pySpark_local_flights_data_1

July 5, 2021

[48]:
```
import findspark
findspark.init()

import pyspark
import random

from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *

from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer,␣
↪MinMaxScaler
```

[49]:
```
#
import os , sys
import multiprocessing
cpu_cores = multiprocessing.cpu_count()
print("-this server has these many CPU CORES --- >> \n",cpu_cores)
print("  "*90)
# SOURCE -- https://psutil.readthedocs.io/en/latest/
import psutil

dict_virual_mem = dict(psutil.virtual_memory()._asdict())
print(dict_virual_mem)
print("  "*90)
free_memory = dict(psutil.virtual_memory()._asdict())['free']
free_memory = int(free_memory)/1024.0/1024.0/1024.0
print(" this server has these GIGA BYTES of FREE MEMORY --- >> \n",free_memory)
print("  "*90)
# number_cores = int(os.environ['NUM_CPUS'])
# memory_gb = int(os.environ['AVAILABLE_MEMORY_MB']) // 1024
# print(" this server has these many CPU CORES --- >> ",number_cores)
# print(" this server has these GIGA BYTES of MEMORY --- >> ",memory_gb)
```

```
-this server has these many CPU CORES --- >>
 8
```

```
{'total': 33607561216, 'available': 26518085632, 'percent': 21.1, 'used':
6757040128, 'free': 21324050432, 'active': 8022589440, 'inactive': 2980438016,
'buffers': 1206226944, 'cached': 4320243712, 'shared': 517824512, 'slab':
931041280}

 this server has these GIGA BYTES of FREE MEMORY --- >>
 19.859569549560547
```

[50]:
```
number_cores = 6
memory_gb = 5
conf = (
    pyspark.SparkConf()
        .setMaster('local[{}]'.format(number_cores))
        .set('spark.driver.memory', '{}g'.format(memory_gb))
)
spark_context_local_multiCore = pyspark.SparkContext(conf=conf)
#spark_context_local = pyspark.SparkContext('local[*]')
#local-cluster[<n>,<c>,<m>] # SparkShell Command
# Check the URL -- http://localhost:4040/
```

[51]:
```
# Viewing all configured parameters
print(dict(spark_context_local_multiCore.getConf().getAll()))
```

```
{'spark.driver.host': '192.168.1.2', 'spark.rdd.compress': 'True',
'spark.app.startTime': '1625462427091', 'spark.app.id': 'local-1625462427152',
'spark.serializer.objectStreamReset': '100', 'spark.submit.pyFiles': '',
'spark.executor.id': 'driver', 'spark.submit.deployMode': 'client',
'spark.driver.port': '41713', 'spark.driver.memory': '5g',
'spark.ui.showConsoleProgress': 'true', 'spark.master': 'local[6]',
'spark.app.name': 'pyspark-shell'}
```

[74]:
```
import time
start = time.time()

rdd = spark_context_local_multiCore.parallelize([1, 4, 9])
#print(type(rdd)) # <class 'pyspark.rdd.RDD'>

sum_squares = rdd.map(lambda elem: float(elem)**2).sum()
print(sum_squares)
end = time.time()
print("time taken for above code block ----\n",end - start)
```

```
98.0
time taken for above code block ----
 0.03461098670959473
```

[47]:
```
#spark_context_local.stop() # Manually STOP the - spark_context_local
spark_context_local_multiCore.stop() # Manually STOP the - spark_context_local_multiCore
#sc.stop() # Manually STOP the - sc
# Check the URL -- http://localhost:4040/
```

