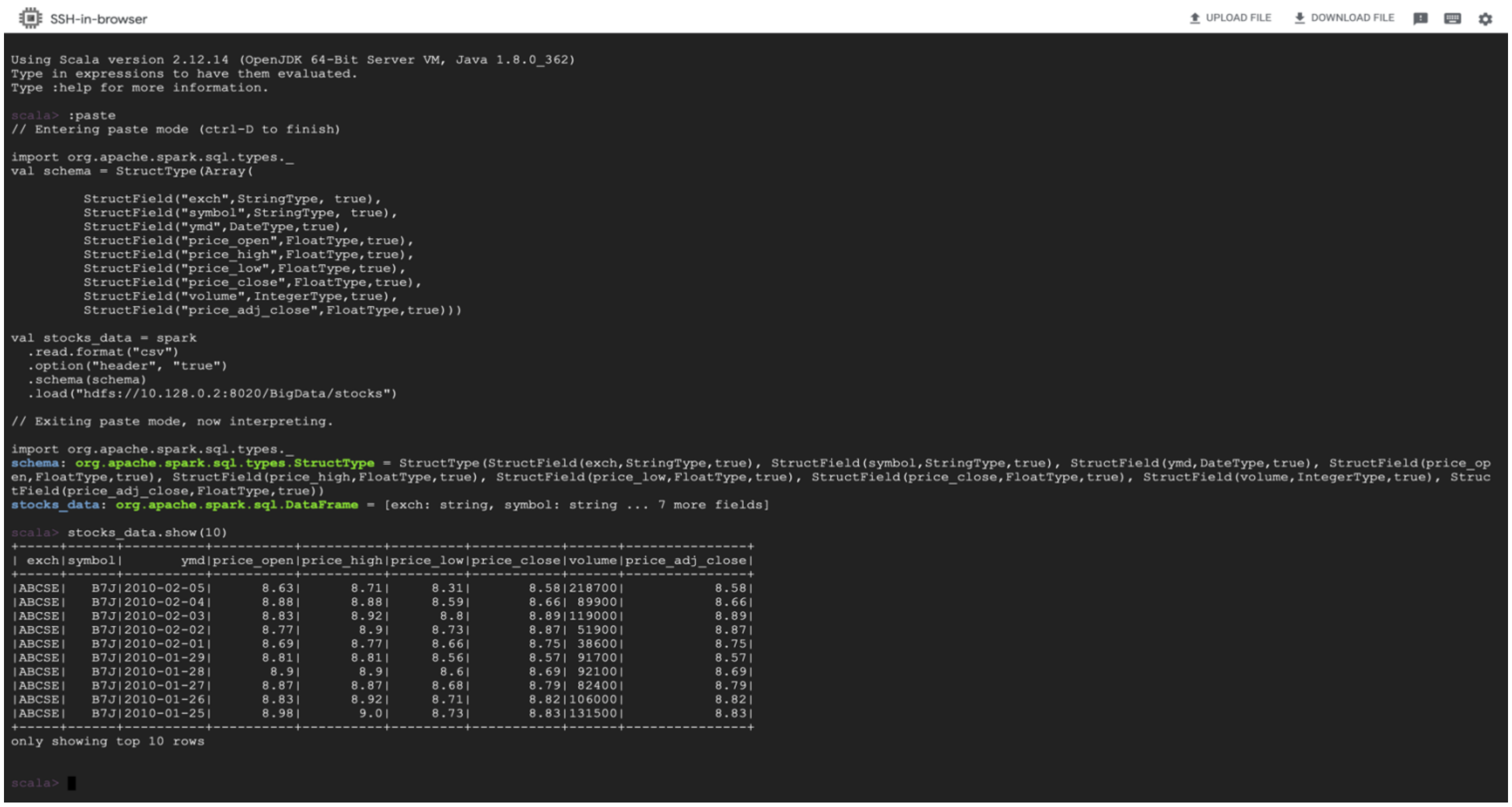


```
Linux lab2-m 5.10.0-0.deb10.16-amd64 #1 SMP Debian 5.10.127-2~bpo10+1 (2022-07-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Feb  5 17:29:02 2023 from 35.235.244.33
agooboy50@lab2-m:~$ ls
stocks
agooboy50@lab2-m:~$ head stocks
ABCSE,B7J,2010-02-08,8.63,8.70,8.57,8.64,78900,8.64
ABCSE,B7J,2010-02-05,8.63,8.71,8.31,8.58,218700,8.58
ABCSE,B7J,2010-02-04,8.88,8.88,8.59,8.66,89900,8.66
ABCSE,B7J,2010-02-03,8.83,8.92,8.80,8.89,119000,8.89
ABCSE,B7J,2010-02-02,8.77,8.90,8.73,8.87,51900,8.87
ABCSE,B7J,2010-02-01,8.69,8.77,8.66,8.75,38600,8.75
ABCSE,B7J,2010-01-29,8.81,8.81,8.56,8.57,91700,8.57
ABCSE,B7J,2010-01-28,8.90,8.90,8.60,8.69,92100,8.69
ABCSE,B7J,2010-01-27,8.87,8.87,8.68,8.79,82400,8.79
ABCSE,B7J,2010-01-26,8.83,8.92,8.71,8.82,106000,8.82
agooboy50@lab2-m:~$ pwd
/home/agooboy50
agooboy50@lab2-m:~$ ls
stocks
agooboy50@lab2-m:~$ hadoop fs -mkdir /BigData
agooboy50@lab2-m:~$ hadoop fs -copyFromLocal stocks /BigData/.
agooboy50@lab2-m:~$ spark-shell
```

