* Data is available in HDFS file system under /public/crime/csv
* Structure of data (ID,Case Number,Date,Block,IUCR,Primary Type,Description,Location Description,Arrest,Domestic,Beat,District,Ward,Community Area,FBI Code,X Coordinate,Y Coordinate,Year,Updated On,Latitude,Longitude,Location)  
  File format - text file
* Delimiter - “,” (use regex while splitting split(",(?=(?:[^\"]\*\"[^\"]\*\")\*[^\"]\*$)", -1), as there are some fields with comma and enclosed using double quotes.
* Get top 3 crime types based on number of incidents in RESIDENCE area using “Location Description”
* Store the result in HDFS path /user/<YOUR\_USER\_ID>/solutions/solution03/RESIDENCE\_AREA\_CRIMINAL\_TYPE\_DATA
* Output Fields: Crime Type, Number of Incidents
* Output File Format: JSON
* Output Delimiter: N/A
* Output Compression: No

>>> crime = sc.textFile("/public/crime/csv")

>>> for i in crime.take(10): print(i)

...

>>> crimeHeader = crime.first()

crimeNoHeader = crime.filter(lambda f: f != crimeHeader)

>>> crimeFile = crime.filter(lambda f: (f != crimeHeader) & (f.split(",(?=(?:[^\"]\*\"[^\"]\*\")\*[^\"]\*$)", -1)[7] == 'RESIDENCE') )

>>> for i in crimeFile.take(10): print(i)

...

[Stage 4:> (0 + 1) / 1]20/09/21 10:07:34 WARN TaskSetManager: Lost task 0.0 in stage 4.0 (TID 10, wn04.itversity.com, executor 8): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/hdp/current/spark2-client/python/pyspark/worker.py", line 229, in main

process()

File "/usr/hdp/current/spark2-client/python/pyspark/worker.py", line 224, in process

serializer.dump\_stream(func(split\_index, iterator), outfile)

File "/usr/hdp/current/spark2-client/python/pyspark/serializers.py", line 372, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/hdp/current/spark2-client/python/pyspark/rdd.py", line 1354, in takeUpToNumLeft

yield next(iterator)

File "<stdin>", line 1, in <lambda>

<https://stackoverflow.com/questions/53384200/regular-expression-pyspark-dataframe-column>

Regular expression

Import re

>>> crimeFile = crime.filter(lambda f: (f != crimeHeader) & (re.split(",(?=(?:[^\"]\*\"[^\"]\*\")\*[^\"]\*$)", f)[7] == 'RESIDENCE') )

>>> for i in crimeFile.take(10): print(i)

...

5679864,HN487195,07/24/2007 02:20:00 PM,034XX N MILWAUKEE AVE,0486,BATTERY,DOMESTIC BATTERY SIMPLE,RESIDENCE,false,true,1731,017,30,16,08B,1148204,1922490,2007,04/15/2016 08:55:02 AM,41.943259605,-87.730682304,"(41.943259605, -87.730682304)"

5679866,HN489706,07/26/2007 03:00:00 AM,048XX W ARGYLE ST,1305,CRIMINAL DAMAGE,CRIMINAL DEFACEMENT,RESIDENCE,false,false,1623,016,45,12,14,1143169,1932807,2007,04/15/2016 08:55:02 AM,41.971665985,-87.74893007,"(41.971665985, -87.74893007)"

5679889,HN485186,07/23/2007 11:15:00 PM,003XX E 89TH ST,0486,BATTERY,DOMESTIC BATTERY SIMPLE,RESIDENCE,true,true,0633,006,6,44,08B,1179726,1846039,2007,04/15/2016 08:55:02 AM,41.732806071,-87.617166805,"(41.732806071, -87.617166805)"

5679895,HN489348,07/25/2007 11:00:00 AM,040XX S LAKE PARK AVE,5002,OTHER OFFENSE,OTHER VEHICLE OFFENSE,RESIDENCE,false,false,2123,002,4,36,26,1183624,1878499,2007,04/15/2016 08:55:02 AM,41.821789385,-87.601875318,"(41.821789385, -87.601875318)"

5679914,HN488460,07/25/2007 04:00:00 PM,064XX S KNOX AVE,1780,OFFENSE INVOLVING CHILDREN,OTHER OFFENSE,RESIDENCE,true,true,0813,008,13,64,26,1146486,1861530,2007,04/15/2016 08:55:02 AM,41.77601059,-87.738549072,"(41.77601059, -87.738549072)"