[61]:
```
# SparkSession.builder. -- BUILDER PATTERN -- https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html
"""
SparkSession is a wrapper around SparkContext and SQLContext , which was directly
used for constructing DataFrame s in the versions prior to Spark 2.0. The Builder
object lets you specify master, appName , and other configuration options, but the
defaults will do.
"""

spark_session1 = SparkSession.builder.appName('pySpark_test_app').getOrCreate()
# Seen below - the above line Doesnt change the APP NAME -- 'spark.app.name': 'pyspark-shell'

#spark_session2 = SparkSession.builder.master("local[*]").getOrCreate()
print("   "*90)
print(type(spark_session1)) #<class 'pyspark.sql.session.SparkSession'>
spark_session1.version # 3.1.1
print("   "*90)
print(spark_session1.sparkContext.getConf().getAll())
print("   "*90)
print(dict(spark_session1.sparkContext.getConf().getAll()))
# Does the above DIFFER from what we did 2 Cells above with --print(dict(spark_context_local_multiCore.getConf().getAll()))
# Check the URL -- http://localhost:4040/
```

```
<class 'pyspark.sql.session.SparkSession'>

[('spark.sql.warehouse.dir', 'file:/home/dhankar/temp/0521/pySpark_june21/GitUp_
PySpark_June21/GitUp/pySpark_jun21/spark-warehouse'), ('spark.driver.host',
'192.168.1.2'), ('spark.app.id', 'local-1625462427152'), ('spark.executor.id',
'driver'), ('spark.driver.port', '41713'), ('spark.driver.memory', '5g'),
('spark.master', 'local[6]'), ('spark.app.name', 'pyspark-shell'),
('spark.rdd.compress', 'True'), ('spark.app.startTime', '1625462427091'),
('spark.serializer.objectStreamReset', '100'), ('spark.submit.pyFiles', ''),
('spark.submit.deployMode', 'client'), ('spark.ui.showConsoleProgress', 'true')]

{'spark.sql.warehouse.dir': 'file:/home/dhankar/temp/0521/pySpark_june21/GitUp_P
```

```
ySpark_June21/GitUp/pySpark_jun21/spark-warehouse', 'spark.driver.host':
'192.168.1.2', 'spark.app.id': 'local-1625462427152', 'spark.executor.id':
'driver', 'spark.driver.port': '41713', 'spark.driver.memory': '5g',
'spark.master': 'local[6]', 'spark.app.name': 'pyspark-shell',
'spark.rdd.compress': 'True', 'spark.app.startTime': '1625462427091',
'spark.serializer.objectStreamReset': '100', 'spark.submit.pyFiles': '',
'spark.submit.deployMode': 'client', 'spark.ui.showConsoleProgress': 'true'}
```

[75]:
```
print(type(spark_session1.read)) #<pyspark.sql.readwriter.DataFrameReader at␣
↪0x7ff7c5b63820>
print("   "*90)
dir(spark_session1.read)
print("   "*90)
#dir(spark_session1.read.__dict__)
print("   "*90)
```

```
<class 'pyspark.sql.readwriter.DataFrameReader'>
```

[3]:

[13]:

[76]:
```
start = time.time()

# below code is OK with -- spark_session1 -- But not with -- spark_context_local
df_flights = spark_session1.read.csv('raw-flight-data.csv', header=True,␣
↪inferSchema=True)

print("    "*90)

# below code is OK with -- spark_context_local - we will create a -- sqlContext
# below code can error out with --
# from pyspark.sql import SQLContext
# sqlContext = SQLContext(spark_context_local)

# df_flights = sqlContext.read.csv('raw-flight-data.csv', header=True,
#                      inferSchema=True)
print(type(df_flights)) # <class 'pyspark.sql.dataframe.DataFrame'>
print("--- "*10)
print(df_flights.count()) # 27,19,418 ROWS
print("--- "*10)
print(df_flights.columns)
print("--- "*10)
print(df_flights.printSchema())
```

```
print("--- "*10)
print(df_flights.head(3))
print("--- "*10)
df_flights.describe().show()
print("--- "*10)
df_flights.describe().show(vertical=True)
print("--- "*10)

end = time.time()
print("time taken for above code block ----\n",end - start)
```

```
<class 'pyspark.sql.dataframe.DataFrame'>
--- --- --- --- --- --- --- --- --- ---
2719418
--- --- --- --- --- --- --- --- --- ---
['DayofMonth', 'DayOfWeek', 'Carrier', 'OriginAirportID', 'DestAirportID',
'DepDelay', 'ArrDelay']
--- --- --- --- --- --- --- --- --- ---
root
 |-- DayofMonth: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- Carrier: string (nullable = true)
 |-- OriginAirportID: integer (nullable = true)
 |-- DestAirportID: integer (nullable = true)
 |-- DepDelay: integer (nullable = true)
 |-- ArrDelay: integer (nullable = true)

None
--- --- --- --- --- --- --- --- --- ---
[Row(DayofMonth=19, DayOfWeek=5, Carrier='DL', OriginAirportID=11433,
DestAirportID=13303, DepDelay=-3, ArrDelay=1), Row(DayofMonth=19, DayOfWeek=5,
Carrier='DL', OriginAirportID=14869, DestAirportID=12478, DepDelay=0,
ArrDelay=-8), Row(DayofMonth=19, DayOfWeek=5, Carrier='DL',
OriginAirportID=14057, DestAirportID=14869, DepDelay=-4, ArrDelay=-15)]
--- --- --- --- --- --- --- --- --- ---
+-------+-----------------+------------------+-------+-----------------+-------
-----------+-----------------+------------------+
|summary|       DayofMonth|         DayOfWeek|Carrier|   OriginAirportID|
DestAirportID|         DepDelay|          ArrDelay|
+-------+-----------------+------------------+-------+-----------------+-------
-----------+-----------------+------------------+
|  count|          2719418|           2719418|2719418|           2719418|
2719418|          2691974|           2690385|
|   mean|15.79747468024408|3.8983907586108497|   null|
12742.26441172339|12742.455345592329|10.53686662649788|  6.63768791455498|
| stddev|8.799860168985422|1.9859881390373262|   null|1501.9729397025571|
```

```
1501.969252892782|36.09952806643139|38.648814893900735|
|    min|                1|                1|     9E|             10140|
10140|             -63|               -94|
|    max|               31|                7|     YV|             15376|
15376|            1863|             1845|
+-------+-----------------+-----------------+-------+------------------+-------
-----------+-----------------+------------------+

--- --- --- --- --- --- --- --- --- ---
-RECORD 0-----------------------------
 summary         | count
 DayofMonth      | 2719418
 DayOfWeek       | 2719418
 Carrier         | 2719418
 OriginAirportID | 2719418
 DestAirportID   | 2719418
 DepDelay        | 2691974
 ArrDelay        | 2690385
-RECORD 1-----------------------------
 summary         | mean
 DayofMonth      | 15.79747468024408
 DayOfWeek       | 3.8983907586108497
 Carrier         | null
 OriginAirportID | 12742.26441172339
 DestAirportID   | 12742.455345592329
 DepDelay        | 10.53686662649788
 ArrDelay        | 6.63768791455498
-RECORD 2-----------------------------
 summary         | stddev
 DayofMonth      | 8.799860168985422
 DayOfWeek       | 1.9859881390373262
 Carrier         | null
 OriginAirportID | 1501.9729397025571
 DestAirportID   | 1501.969252892782
 DepDelay        | 36.09952806643139
 ArrDelay        | 38.648814893900735
-RECORD 3-----------------------------
 summary         | min
 DayofMonth      | 1
 DayOfWeek       | 1
 Carrier         | 9E
 OriginAirportID | 10140
 DestAirportID   | 10140
 DepDelay        | -63
 ArrDelay        | -94
-RECORD 4-----------------------------
 summary         | max
 DayofMonth      | 31
```

```
 DayOfWeek       | 7
 Carrier         | YV
 OriginAirportID | 15376
 DestAirportID   | 15376
 DepDelay        | 1863
 ArrDelay        | 1845

--- --- --- --- --- --- --- --- --- ---
time taken for above code block ----
 7.774825811386108
```

[53]:
```
# Display One Col  -- Carrier
df_flights.select("Carrier").show(5)
# Display One Col  -- DepDelay
df_flights.select("DepDelay").show(5)
df_flights.select("DepDelay").show(5,vertical=True)
```

```
+-------+
|Carrier|
+-------+
|     DL|
|     DL|
|     DL|
|     DL|
|     DL|
+-------+
only showing top 5 rows

+--------+
|DepDelay|
+--------+
|      -3|
|       0|
|      -4|
|      28|
|      -6|
+--------+
only showing top 5 rows

-RECORD 0-------
 DepDelay | -3
-RECORD 1-------
 DepDelay | 0
-RECORD 2-------
 DepDelay | -4
-RECORD 3-------
 DepDelay | 28
-RECORD 4-------
```

```
 DepDelay | -6
only showing top 5 rows
```

[54]:
```
# Lower Case All text in a Column
from pyspark.sql.functions import lower
df_carrier_lower = df_flights.select(lower(col("Carrier")).
 ↪alias("Carrier_lower"))
print(type(df_carrier_lower)) #<class 'pyspark.sql.dataframe.DataFrame'>
print(df_carrier_lower.show(5))
```

```
<class 'pyspark.sql.dataframe.DataFrame'>
+-------------+
|Carrier_lower|
+-------------+
|           dl|
|           dl|
|           dl|
|           dl|
|           dl|
+-------------+
only showing top 5 rows

None
```