Loading the dataset



```
Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_362)
Type in expressions to have them evaluated.
Type :help for more information.

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

import org.apache.spark.sql.types._
val schema = StructType(Array(

         StructField("exch",StringType, true),
         StructField("symbol",StringType, true),
         StructField("ymd",DateType,true),
         StructField("price_open",FloatType,true),
         StructField("price_high",FloatType,true),
         StructField("price_low",FloatType,true),
         StructField("price_close",FloatType,true),
         StructField("volume",IntegerType,true),
         StructField("price_adj_close",FloatType,true)))

val stocks_data = spark
  .read.format("csv")
  .option("header", "true")
  .schema(schema)
  .load("hdfs://10.128.0.2:8020/BigData/stocks")

// Exiting paste mode, now interpreting.

import org.apache.spark.sql.types._
schema: org.apache.spark.sql.types.StructType = StructType(StructField(exch,StringType,true), StructField(symbol,StringType,true), StructField(ymd,DateType,true), StructField(price_op
en,FloatType,true), StructField(price_high,FloatType,true), StructField(price_low,FloatType,true), StructField(price_close,FloatType,true), StructField(volume,IntegerType,true), Struc
tField(price_adj_close,FloatType,true))
stocks_data: org.apache.spark.sql.DataFrame = [exch: string, symbol: string ... 7 more fields]

scala> stocks_data.show(10)
+-----+------+----------+----------+----------+---------+-----------+------+---------------+
| exch|symbol|       ymd|price_open|price_high|price_low|price_close|volume|price_adj_close|
+-----+------+----------+----------+----------+---------+-----------+------+---------------+
|ABCSE|   B7J|2010-02-05|      8.63|      8.71|     8.31|       8.58|218700|           8.58|
|ABCSE|   B7J|2010-02-04|      8.88|      8.88|     8.59|       8.66| 89900|           8.66|
|ABCSE|   B7J|2010-02-03|      8.83|      8.92|      8.8|       8.89|119000|           8.89|
|ABCSE|   B7J|2010-02-02|      8.77|       8.9|     8.73|       8.87| 51900|           8.87|
|ABCSE|   B7J|2010-02-01|      8.69|      8.77|     8.66|       8.75| 38600|           8.75|
|ABCSE|   B7J|2010-01-29|      8.81|      8.81|     8.56|       8.57| 91700|           8.57|
|ABCSE|   B7J|2010-01-28|       8.9|       8.9|      8.6|       8.69| 92100|           8.69|
|ABCSE|   B7J|2010-01-27|      8.87|      8.87|     8.68|       8.79| 82400|           8.79|
|ABCSE|   B7J|2010-01-26|      8.83|      8.92|     8.71|       8.82|106000|           8.82|
|ABCSE|   B7J|2010-01-25|      8.98|       9.0|     8.73|       8.83|131500|           8.83|
+-----+------+----------+----------+----------+---------+-----------+------+---------------+
only showing top 10 rows


scala> 
```

