07_ProjectExercise

March 29, 2020

1 Spark DataFrames Project Exercise

Let's get some quick practice with your new Spark DataFrame skills, you will be asked some basic questions about some stock market data, in this case Walmart Stock from the years 2012-2017. This exercise will just ask a bunch of questions, unlike the future machine learning exercises, which will be a little looser and be in the form of "Consulting Projects", but more on that later!

For now, just answer the questions and complete the tasks below.

Use the walmart_stock.csv file to Answer and complete the tasks below!

Start a simple Spark Session

[1]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ProjExercise").getOrCreate()
```

Load the Walmart Stock CSV File, have Spark infer the data types.

[2]:
```
path = "Python-and-Spark-for-Big-Data-master/Spark_DataFrame_Project_Exercise/
 ↪walmart_stock.csv"

df = spark.read.csv(path, inferSchema=True, header=True)
df.show(5)
```

```
+-------------------+------------------+---------+---------+------------------+-
-------+------------------+
|               Date|              Open|     High|      Low|             Close|
Volume|         Adj Close|
+-------------------+------------------+---------+---------+------------------+-
-------+------------------+
|2012-01-03 00:00:00|         59.970001|61.060001|59.869999|
60.330002|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.209998999999996|60.349998|59.470001|59.709998999999996|
9593300|         52.078475|
|2012-01-05 00:00:00|         59.349998|59.619999|58.369999|
59.419998|12768200|         51.825539|
|2012-01-06 00:00:00|         59.419998|59.450001|58.869999|              59.0|
8069400|          51.45922|
|2012-01-09 00:00:00|         59.029999|59.549999|58.919998|             59.18|
6679300|51.616215000000004|
+-------------------+------------------+---------+---------+------------------+-
-------+------------------+
only showing top 5 rows
```

What are the column names?

[3]:
```
df.columns
```

[3]:
```
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
```

What does the Schema look like?

[4]:
```
df.printSchema()
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

Print out the first 5 columns.

[5]:
```
df.head(5)
```

[5]:
```
[Row(Date=datetime.datetime(2012, 1, 3, 0, 0), Open=59.970001, High=61.060001,
Low=59.869999, Close=60.330002, Volume=12668800, Adj Close=52.619234999999996),
 Row(Date=datetime.datetime(2012, 1, 4, 0, 0), Open=60.209998999999996,
High=60.349998, Low=59.470001, Close=59.709998999999996, Volume=9593300, Adj
Close=52.078475),
 Row(Date=datetime.datetime(2012, 1, 5, 0, 0), Open=59.349998, High=59.619999,
Low=58.369999, Close=59.419998, Volume=12768200, Adj Close=51.825539),
 Row(Date=datetime.datetime(2012, 1, 6, 0, 0), Open=59.419998, High=59.450001,
Low=58.869999, Close=59.0, Volume=8069400, Adj Close=51.45922),
 Row(Date=datetime.datetime(2012, 1, 9, 0, 0), Open=59.029999, High=59.549999,
Low=58.919998, Close=59.18, Volume=6679300, Adj Close=51.616215000000004)]
```

Use describe() to learn about the DataFrame.

[6]:
```
df.describe().show()
```

```
+-------+------------------+-----------------+-----------------+----------------
-+-----------------+-----------------+
|summary|              Open|              High|               Low|
Close|            Volume|         Adj Close|
+-------+------------------+-----------------+-----------------+----------------
-+-----------------+-----------------+
|  count|              1258|              1258|              1258|
1258|              1258|              1258|
|   mean| 72.35785375357709|72.83938807631165|
71.9186009594594|72.38844998012726|8222093.481717011|67.23883848728146|
| stddev|
6.76809024470826|6.768186808159218|6.744075756255496|6.756859163732991|
4519780.8431556|6.722609449996857|
|    min|56.389998999999996|        57.060001|        56.299999|
56.419998|          2094900|        50.363689|
|    max|         90.800003|        90.970001|            89.25|
90.470001|         80898100|84.91421600000001|
+-------+------------------+-----------------+-----------------+----------------
-+-----------------+-----------------+
```

1.1 Bonus Question!

There are too many decimal places for mean and stddev in the describe() dataframe. Format the numbers to just show up to two decimal places. Pay careful attention to the datatypes that .describe() returns, we didn't cover how to do this exact formatting, but we covered something very similar. **Check this link for a hint** If you get stuck on this, don't worry, just view the solutions.

