

Spark Walmart Data Analysis 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

In [2]:

```
1  #import findspark
2  #findspark.init('/home/jubinsoni/spark-2.1.0-bin-hadoop2.7')
3
4  from pyspark.sql import SparkSession
5
6  spark = SparkSession.builder.appName('walmart').getOrCreate()
```

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

In [1]:

```
1  df = spark.read.csv('walmart_stock.csv', inferSchema=True, header=True)
```

What are the column names?

In [3]:

```
1  df.columns
```

Out[3]:

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

What does the Schema look like?

In [4]:

```
1 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.

In [5]:

```
1 for line in df.head(5):
2     print(line, '\n')
```

```
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=60.209998999999999, 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.

In [6]:

```
1 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
stddev	6.76809024470826	6.768186808159218	6.744075756255496	6.756859163732991
min	56.389998999999996	57.060001	56.299999	56.419998
max	90.800003	90.970001	89.25	90.470001

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](http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.Column.cast) (<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.Column.cast>)

If you get stuck on this, don't worry, just view the solutions.

In [7]:

```
1  '''
2  from pyspark.sql.types import (StructField, StringType,
3                                IntegerType, StructType)
4
5  data_schema = [StructField('summary', StringType(), True),
6                  StructField('Open', StringType(), True),
7                  StructField('High', StringType(), True),
8                  StructField('Low', StringType(), True),
9                  StructField('Close', StringType(), True),
10                 StructField('Volume', StringType(), True),
11                 StructField('Adj Close', StringType(), True)
12                 ]
13
14  final_struct = StructType(fields=data_schema)
15
16  '''
17  df = spark.read.csv('walmart_stock.csv', inferSchema=True, header=True)
18
19  df.printSchema()
20  #The schema given below is wrong, as it is mostly from an older version.
21  #Spark is able to predict the schema correctly now
```

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)
```

In [19]:

```
1  from pyspark.sql.functions import format_number
2
3  summary = df.describe()
4  summary.select(summary['summary'],
5                  format_number(summary['Open'].cast('float'), 2).alias('Ope
6                  format_number(summary['High'].cast('float'), 2).alias('Hig
7                  format_number(summary['Low'].cast('float'), 2).alias('Low'
8                  format_number(summary['Close'].cast('float'), 2).alias('Cl
9                  format_number(summary['Volume'].cast('int'), 0).alias('Volu
10 ).show()
```

summary	Open	High	Low	Close	Volume
count	1,258.00	1,258.00	1,258.00	1,258.00	1,258
mean	72.36	72.84	71.92	72.39	8,222,093
stddev	6.77	6.77	6.74	6.76	4,519,780
min	56.39	57.06	56.30	56.42	2,094,900
max	90.80	90.97	89.25	90.47	80,898,100

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.

In [20]:

```
1 df_hv = df.withColumn('HV Ratio', df['High']/df['Volume']).select(['HV Ratio'])
2 df_hv.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?

In [10]:

```
1 df.orderBy(df['High'].desc()).select(['Date']).head(1)[0]['Date']
```

Out[10]:

datetime.datetime(2015, 1, 13, 0, 0)

What is the mean of the Close column?

In [11]:

```
1  from pyspark.sql.functions import mean
2
3  df.select(mean('Close')).show()
```

+	-----+
	avg(Close)
+	-----+
	72.38844998012726
+	-----+

What is the max and min of the Volume column?

In [12]:

```
1  from pyspark.sql.functions import min, max
2
3  df.select(max('Volume'),min('Volume')).show()
```

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

How many days was the Close lower than 60 dollars?

In [13]:

```
1  df.filter(df['Close'] < 60).count()
```

Out[13]:

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)

In [14]:

```
1 df.filter('High > 80').count() * 100/df.count()
```

Out[14]:

9.141494435612083

What is the Pearson correlation between High and Volume?

Hint

<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrameStatFunction>

In [15]:

```
1 df.corr('High', 'Volume')
```

Out[15]:

-0.3384326061737161

In [16]:

```
1 from pyspark.sql.functions import corr
2
3 df.select(corr(df['High'], df['Volume'])).show()
```

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

What is the max High per year?

In [17]:

```
1  from pyspark.sql.functions import (dayofmonth, hour,
2                                     dayofyear, month,
3                                     year, weekofyear,
4                                     format_number, date_format)
5
6  year_df = df.withColumn('Year', year(df['Date']))
7
8  year_df.groupBy('Year').max()['Year', 'max(High)'].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.

In [18]:

```
1  #Create a new column Month from existing Date column
2  month_df = df.withColumn('Month', month(df['Date']))
3
4  #Group by month and take average of all other columns
5  month_df = month_df.groupBy('Month').mean()
6
7  #Sort by month
8  month_df = month_df.orderBy('Month')
9
10 #Display only month and avg(Close), the desired columns
11 month_df['Month', 'avg(Close)'].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|
+-----+-----+
```

Thank you!

Spark Walmart Data Analysis 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

In [2]:

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

In [1]:

What are the column names?

In [3]:

Out[3]:

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

What does the Schema look like?

In [4]:

```
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.

In [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=60.209998999999999, 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=51.6162150000000004)
```

Use describe() to learn about the DataFrame.

In [6]:

summary				
Open	High	Low	Close	Volume
1258	1258	1258	1258	1258
mean	72.35785375357709	72.83938807631165	71.9186009594594	72.38844998012726
stddev	6.76809024470826	6.768186808159218	6.744075756255496	6.756859163732991
min	56.389998999999996	57.060001	56.299999	56.419998
max	90.800003	90.970001	89.25	90.470001

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](http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.Column.cast) (<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.Column.cast>)

If you get stuck on this, don't worry, just view the solutions.

In [7]:

```
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)
```

In [8]:

summary	Open	High	Low	Close	Volume
count	1,258.00	1,258.00	1,258.00	1,258.00	1,258
mean	72.36	72.84	71.92	72.39	8,222,093
stddev	6.77	6.77	6.74	6.76	4,519,780
min	56.39	57.06	56.30	56.42	2,094,900
max	90.80	90.97	89.25	90.47	80,898,100

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.

In [9]:

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...
...

What day had the Peak High in Price?

In [10]:

Out[10]:

datetime.datetime(2015, 1, 13, 0, 0)

What is the mean of the Close column?

In [11]:

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

What is the max and min of the Volume column?

In [12]:

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

How many days was the Close lower than 60 dollars?

In [13]:

Out[13]:

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)

In [14]:

Out[14]:

9.141494435612083

What is the Pearson correlation between High and Volume?

Hint

<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrameStatFunction>

In [15]:

Out[15]:

-0.3384326061737161

In [16]:

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

What is the max High per year?

In [17]:

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.

In [18]:

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

Thank you!