[55]:
```
# Add a Column with Literal Values --- Same Values in All Rows
from pyspark.sql.functions import col, lit, when
df_carrier_lower2 = df_carrier_lower.select(col("Carrier_lower"),lit("22").
 ↪alias("dummy_col_lit"))
print(df_carrier_lower2.show(5))
```

```
+-------------+-------------+
|Carrier_lower|dummy_col_lit|
+-------------+-------------+
|           dl|           22|
|           dl|           22|
|           dl|           22|
|           dl|           22|
|           dl|           22|
+-------------+-------------+
only showing top 5 rows

None
```

[56]:
```
# get values coded in a new Column when both -DepDelay and ArrDelay -- are␣
 ↪Positive
```

```
df_flights2 = df_flights.withColumn("col_coded", when((col("DepDelay") >=1) &␣
 ↪(col("ArrDelay") >=1),lit("1")).otherwise(lit("0")))
print(df_flights2.show(5))
```

```
+----------+---------+-------+--------------+-------------+--------+--------+--
-------+
|DayofMonth|DayOfWeek|Carrier|OriginAirportID|DestAirportID|DepDelay|ArrDelay|co
l_coded|
+----------+---------+-------+--------------+-------------+--------+--------+--
-------+
|        19|        5|     DL|         11433|        13303|      -3|       1|
0|
|        19|        5|     DL|         14869|        12478|       0|      -8|
0|
|        19|        5|     DL|         14057|        14869|      -4|     -15|
0|
|        19|        5|     DL|         15016|        11433|      28|      24|
1|
|        19|        5|     DL|         11193|        12892|      -6|     -11|
0|
+----------+---------+-------+--------------+-------------+--------+--------+--
-------+
only showing top 5 rows

None
```

[ ]:
```
# at this stage we should look at the DAG and details for various JOBS in the␣
 ↪UI
#http://localhost:4040/jobs/job/?id=1
#http://localhost:4040/jobs/job/?id=2
#http://localhost:4040/jobs/job/?id=3
#
```

[ ]:
```
#  spark.mllib package VS.  spark.ml package
#
# As of Spark 2.0, the RDD-based APIs in the spark.mllib package have entered␣
 ↪maintenance mode. The primary Machine Learning API for Spark is now the␣
 ↪DataFrame-based API in the spark.ml package.

Source - https://spark.apache.org/docs/latest/ml-guide.html
```

[ ]:
```
# map -- is a transformation
```

[ ]:
```
#reduce -- is an action
```

[80]:
```
# Create a view or table
```

```
temp_table_name = "flights_csv_table_view"

df_flights.createOrReplaceTempView(temp_table_name)
```

[82]:
```
# SQL
spark_session1.sql("""SHOW COLUMNS from flights_csv_table_view""").show()
spark_session1.sql("""select count(*) from flights_csv_table_view""").show()
```

```
+---------------+
|       col_name|
+---------------+
|     DayofMonth|
|      DayOfWeek|
|        Carrier|
|OriginAirportID|
|  DestAirportID|
|       DepDelay|
|       ArrDelay|
+---------------+

+--------+
|count(1)|
+--------+
| 2719418|
+--------+
```