[7]:
```
df.describe().printSchema()
```

```
root
 |-- summary: string (nullable = true)
 |-- Open: string (nullable = true)
 |-- High: string (nullable = true)
 |-- Low: string (nullable = true)
 |-- Close: string (nullable = true)
 |-- Volume: string (nullable = true)
 |-- Adj Close: string (nullable = true)
```

[8]:
```
result = df.describe()
```

[9]:
```
from pyspark.sql.functions import format_number

new_res = result.select(format_number(result["Open"].cast("float"), 2).
 ↪alias("Open"),
                       format_number(result["High"].cast("float"), 2).
 ↪alias("High"),
                       format_number(result["Low"].cast("float"), 2).
 ↪alias("Low"),
                       format_number(result["Close"].cast("float"), 2).
 ↪alias("Close"),
                       format_number(result["Adj Close"].cast("float"), 2).
 ↪alias("Adj Close"),
                       result["Volume"].cast("int").alias("Volume"))
```

[10]:
```
new_res.show()
```

```
+--------+--------+--------+--------+---------+--------+
|    Open|    High|     Low|   Close|Adj Close|  Volume|
+--------+--------+--------+--------+---------+--------+
|1,258.00|1,258.00|1,258.00|1,258.00| 1,258.00|    1258|
|   72.36|   72.84|   71.92|   72.39|    67.24| 8222093|
|    6.77|    6.77|    6.74|    6.76|     6.72| 4519780|
|   56.39|   57.06|   56.30|   56.42|    50.36| 2094900|
|   90.80|   90.97|   89.25|   90.47|    84.91|80898100|
+--------+--------+--------+--------+---------+--------+
```

Create a new dataframe with a column called HV Ratio that is the ratio of the High Price versus volume of stock traded for a day.

[11]:
```
#df1 = df.withColumn("HV Ratio", df["High"]/df["Volume"])
df.withColumn("HV Ratio", df["High"]/df["Volume"]).select("HV Ratio").show()
```

```
+--------------------+
|            HV Ratio|
+--------------------+
|4.819714653321546E-6|
|6.290848613094555E-6|
|4.669412994783916E-6|
|7.367338463826307E-6|
|8.915604778943901E-6|
|8.644477436914568E-6|
|9.351828421515645E-6|
| 8.29141562102703E-6|
|7.712212102001476E-6|
|7.071764823529412E-6|
|1.015495466386981E-5|
|6.576354146362592...|
| 5.90145296180676E-6|
|8.547679455011844E-6|
|8.420709512685392E-6|
|1.041448341728929...|
|8.316075414862431E-6|
|9.721183814992126E-6|
|8.029436027707578E-6|
|6.307432259386365E-6|
+--------------------+
only showing top 20 rows
```

What day had the Peak High in Price?

[12]:
```
maxhigh = df.agg({"High":"max"}).collect()[0][0]
print(maxhigh)
myrow = df.filter(df["High"]==maxhigh).collect()
print(myrow)
# result
print(myrow[0][0])
print(type(myrow[0][0]))
```

```
90.970001
[Row(Date=datetime.datetime(2015, 1, 13, 0, 0), Open=90.800003, High=90.970001,
Low=88.93, Close=89.309998, Volume=8215400, Adj Close=83.825448)]
2015-01-13 00:00:00
<class 'datetime.datetime'>
```

[13]:
```
# another way is orderby and then take the first row
myrow = df.orderBy("High", ascending=False).head(1)
mydate = myrow[0][0]
print(mydate)
```

```
2015-01-13 00:00:00
```

What is the mean of the Close column?

[14]:
```
from pyspark.sql.functions import mean

close_mean = df.agg({"Close":"mean"}).collect()
close_mean[0][0]
```

[14]:
```
72.38844998012726
```

[15]:
```
# another way
df.select(mean(df["Close"])).show()
```

```
+-----------------+
|       avg(Close)|
+-----------------+
|72.38844998012726|
+-----------------+
```

What is the max and min of the Volume column?