Creating the schema

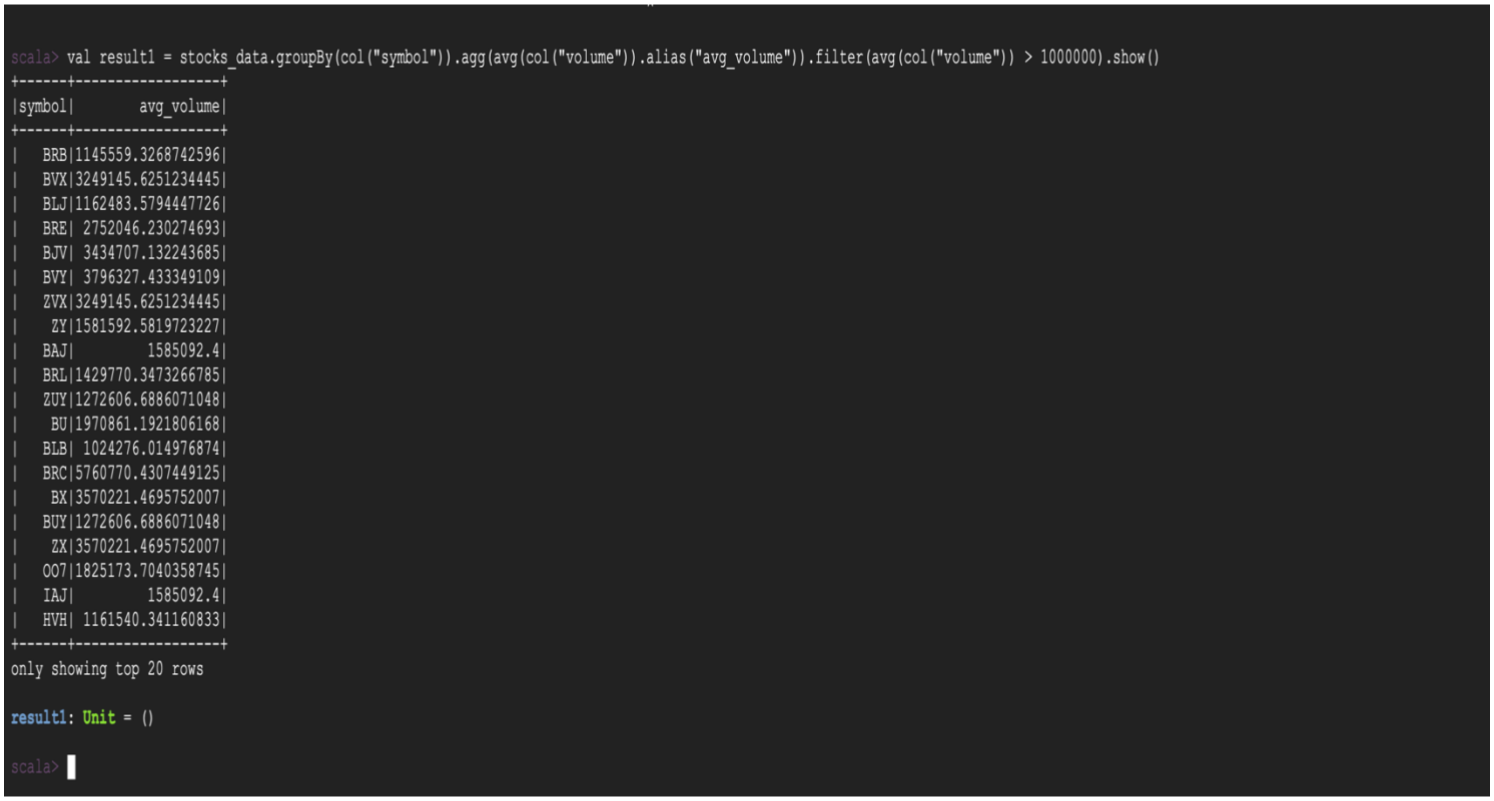
```
import org.apache.spark.sql.types._
val schema = StructType(Array(

                StructField("exch",StringType, true),
                StructField("symbol",StringType, true),
                StructField("ymd",DateType,true),
                StructField("price_open",FloatType,true),
                StructField("price_high",FloatType,true),
                StructField("price_low",FloatType,true),
                StructField("price_close",FloatType,true),
                StructField("volume",IntegerType,true),
                StructField("price_adj_close",FloatType,true)))

val stocks_data = spark
  .read.format("csv")
  .option("header", "true")
  .schema(schema)
  .load("hdfs://10.128.0.2:8020/BigData/stocks")
```