5679916,HN485355,07/24/2007 02:45:00 AM,076XX S ABERDEEN ST,0610,BURGLARY,FORCIBLE ENTRY,RESIDENCE,true,false,0612,006,17,71,05,1170307,1854152,2007,04/15/2016 08:55:02 AM,41.755279135,-87.651437311,"(41.755279135, -87.651437311)"

5679921,HN489154,07/25/2007 10:35:00 PM,053XX S BISHOP ST,1310,CRIMINAL DAMAGE,TO PROPERTY,RESIDENCE,false,false,0933,009,16,61,14,1167576,1869236,2007,04/15/2016 08:55:02 AM,41.796730504,-87.661013549,"(41.796730504, -87.661013549)"

5679925,HN486629,07/24/2007 06:00:00 PM,122XX S THROOP ST,1310,CRIMINAL DAMAGE,TO PROPERTY,RESIDENCE,false,false,0524,005,34,53,14,1169883,1823564,2007,04/15/2016 08:55:02 AM,41.671350221,-87.653875744,"(41.671350221, -87.653875744)"

5679930,HN489670,07/25/2007 09:00:00 AM,012XX N LA SALLE DR,1150,DECEPTIVE PRACTICE,CREDIT CARD FRAUD,RESIDENCE,false,false,1821,018,43,8,11,1174910,1908568,2007,04/15/2016 08:55:02 AM,41.904499908,-87.632942733,"(41.904499908, -87.632942733)"

5679935,HN488333,07/25/2007 02:46:00 PM,016XX W ERIE ST,1310,CRIMINAL DAMAGE,TO PROPERTY,RESIDENCE,false,false,1324,012,1,24,14,1165327,1904420,2007,04/15/2016 08:55:02 AM,41.893326808,-87.668261706,"(41.893326808, -87.668261706)"

>>> crimeFileMap = crimeFile.map(lambda cf: (cf.split(",")[5], 1))

>>> for i in crimeFileMap.take(10): print(i)

...

(u'BATTERY', 1)

(u'CRIMINAL DAMAGE', 1)

(u'BATTERY', 1)

(u'OTHER OFFENSE', 1)

(u'OFFENSE INVOLVING CHILDREN', 1)

(u'BURGLARY', 1)

(u'CRIMINAL DAMAGE', 1)

(u'CRIMINAL DAMAGE', 1)

(u'DECEPTIVE PRACTICE', 1)

(u'CRIMINAL DAMAGE', 1)

>>> crimeFileReduce = crimeFileMap(lambda x, y: x + y)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'PipelinedRDD' object is not callable

>>> crimeFileReduce = crimeFileMap.reduceByKey(lambda x, y: x + y)

>>> for i in crimeFileReduce.take(10): print(i)

...

(u'OTHER NARCOTIC VIOLATION', 2)

(u'GAMBLING', 107)

(u'DECEPTIVE PRACTICE', 64478)

(u'ARSON', 2007)

(u'WEAPONS VIOLATION', 7259)

(u'KIDNAPPING', 1606)

(u'CRIM SEXUAL ASSAULT', 8089)

(u'STALKING', 842)

(u'MOTOR VEHICLE THEFT', 2612)

(u'HOMICIDE', 1)

>>> crimeFileSorted = crimeFileReduce.sortByKey(ascending = False)

>>> type(crimeFileSorted)

<class 'pyspark.rdd.PipelinedRDD'>

>>> for i in crimeFileSorted.take(10): print(i)

...

(u'WEAPONS VIOLATION', 7259)

(u'THEFT', 142273)

(u'STALKING', 842)

(u'SEX OFFENSE', 5628)

(u'ROBBERY', 4613)

(u'RITUALISM', 5)

(u'PUBLIC PEACE VIOLATION', 4660)

(u'PUBLIC INDECENCY', 1)

(u'PROSTITUTION', 588)

(u'OTHER OFFENSE', 184667)

>>> crimeFilesorted = crimeFileReduce.map(lambda rm: (rm[1], (rm[0], rm[1])))

>>> for i in crimeFilesorted.take(10): print(i)

...