[16]:
```
from pyspark.sql.functions import max, min
```

[17]:
```
df.select(max(df["Volume"]), min(df["Volume"])).show()
```

```
+-----------+-----------+
|max(Volume)|min(Volume)|
+-----------+-----------+
|   80898100|    2094900|
+-----------+-----------+
```

How many days was the Close lower than 60 dollars?

[18]:
```
df2 = df.filter(df["Close"]<60)
df2.show(5)
```

```
+-------------------+------------------+------------------+---------+-----------
-------+--------+------------------+
|               Date|              Open|              High|      Low|
Close|  Volume|         Adj Close|
+-------------------+------------------+------------------+---------+-----------
-------+--------+------------------+
|2012-01-04 00:00:00|60.209998999999996|
60.349998|59.470001|59.709998999999996| 9593300|         52.078475|
|2012-01-05 00:00:00|         59.349998|         59.619999|58.369999|
59.419998|12768200|         51.825539|
|2012-01-06 00:00:00|         59.419998|         59.450001|58.869999|
59.0| 8069400|          51.45922|
|2012-01-09 00:00:00|         59.029999|         59.549999|58.919998|
59.18| 6679300|51.616215000000004|
|2012-01-10 00:00:00|             59.43|59.709998999999996|
58.98|59.040001000000004| 6907300|         51.494109|
+-------------------+------------------+------------------+---------+-----------
-------+--------+------------------+
only showing top 5 rows
```

[19]:
```
df2.count()
```

[19]: 81

What percentage of the time was the High greater than 80 dollars ?

In other words, (Number of Days High>80)/(Total Days in the dataset)

[20]:
```
totalNum = df.count()

num1 = df.filter(df["High"]>80).count()

res = num1 * 100/totalNum
```

```
res
```

[20]:
```
9.141494435612083
```

What is the Pearson correlation between High and Volume?

Hint

[21]:
```
from pyspark.sql.functions import corr

df.select(corr(df["High"], df["Volume"])).show()
#df.select(corr("High", "Volume")).show() # this should work as well
```

```
+-------------------+
| corr(High, Volume)|
+-------------------+
|-0.3384326061737161|
+-------------------+
```

What is the max High per year?

[22]:
```
from pyspark.sql.functions import year

dfyear = df.withColumn("Year", year(df["Date"]))
```

[23]:
```
max_peryear = dfyear.groupBy("Year").max("High")
max_peryear.show()
```

```
+----+---------+
|Year|max(High)|
+----+---------+
|2015|90.970001|
|2013|81.370003|
|2014|88.089996|
|2012|77.599998|
|2016|75.190002|
+----+---------+
```

What is the average Close for each Calendar Month?

In other words, across all the years, what is the average Close price for Jan,Feb, Mar, etc… Your result will have a value for each of these months.

[24]:
```
from pyspark.sql.functions import month, format_number

monthdf = df.withColumn("Month", month("Date")) #note that we can use␣
 ↪df["Date"] or just directly pass in the column nam"Date"
```

```
avgmonth = monthdf.groupBy("Month").mean("Close").orderBy("Month")
avgmonth.show()
```

```
+-----+-----------------+
|Month|       avg(Close)|
+-----+-----------------+
|    1|71.44801958415842|
|    2|  71.306804443299|
|    3|71.77794377570092|
|    4|72.97361900952382|
|    5|72.30971688679247|
|    6| 72.4953774245283|
|    7|74.43971943925233|
|    8|73.02981855454546|
|    9|72.18411785294116|
|   10|71.57854545454543|
|   11| 72.1110893069307|
|   12|72.84792478301885|
+-----+-----------------+
```

[25]:
```
avgmonth_fin = avgmonth.select(["Month", format_number("avg(Close)", 2).
 ↪alias("Avg Close")])
avgmonth_fin.show()
```

```
+-----+---------+
|Month|Avg Close|
+-----+---------+
|    1|    71.45|
|    2|    71.31|
|    3|    71.78|
|    4|    72.97|
|    5|    72.31|
|    6|    72.50|
|    7|    74.44|
|    8|    73.03|
|    9|    72.18|
|   10|    71.58|
|   11|    72.11|
|   12|    72.85|
+-----+---------+
```

2 Great Job!