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

val result1 =
stocks_data.groupBy(col("symbol")).agg(avg(col("volume")).alias("avg_volume")).filter(avg(col("
volume")) > 1000000).show()



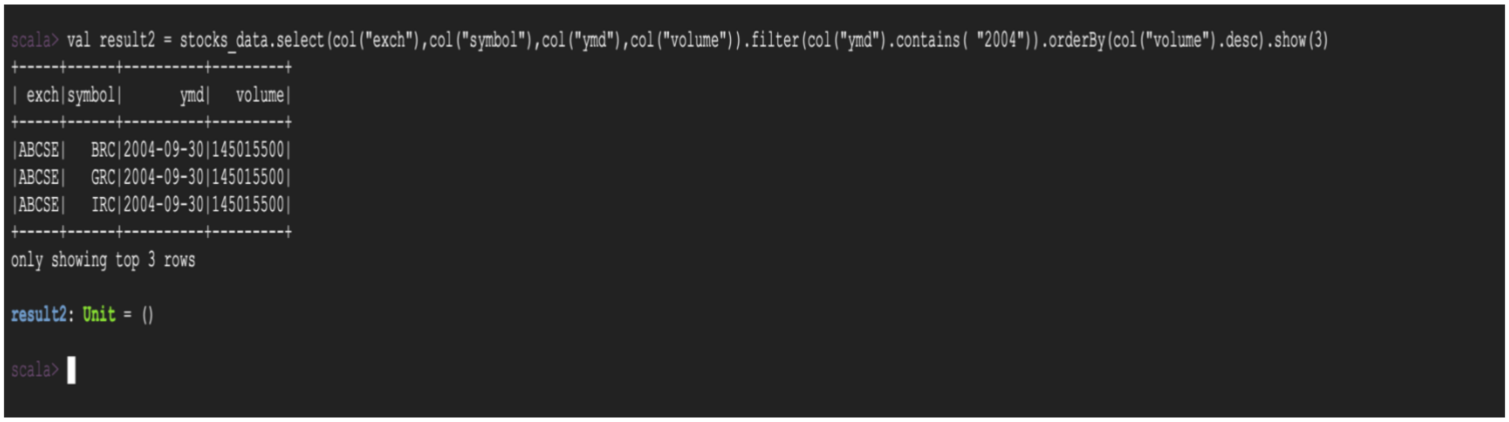
```
scala> val result1 = stocks_data.groupBy(col("symbol")).agg(avg(col("volume")).alias("avg_volume")).filter(avg(col("volume")) > 1000000).show()
+------+------------------+
|symbol|        avg_volume|
+------+------------------+
|   BRB|1145559.3268742596|
|   BVX|3249145.6251234445|
|   BLJ|1162483.5794447726|
|   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

result1: Unit = ()

scala> 
```

