app development

September 26, 2019

make this notebook executable with arguments by papermill

[43]:
```
ISIN = 'DE000A0H08M3'
```

initialize Spark environment

[1]:
```
import findspark
findspark.init()

import pyspark
```

[2]:
```
from pyspark.sql import SparkSession

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

download part of online data set to local machine

[42]:
```
! task sync.files
```

```
aws s3 sync s3://deutsche-boerse-xetra-pds data --exclude "*" --include
"*2019-09*" --exclude "*$(date +"%Y-%m-%d")*"
```

read whole data set

[14]:
```
data = spark.read.format("csv").option("header", "true").option("inferSchema",␣
↪"true").load("./data/*")
```

[15]:
```
data.printSchema()
```

```
root
 |-- ISIN: string (nullable = true)
 |-- Mnemonic: string (nullable = true)
 |-- SecurityDesc: string (nullable = true)
 |-- SecurityType: string (nullable = true)
 |-- Currency: string (nullable = true)
 |-- SecurityID: integer (nullable = true)
 |-- Date: timestamp (nullable = true)
 |-- Time: string (nullable = true)
 |-- StartPrice: double (nullable = true)
 |-- MaxPrice: double (nullable = true)
```

```
 |-- MinPrice: double (nullable = true)
 |-- EndPrice: double (nullable = true)
 |-- TradedVolume: integer (nullable = true)
 |-- NumberOfTrades: integer (nullable = true)
```

[16]:
```
data.groupBy(data.Date).count().orderBy(data.Date).show(35)
```

```
+-------------------+-----+
|               Date|count|
+-------------------+-----+
|2018-01-02 00:00:00|68154|
|2018-01-03 00:00:00|70608|
|2019-09-02 00:00:00|65223|
|2019-09-03 00:00:00|69577|
|2019-09-04 00:00:00|67250|
|2019-09-05 00:00:00|73780|
|2019-09-06 00:00:00|70237|
|2019-09-08 00:00:00| 2652|
|2019-09-09 00:00:00|66905|
|2019-09-10 00:00:00|74799|
|2019-09-11 00:00:00|75005|
|2019-09-12 00:00:00|80295|
|2019-09-13 00:00:00|69910|
|2019-09-16 00:00:00|69910|
|2019-09-17 00:00:00|71067|
|2019-09-18 00:00:00|64271|
|2019-09-19 00:00:00|69321|
|2019-09-20 00:00:00|69135|
|2019-09-21 00:00:00|   26|
|2019-09-22 00:00:00| 2657|
|2019-09-23 00:00:00|68893|
|2019-09-24 00:00:00|67511|
+-------------------+-----+
```

use Spark SQL to apply Analytic Functions

[17]:
```
data.createOrReplaceTempView("data")
```

choose an ISIN from the data exploration example

[44]:
```
# removed in favour of papermill parameters
# ISIN = 'DE000A0H08M3'
```

step 1/2 - prepare result set

[31]:
```
spark.sql(f"""
SELECT
    ISIN,
    Date,
    Time,
    StartPrice,
    First_value(StartPrice) OVER ( partition BY Date ORDER BY Time) AS␣
 ↪FirstStartPrice,
    EndPrice,
    Last_value(EndPrice) OVER ( partition BY Date ORDER BY Time) AS␣
 ↪LastEndPrice,
    TradedVolume
FROM data
WHERE ISIN = '{ISIN}'
-- AND Date = '2018-01-03'
-- AND TIME BETWEEN '08:00' AND '09:00'
""").createOrReplaceTempView("tmp")

spark.sql("SELECT * FROM tmp").show()
```

```
+------------+-------------------+-----+----------+---------------+--------+----
--------+------------+
|        ISIN|               Date|
Time|StartPrice|FirstStartPrice|EndPrice|LastEndPrice|TradedVolume|
+------------+-------------------+-----+----------+---------------+--------+----
--------+------------+
|DE000A0H08M3|2019-09-03 00:00:00|04:00|    30.565|         30.565|  30.565|
30.565|           0|
|DE000A0H08M3|2019-09-03 00:00:00|07:04|    30.545|         30.565|  30.545|
30.545|          65|
|DE000A0H08M3|2019-09-03 00:00:00|07:11|     30.48|         30.565|   30.48|
30.48|         155|
|DE000A0H08M3|2019-09-03 00:00:00|07:25|    30.475|         30.565|  30.475|
30.475|         300|
|DE000A0H08M3|2019-09-03 00:00:00|07:31|    30.495|         30.565|  30.495|
30.495|         522|
|DE000A0H08M3|2019-09-03 00:00:00|07:51|     30.48|         30.565|   30.48|
30.48|         964|
|DE000A0H08M3|2019-09-03 00:00:00|07:53|    30.465|         30.565|  30.465|
30.465|         350|
|DE000A0H08M3|2019-09-03 00:00:00|07:55|    30.465|         30.565|  30.465|
30.465|         234|
|DE000A0H08M3|2019-09-03 00:00:00|08:05|     30.48|         30.565|   30.48|
30.48|         155|
|DE000A0H08M3|2019-09-03 00:00:00|08:14|    30.505|         30.565|  30.505|
30.505|         820|
|DE000A0H08M3|2019-09-03 00:00:00|08:37|    30.395|         30.565|  30.395|
```

