# 06_DateTime

March 29, 2020

### 0.1 Load data

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

[2]:
```
spark = SparkSession.builder.appName("DateTime").getOrCreate()
```

[3]:
```
path = "Python-and-Spark-for-Big-Data-master/Spark_DataFrames/appl_stock.csv"
```

[4]:
```
df = spark.read.csv(path, inferSchema=True, header=True)
```

[5]:
```
df.show(6)
```

```
+-------------------+------------------+----------+------------------+----------
--------+---------+------------------+
|               Date|              Open|      High|               Low|
Close|   Volume|         Adj Close|
+-------------------+------------------+----------+------------------+----------
--------+---------+------------------+
|2010-01-04 00:00:00|        213.429998|214.499996|212.38000099999996|
214.009998|123432400|         27.727039|
|2010-01-05 00:00:00|        214.599998|215.589994|        213.249994|
214.379993|150476200|27.774976000000002|
|2010-01-06 00:00:00|        214.379993|    215.23|        210.750004|
210.969995|138040000|27.333178000000004|
|2010-01-07 00:00:00|            211.75|212.000006|        209.050005|
210.58|119282800|          27.28265|
|2010-01-08 00:00:00|
210.299994|212.000006|209.06000500000002|211.98000499999998|111902700|
27.464034|
|2010-01-11 00:00:00|212.79999700000002|213.000002|
208.450005|210.11000299999998|115557400|         27.221758|
+-------------------+------------------+----------+------------------+----------
--------+---------+------------------+
only showing top 6 rows
```

[6]:
```
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)
```

[7]:
```
# if we take the first row, we'll see that the first column is in datetime␣
 ↪format
df.head(1)
```

[7]:
```
[Row(Date=datetime.datetime(2010, 1, 4, 0, 0), Open=213.429998, High=214.499996,
Low=212.38000099999996, Close=214.009998, Volume=123432400, Adj
Close=27.727039)]
```

### 0.2 Basic functions for working with datetime

[8]:
```
from pyspark.sql.functions import (dayofmonth, dayofyear, hour,
                                   month,year, weekofyear,
                                  format_number, date_format)
```

[9]:
```
# get day of month
df.select(dayofmonth(df["Date"])).show(5)
```

```
+----------------+
|dayofmonth(Date)|
+----------------+
|               4|
|               5|
|               6|
|               7|
|               8|
+----------------+
only showing top 5 rows
```

[10]:
```
# get hour
df.select(hour(df["Date"])).show(5)
```

```
+----------+
|hour(Date)|
+----------+
|         0|
|         0|
```

```
|         0|
|         0|
|         0|
+----------+
only showing top 5 rows
```

[11]:
```
# get month
df.select(month(df["Date"])).show(5)
```

```
+-----------+
|month(Date)|
+-----------+
|          1|
|          1|
|          1|
|          1|
|          1|
+-----------+
only showing top 5 rows
```

### 0.3 Get averaged value per year

This should be similar to a lot of tasks we have done for JMA

[12]:
```
# first add a column of year
newdf = df.withColumn("Year", year(df["Date"]))   #interesting that here we␣
 ↪don't need to use select() method
```

[13]:
```
newdf.show(5)
```

```
+-------------------+----------+----------+------------------+------------------
+---------+------------------+----+
|               Date|      Open|      High|               Low|
Close|   Volume|         Adj Close|Year|
+-------------------+----------+----------+------------------+------------------
+---------+------------------+----+
|2010-01-04 00:00:00|213.429998|214.499996|212.38000099999996|
214.009998|123432400|         27.727039|2010|
|2010-01-05 00:00:00|214.599998|215.589994|        213.249994|
214.379993|150476200|27.774976000000002|2010|
|2010-01-06 00:00:00|214.379993|    215.23|        210.750004|
210.969995|138040000|27.333178000000004|2010|
|2010-01-07 00:00:00|    211.75|212.000006|        209.050005|
210.58|119282800|          27.28265|2010|
|2010-01-08
00:00:00|210.299994|212.000006|209.06000500000002|211.98000499999998|111902700|
```

```
27.464034|2010|
+------------------+----------+----------+-----------------+------------------
+---------+-----------------+----+
only showing top 5 rows
```

[14]:
```
# groupby year and calculate mean
results = newdf.groupBy("Year").mean()
results.show(3)
```

```
+----+-----------------+-----------------+-----------------+----------------
-+------------------+-----------------+---------+
|Year|         avg(Open)|         avg(High)|          avg(Low)|
avg(Close)|        avg(Volume)|    avg(Adj Close)|avg(Year)|
+----+-----------------+-----------------+-----------------+----------------
-+------------------+-----------------+---------+
|2015|120.17575393253965|121.24452385714291|
118.8630954325397|120.03999980555547| 5.18378869047619E7|115.96740080555561|
2015.0|
|2013| 473.1281355634922| 477.6389272301587|468.24710264682557|
472.6348802857143|          1.016087E8| 62.61798788492063|   2013.0|
|2014| 295.1426195357143|297.56103184523823| 292.9949599801587|
295.4023416507935|6.315273055555555E7| 87.63583323809523|   2014.0|
+----+-----------------+-----------------+-----------------+----------------
-+------------------+-----------------+---------+
only showing top 3 rows
```

[15]:
```
# if we only interested in average close value
final = results.select(["Year", "avg(Close)"])
```

[16]:
```
final.show(5)
```

```
+----+------------------+
|Year|        avg(Close)|
+----+------------------+
|2015|120.03999980555547|
|2013| 472.6348802857143|
|2014| 295.4023416507935|
|2012| 576.0497195640002|
|2016|104.60400786904763|
+----+------------------+
only showing top 5 rows
```

[17]:
```
# format the number
```

```
finalnew = final.select(["Year", format_number("avg(Close)", 2).alias("Avg␣
↪Close")])
```

[18]:
```
finalnew1 = finalnew.orderBy("Year")
```

[19]:
```
finalnew1.show()
```

```
+----+---------+
|Year|Avg Close|
+----+---------+
|2010|   259.84|
|2011|   364.00|
|2012|   576.05|
|2013|   472.63|
|2014|   295.40|
|2015|   120.04|
|2016|   104.60|
+----+---------+
```