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

val result2 =
stocks_data.select(col("exch"),col("symbol"),col("ymd"),col("volume")).filter(col("ymd").contain
s("2004")).orderBy(col("volume").desc).show(3)



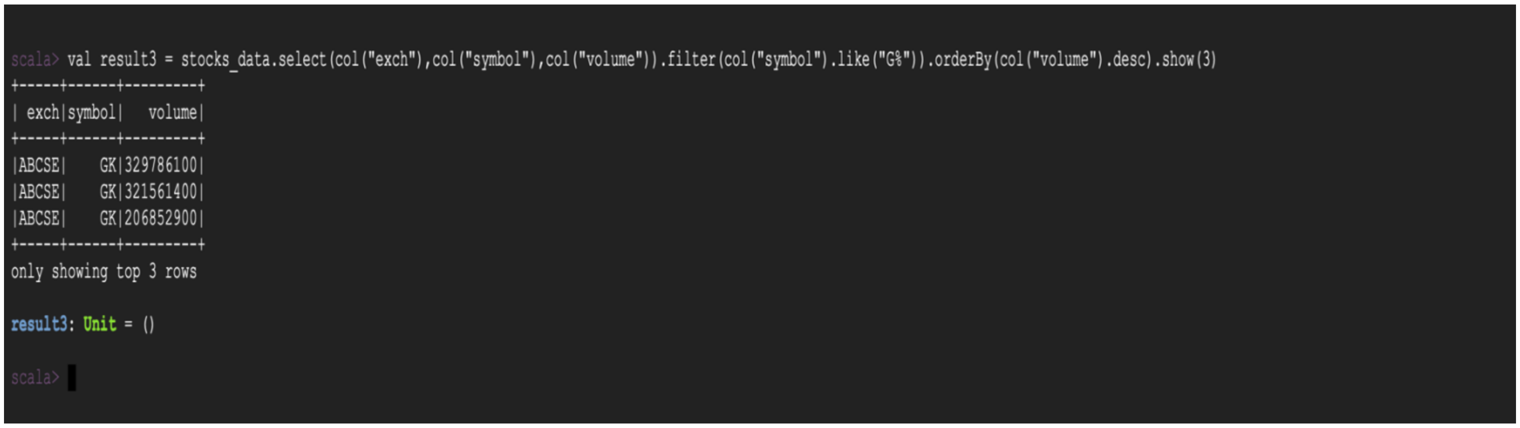
```
scala> val result2 = stocks_data.select(col("exch"),col("symbol"),col("ymd"),col("volume")).filter(col("ymd").contains( "2004")).orderBy(col("volume").desc).show(3)
+-----+------+----------+---------+
| exch|symbol|       ymd|   volume|
+-----+------+----------+---------+
|ABCSE|   BRC|2004-09-30|145015500|
|ABCSE|   GRC|2004-09-30|145015500|
|ABCSE|   IRC|2004-09-30|145015500|
+-----+------+----------+---------+
only showing top 3 rows

result2: Unit = ()

scala> 
```

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 result3 =
stocks_data.select(col("exch"),col("symbol"),col("volume")).filter(col("symbol").like("G%")).orderBy(col("volume").desc).show(3)