```
30.395|         150|
|DE000A0H08M3|2019-09-03 00:00:00|08:38|     30.41|         30.565|   30.41|
30.41|          80|
|DE000A0H08M3|2019-09-03 00:00:00|08:54|     30.39|         30.565|   30.39|
30.39|         201|
|DE000A0H08M3|2019-09-03 00:00:00|09:02|    30.425|         30.565|  30.425|
30.425|        1439|
|DE000A0H08M3|2019-09-03 00:00:00|09:20|     30.42|         30.565|  30.415|
30.415|         160|
|DE000A0H08M3|2019-09-03 00:00:00|09:26|     30.39|         30.565|   30.39|
30.39|           5|
|DE000A0H08M3|2019-09-03 00:00:00|09:58|    30.305|         30.565|  30.305|
30.305|        1206|
|DE000A0H08M3|2019-09-03 00:00:00|10:25|     30.35|         30.565|   30.35|
30.35|         500|
|DE000A0H08M3|2019-09-03 00:00:00|10:30|     30.34|         30.565|   30.34|
30.34|          16|
|DE000A0H08M3|2019-09-03 00:00:00|11:12|     30.28|         30.565|   30.28|
30.28|           0|
+------------+-------------------+-----+----------+---------------+--------+----
--------+------------+
only showing top 20 rows
```

step 2/2 - aggregate result set

[32]:
```
spark.sql("""
SELECT
    ISIN,
    Date,
    FirstStartPrice AS OpeningPrice,
    last(LastEndPrice) AS ClosingPrice,
    sum(TradedVolume) AS DailyTradedVolume,
    format_number(last(LastEndPrice) / Lag(last(LastEndPrice)) OVER (partition␣
 ↪BY ISIN ORDER BY Date) *100 -100,2) AS PctChgPrvCls
FROM tmp
GROUP BY 1, 2, 3""").createOrReplaceTempView("result_set")

spark.sql("SELECT * FROM result_set").show(35)
```

```
+------------+-------------------+-----------+-----------+-----------------+--
----------+
|        ISIN|
Date|OpeningPrice|ClosingPrice|DailyTradedVolume|PctChgPrvCls|
+------------+-------------------+-----------+-----------+-----------------+--
----------+
|DE000A0H08M3|2018-01-02 00:00:00|       31.61|       31.6|           433588|
null|
```

```
|DE000A0H08M3|2018-01-03 00:00:00|      31.855|      32.075|           251182|
1.50|
|DE000A0H08M3|2019-09-02 00:00:00|       30.54|      30.565|            47694|
-4.71|
|DE000A0H08M3|2019-09-03 00:00:00|      30.565|       30.29|            12431|
-0.90|
|DE000A0H08M3|2019-09-04 00:00:00|       30.29|       30.66|            38723|
1.22|
|DE000A0H08M3|2019-09-05 00:00:00|       30.66|      31.195|            28348|
1.74|
|DE000A0H08M3|2019-09-06 00:00:00|      31.195|       30.85|            39453|
-1.11|
|DE000A0H08M3|2019-09-08 00:00:00|       30.85|       30.85|                0|
0.00|
|DE000A0H08M3|2019-09-09 00:00:00|      31.085|       31.23|            13059|
1.23|
|DE000A0H08M3|2019-09-10 00:00:00|       31.23|      31.865|            62853|
2.03|
|DE000A0H08M3|2019-09-11 00:00:00|      31.865|      31.835|            40772|
-0.09|
|DE000A0H08M3|2019-09-12 00:00:00|      31.835|       31.42|            69554|
-1.30|
|DE000A0H08M3|2019-09-13 00:00:00|       31.42|      31.555|            67551|
0.43|
|DE000A0H08M3|2019-09-16 00:00:00|      32.495|       32.47|           190072|
2.90|
|DE000A0H08M3|2019-09-17 00:00:00|       32.47|       32.23|           104333|
-0.74|
|DE000A0H08M3|2019-09-18 00:00:00|       32.23|      32.335|           151389|
0.33|
|DE000A0H08M3|2019-09-19 00:00:00|      32.335|      32.485|           214190|
0.46|
|DE000A0H08M3|2019-09-20 00:00:00|      32.485|       32.82|           536527|
1.03|
|DE000A0H08M3|2019-09-22 00:00:00|       32.82|       32.82|                0|
0.00|
|DE000A0H08M3|2019-09-23 00:00:00|      32.925|      32.595|           123195|
-0.69|
|DE000A0H08M3|2019-09-24 00:00:00|      32.595|      32.195|            45309|
-1.23|
+------------+-------------------+------------+------------+-----------------+--
----------+
```

save result set to disk

[36]:
```
spark.sql("SELECT * FROM result_set") \
.repartition(1) \
```

```
.write \
.mode ("overwrite") \
.format("csv") \
.option("header", "true") \
.save("output")
```

[40]:
```
! task clean.output
```

```
rm -f output/_SUCCESS
rm -f output/.*.crc
mv -u output/*.csv output/result_set.csv || true
mv: 'output/result_set.csv' and 'output/result_set.csv' are the same file
```

[ ]:
```

```