

Basic Operations in PySpark Using a Sample Dataset

Loading PySpark

```
In [105]: import findspark
findspark.init('/home/donis/spark-2.1.1-bin-hadoop2.7')

from pyspark.sql import SparkSession

from pyspark.sql.functions import format, number, year
```

```
In [106]: spark = SparkSession.builder.appName('PySpark Sample').getOrCreate()
```

We will be working with historical stock market data for Walmart:

```
In [107]: walmart_data = spark.read.csv('walmart_stock.csv', header=True, inferSchema=True)
```

We can run several actions to get a general understanding of the data:

```
In [108]: walmart_data.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)
```

```
In [109]: for entry in walmart_data.head(3):
          print(entry)
          print('\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=60.209998999999996, 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)


```

```
In [110]: description = walmart_data.describe()
```

```
In [111]: description.select(description['summary'],
format_number(description['Open'].cast('float'), 2).alias('Open'),
format_number(description['High'].cast('float'), 2).alias('High'),
format_number(description['Low'].cast('float'), 2).alias('Low'),
format_number(description['Close'].cast('float'), 2).alias('Close'),
format_number(description['Volume'].cast('float'), 2).alias('Volume'),
format_number(description['Adj Close'].cast('float'), 2).alias('Adjusted Close'))
```

summary	Open	High	Low	Close	Volume	Adjusted Close
count	1,258.00	1,258.00	1,258.00	1,258.00	1,258.00	1,258.00
mean	72.36	72.84	71.92	72.39	8,222,093.50	67.24
stddev	6.77	6.77	6.74	6.76	4,519,781.00	6.72
min	56.39	57.06	56.30	56.42	2,094,900.00	50.36
max	90.80	90.97	89.25	90.47	80,898,096.00	84.91

The historical volatility ratio can be calculated:

```
In [112]: walmart_data = walmart_data.withColumn("HV Ratio", walmart_data['High']/walmart_data['Low'])
```

```
In [113]: walmart_data.select(format_number(walmart_data['HV Ratio'], 10).alias('HV Ratio'))
```

HV Ratio
0.0000048197
0.0000062908
0.0000046694
0.0000073673
0.0000089156
0.0000086445
0.0000093518
0.0000082914
0.0000077122
0.0000070718
0.0000101550
0.0000065764
0.0000059015
0.0000085477
0.0000084207
0.0000104145
0.0000083161
0.0000097212
0.0000080294
0.0000063074

only showing top 20 rows

We can also find the days in which the stock closed at a higher price than it opened:

```
In [114]: stock_increase = walmart_data.filter(walmart_data['Open'] < walmart_data['Close'])
```

```
In [115]: stock_increase = stock_increase.select(['Date',
          format_number(stock_increase['Open'],2).alias('Open'),
          format_number(stock_increase['Close'],2).alias('Close')]).
stock_increase.show()
```

```
+-----+-----+-----+
|          Date| Open|Close|
+-----+-----+-----+
|2012-01-03 00:00:...| 59.97|60.33|
|2012-01-05 00:00:...| 59.35|59.42|
|2012-01-09 00:00:...| 59.03|59.18|
|2012-01-11 00:00:...| 59.06|59.40|
|2012-01-13 00:00:...| 59.18|59.54|
|2012-01-18 00:00:...| 59.79|60.01|
|2012-01-19 00:00:...| 59.93|60.61|
|2012-01-20 00:00:...| 60.75|61.01|
|2012-01-23 00:00:...| 60.81|60.91|
|2012-01-24 00:00:...| 60.75|61.39|
|2012-01-25 00:00:...| 61.18|61.47|
|2012-01-30 00:00:...| 60.47|61.30|
|2012-02-01 00:00:...| 61.79|62.18|
|2012-02-06 00:00:...| 61.85|61.88|
|2012-02-07 00:00:...| 61.62|61.69|
|2012-02-09 00:00:...| 61.58|61.96|
|2012-02-10 00:00:...| 61.68|61.90|
|2012-02-14 00:00:...| 61.91|62.22|
|2012-02-16 00:00:...| 61.77|62.04|
|2012-02-17 00:00:...| 62.32|62.48|
+-----+-----+-----+
only showing top 20 rows
```

And finally, we can figure out what the maximum stock price was each year:

```
In [128]: years = walmart_data.withColumn('Year', year('Date'))
years = years.orderBy('Year')
years.groupBy('Year').max().select(['Year', format_number('max(High)', 2).alia
```

```
+-----+-----+
|Year|Maximum Value|
+-----+-----+
|2012|          77.60|
|2013|          81.37|
|2014|          88.09|
|2015|          90.97|
|2016|          75.19|
+-----+-----+
```