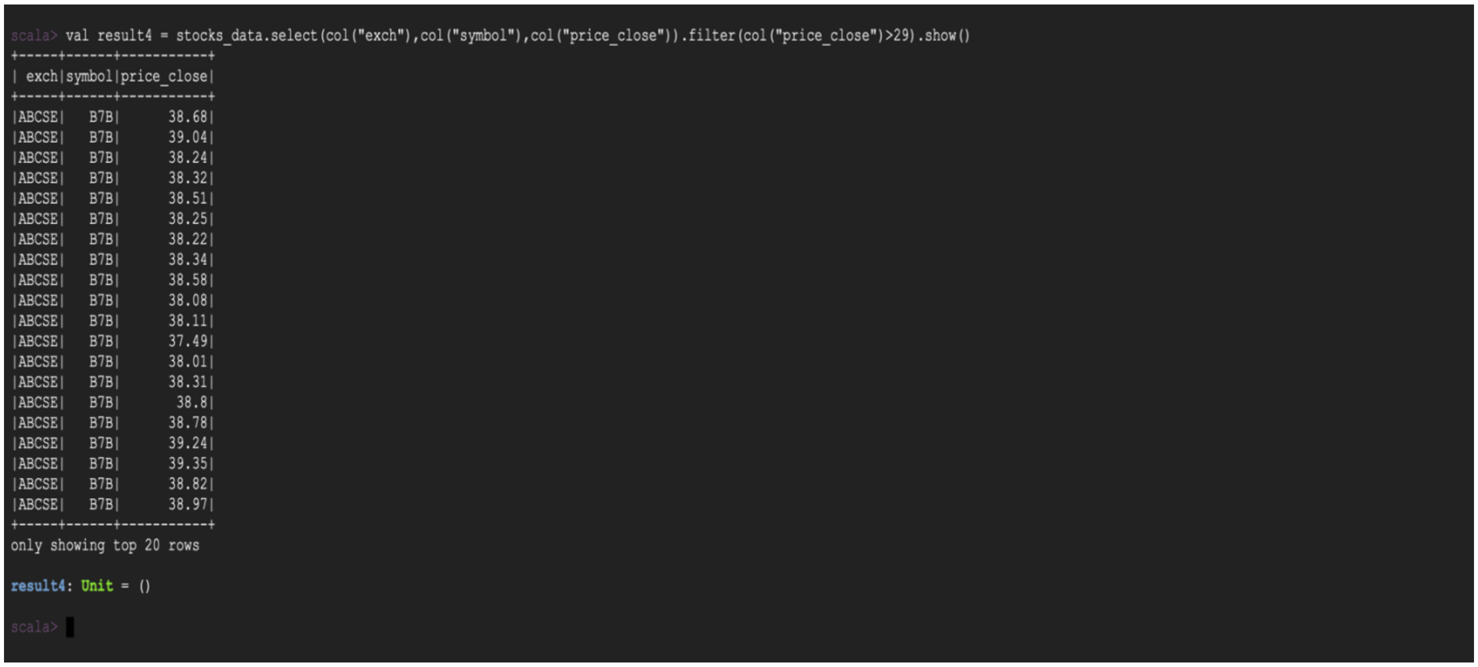
```
scala> val result3 = stocks_data.select(col("exch"),col("symbol"),col("volume")).filter(col("symbol").like("G%")).orderBy(col("volume").desc).show(3)
+-----+------+---------+
| exch|symbol|   volume|
+-----+------+---------+
|ABCSE|    GK|329786100|
|ABCSE|    GK|321561400|
|ABCSE|    GK|206852900|
+-----+------+---------+
only showing top 3 rows

result3: Unit = ()

scala>
```

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

val result4 =
stocks_data.select(col("exch"),col("symbol"),col("price_close")).filter(col("price_close")>29).show()