[77]:
```
print(spark_session1.read.__getattribute__)
print(spark_session1.read.parquet)
print("  "*90)
print("----get help for the --- spark_session1.read.parquet?-------")
print("  "*90)
spark_session1.read.parquet?
print("  "*90)
```

```
<method-wrapper '__getattribute__' of DataFrameReader object at 0x7f1633f60ca0>
<bound method DataFrameReader.parquet of <pyspark.sql.readwriter.DataFrameReader
object at 0x7f1633f44b20>>

----get help for the --- spark_session1.read.parquet?-------


Signature: spark_session1.read.
parquet(*paths,
**options)
Docstring:
Loads Parquet files, returning the result as a :class:`DataFrame`.
```

```
.. versionadded:: 1.4.0

Parameters
----------
paths : str

Other Parameters
----------------
mergeSchema : str or bool, optional
    sets whether we should merge schemas collected from all
    Parquet part-files. This will override
    ``spark.sql.parquet.mergeSchema``. The default value is specified in
    ``spark.sql.parquet.mergeSchema``.
pathGlobFilter : str or bool, optional
    an optional glob pattern to only include files with paths matching
    the pattern. The syntax follows `org.apache.hadoop.fs.GlobFilter`.
    It does not change the behavior of
    `partition discovery <https://spark.apache.org/docs/latest/sql-data-sources-
parquet.html#partition-discovery>`_.  # noqa
recursiveFileLookup : str or bool, optional
    recursively scan a directory for files. Using this option
    disables
    `partition discovery <https://spark.apache.org/docs/latest/sql-data-sources-
parquet.html#partition-discovery>`_.  # noqa

    modification times occurring before the specified time. The provided
timestamp
    must be in the following format: YYYY-MM-DDTHH:mm:ss (e.g.
2020-06-01T13:00:00)
modifiedBefore (batch only) : an optional timestamp to only include files with
    modification times occurring before the specified time. The provided
timestamp
    must be in the following format: YYYY-MM-DDTHH:mm:ss (e.g.
2020-06-01T13:00:00)
modifiedAfter (batch only) : an optional timestamp to only include files with
    modification times occurring after the specified time. The provided
timestamp
    must be in the following format: YYYY-MM-DDTHH:mm:ss (e.g.
2020-06-01T13:00:00)

Examples
--------
>>> df = spark.read.parquet('python/test_support/sql/parquet_partitioned')
>>> df.dtypes
[('name', 'string'), ('year', 'int'), ('month', 'int'), ('day', 'int')]
File:      /opt/spark/python/pyspark/sql/readwriter.py
Type:      method
```

[78]:
```
## generic HELP from the Spark DOCS with a ? at the end
spark_session1.read?
```

```
Type:        property
String form: <property object at 0x7f16393fa180>
Docstring:
Returns a :class:`DataFrameReader` that can be used to read data
in as a :class:`DataFrame`.

.. versionadded:: 2.0.0

Returns
-------
:class:`DataFrameReader`
```

[81]:
```

```

```
+---------------+
|       col_name|
+---------------+
|     DayofMonth|
|      DayOfWeek|
|        Carrier|
|OriginAirportID|
|  DestAirportID|
|       DepDelay|
|       ArrDelay|
+---------------+

+--------+
|count(1)|
+--------+
| 2719418|
+--------+
```

[27]:
```
import time
start = time.time()


from pyspark.sql.functions import count
df_flights.write.format("parquet").mode("overwrite").
 ↪partitionBy("OriginAirportID").save("/tmp/flights_parquet")
flights_parquet = spark_session1.read.format("parquet").load("/tmp/
 ↪flights_parquet")
print(type(flights_parquet)) #<class 'pyspark.sql.dataframe.DataFrame'>
#
```

```
display(flights_parquet)
print(flights_parquet.head(3))
print("--- "*10)
flights_parquet.describe().show()
print("--- "*10)
#display(flights_parquet.filter("DayOfWeek = 1").
 ↪groupBy("DayofMonth","OriginAirportID").agg(count("*").
 ↪alias("TotalFlights")).orderBy("TotalFlights", ascending=False).limit(20))
## TODO -- Use -- https://docs.python.org/3/library/timeit.html

end = time.time()
print("time taken for above code block ----\n",end - start)
#5.808667421340942
#5.463586807250977
# TODO - geta average time taken

# TODO -- get time with a -  Databricks Delta table --- https://docs.
 ↪azuredatabricks.net/_static/notebooks/delta/optimize-python.html
```

```
<class 'pyspark.sql.dataframe.DataFrame'>
DataFrame[DayofMonth: int, DayOfWeek: int, Carrier: string, DestAirportID: int, DepDelay: int,


[Row(DayofMonth=4, DayOfWeek=2, Carrier='YV', DestAirportID=11057, DepDelay=-4,
ArrDelay=-18, OriginAirportID=10397), Row(DayofMonth=4, DayOfWeek=2,
Carrier='YV', DestAirportID=12264, DepDelay=2, ArrDelay=-17,
OriginAirportID=10397), Row(DayofMonth=4, DayOfWeek=2, Carrier='YV',
DestAirportID=13930, DepDelay=-1, ArrDelay=-20, OriginAirportID=10397)]
--- --- --- --- --- --- --- --- --- ---
+-------+-----------------+------------------+-------+------------------+-------
-----------+-----------------+------------------+
|summary|       DayofMonth|         DayOfWeek|Carrier|     DestAirportID|
DepDelay|         ArrDelay|   OriginAirportID|
+-------+-----------------+------------------+-------+------------------+-------
-----------+-----------------+------------------+
|  count|          2719418|           2719418|2719418|           2719418|
2691974|          2690385|           2719418|
|   mean|15.79747468024408|3.8983907586108497|   null|12742.455345592329|
10.53686662649788| 6.63768791455498| 12742.26441172339|
| stddev|8.799860168985404|1.9859881390373395|   null|
1501.969252892776|36.099528066431404|38.64881489390083|1501.9729397025776|
|    min|                1|                 1|     9E|             10140|
-63|             -94|             10140|
|    max|               31|                 7|     YV|             15376|
1863|             1845|             15376|
+-------+-----------------+------------------+-------+------------------+-------
-----------+-----------------+------------------+
```

```
--- --- --- --- --- --- --- --- --- ---
time taken for above code block ----
 5.463586807250977
```