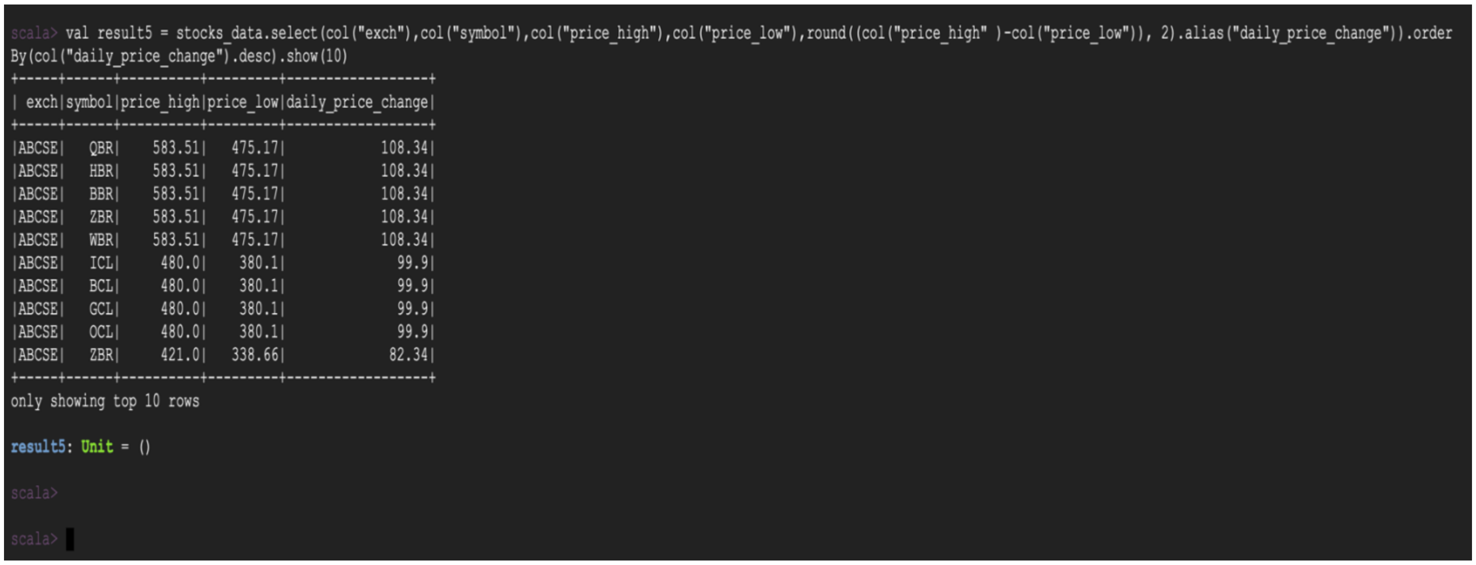
```
scala> val result4 = stocks_data.select(col("exch"),col("symbol"),col("price_close")).filter(col("price_close")>29).show()
+-----+------+-----------+
| exch|symbol|price_close|
+-----+------+-----------+
|ABCSE|   B7B|      38.68|
|ABCSE|   B7B|      39.04|
|ABCSE|   B7B|      38.24|
|ABCSE|   B7B|      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

result4: Unit = ()

scala> 
```

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 result5 =
stocks_data.select(col("exch"),col("symbol"),col("price_high"),col("price_low"),round((col("price_high")-col("price_low")),
2).alias("daily_price_change")).orderBy(col("daily_price_change").desc).show(10)



```
scala> val result5 = stocks_data.select(col("exch"),col("symbol"),col("price_high"),col("price_low"),round((col("price_high" )-col("price_low")), 2).alias("daily_price_change")).order
By(col("daily_price_change").desc).show(10)
+-----+------+----------+---------+------------------+
| exch|symbol|price_high|price_low|daily_price_change|
+-----+------+----------+---------+------------------+
|ABCSE|   QBR|    583.51|   475.17|            108.34|
|ABCSE|   HBR|    583.51|   475.17|            108.34|
|ABCSE|   BBR|    583.51|   475.17|            108.34|
|ABCSE|   ZBR|    583.51|   475.17|            108.34|
|ABCSE|   WBR|    583.51|   475.17|            108.34|
|ABCSE|   ICL|     480.0|    380.1|              99.9|
|ABCSE|   BCL|     480.0|    380.1|              99.9|
|ABCSE|   GCL|     480.0|    380.1|              99.9|
|ABCSE|   OCL|     480.0|    380.1|              99.9|
|ABCSE|   ZBR|     421.0|   338.66|             82.34|
+-----+------+----------+---------+------------------+
only showing top 10 rows

result5: Unit = ()

scala>

scala>
```