[28]:
```
# Train / Test Split - 70% / 30%

splits = flights_parquet.randomSplit([0.7, 0.3])
print(type(splits))
print("--- "*10)

train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
train_rows = train.count()
test_rows = test.count()
print("Training Rows:", train_rows, " Testing Rows:", test_rows)
#Training Rows: 1903609  Testing Rows: 815809
print("--- "*10)
```

```
<class 'list'>
--- --- --- --- --- --- --- --- --- ---
Training Rows: 1903609  Testing Rows: 815809
--- --- --- --- --- --- --- --- --- ---
```

[12]:
```
data_for_model = df_flights.select("DayofMonth", "DayOfWeek",␣
 ↪"OriginAirportID", "DestAirportID", "DepDelay", col("ArrDelay").
 ↪alias("label"))
print(type(data_for_model))
print(data_for_model.head(3)) # Notice the LABEL given to the Column - ArrDelay
```

```
<class 'pyspark.sql.dataframe.DataFrame'>
[Row(DayofMonth=19, DayOfWeek=5, OriginAirportID=11433, DestAirportID=13303,
DepDelay=-3, label=1), Row(DayofMonth=19, DayOfWeek=5, OriginAirportID=14869,
DestAirportID=12478, DepDelay=0, label=-8), Row(DayofMonth=19, DayOfWeek=5,
OriginAirportID=14057, DestAirportID=14869, DepDelay=-4, label=-15)]
```

[5]:
```
from pyspark.sql.functions import mean
df_walmart.select(mean("Open")).show()
df_walmart.select(mean("High")).show()
df_walmart.select(mean("Low")).show()
df_walmart.select(mean("Close")).show()
```

```
+-----------------+
|        avg(Open)|
+-----------------+
|72.35785375357709|
+-----------------+
```

```
+-----------------+
|        avg(High)|
+-----------------+
|72.83938807631165|
+-----------------+

+----------------+
|        avg(Low)|
+----------------+
|71.9186009594594|
+----------------+

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

[26]:
```
print(df_walmart.sort("High").explain())
print(df_walmart.sort("Low").explain())
```

```
== Physical Plan ==
*(1) Sort [High#677 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(High#677 ASC NULLS FIRST, 200),
ENSURE_REQUIREMENTS, [id=#383]
   +- FileScan csv [Date#675,Open#676,High#677,Low#678,Close#679,Volume#680,Adj
Close#681] Batched: false, DataFilters: [], Format: CSV, Location: InMemoryFileI
ndex[file:/home/dhankar/temp/0521/pySpark_june21/GitUp_PySpark_June21/GitUp/pySp
ark_…, PartitionFilters: [], PushedFilters: [], ReadSchema: struct<Date:string
,Open:double,High:double,Low:double,Close:double,Volume:int,Adj Close:double>


None
== Physical Plan ==
*(1) Sort [Low#678 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(Low#678 ASC NULLS FIRST, 200),
ENSURE_REQUIREMENTS, [id=#395]
   +- FileScan csv [Date#675,Open#676,High#677,Low#678,Close#679,Volume#680,Adj
Close#681] Batched: false, DataFilters: [], Format: CSV, Location: InMemoryFileI
ndex[file:/home/dhankar/temp/0521/pySpark_june21/GitUp_PySpark_June21/GitUp/pySp
ark_…, PartitionFilters: [], PushedFilters: [], ReadSchema: struct<Date:string
,Open:double,High:double,Low:double,Close:double,Volume:int,Adj Close:double>


None
```

[ ]:
```
from pyspark.sql.functions import size , split
df_walmart.select(size(split(col("High"), " "))).show(2)
```

[ ]:
```

```

[ ]:
```

```