(2, (u'OTHER NARCOTIC VIOLATION', 2))

(107, (u'GAMBLING', 107))

(64478, (u'DECEPTIVE PRACTICE', 64478))

(2007, (u'ARSON', 2007))

(7259, (u'WEAPONS VIOLATION', 7259))

(1606, (u'KIDNAPPING', 1606))

(8089, (u'CRIM SEXUAL ASSAULT', 8089))

(2612, (u'MOTOR VEHICLE THEFT', 2612))

(842, (u'STALKING', 842))

(1, (u'HOMICIDE', 1))

>>> crimeFileSortedReal = crimeFilesorted.sortByKey(ascending = False)

>>> for i in crimeFileSortedReal.take(10): print(i)

...

(244394, (u'BATTERY', 244394))

(184667, (u'OTHER OFFENSE', 184667))

(142273, (u'THEFT', 142273))

(139492, (u'CRIMINAL DAMAGE', 139492))

(127833, (u'BURGLARY', 127833))

(67997, (u'ASSAULT', 67997))

(64478, (u'DECEPTIVE PRACTICE', 64478))

(26446, (u'NARCOTICS', 26446))

(23041, (u'CRIMINAL TRESPASS', 23041))

(22740, (u'OFFENSE INVOLVING CHILDREN', 22740))

>>> crimeFileSortedReal[:3]

>>> crimeFileSortRequired = crimeFileSortedReal.map(lambda cf: cf[1][0] +"," + cf[1][1])

>>> for i in crimeFileSortRequired.take(10): print(i)

...

[Stage 17:> (0 + 1) / 1]20/09/21 13:04:41 WARN TaskSetManager: Lost task 0.0 in stage 17.0 (TID 78, wn05.itversity.com, executor 11): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/hdp/current/spark2-client/python/pyspark/worker.py", line 229, in main

process()

File "/usr/hdp/current/spark2-client/python/pyspark/worker.py", line 224, in process

serializer.dump\_stream(func(split\_index, iterator), outfile)

File "/usr/hdp/current/spark2-client/python/pyspark/serializers.py", line 372, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/hdp/current/spark2-client/python/pyspark/rdd.py", line 1354, in takeUpToNumLeft

yield next(iterator)

File "<stdin>", line 1, in <lambda>

TypeError: coercing to Unicode: need string or buffer, int found

>>> crimeFileSortRequired = crimeFileSortedReal.map(lambda cf: cf[1][0] +"," + str(cf[1][1]))

>>> for i in crimeFileSortRequired.take(10): print(i)

...

BATTERY,244394

OTHER OFFENSE,184667

THEFT,142273

CRIMINAL DAMAGE,139492

BURGLARY,127833

ASSAULT,67997

DECEPTIVE PRACTICE,64478

NARCOTICS,26446

CRIMINAL TRESPASS,23041

OFFENSE INVOLVING CHILDREN,22740

>>>

>>> crimeFileOutput = crimeFileSortRequired.take(3)

>>> type(crimeFileOutput)

<type 'list'>

>>> for i in crimeFileOutputRDD.take(10): print(i)

...

BATTERY,244394

OTHER OFFENSE,184667

THEFT,142273

>>>

INDEXES', 'LOCKS', 'OPTION', 'ANTI', 'LOCAL', 'INPATH', IDENTIFIER, BACKQUOTED\_IDENTIFIER}(line 1, pos 12)\n\n== SQL ==\nPrimary\_Type\n------------^^^\n"

>>> crimeFileDF = crimeFileOutputRDD.toDF(['Primary\_Type', 'Number\_of\_Incidents'])

>>> crimeFileDF.show()

+-------------+-------------------+

| Primary\_Type|Number\_of\_Incidents|

+-------------+-------------------+

| BATTERY| 244394|

|OTHER OFFENSE| 184667|

| THEFT| 142273|

+-------------+-------------------+

crimeFileDF.coalesce(1).write.json("/user/saurabhbhanwala/solutions/solution03/crimeTypeRDD")

Details - Duration 10 minutes

* Data is available in local file system under /data/nyse (ls -ltr /data/nyse)
* Fields (stockticker:string, transactiondate:string, openprice:float, highprice:float, lowprice:float, closeprice:float, volume:bigint)
* Convert file format to parquet
* Save it /user/<YOUR\_USER\_ID>/nyse\_parquet

[saurabhbhanwala@gw03 ~]$ hdfs dfs -put /data/nyse /user/saurabhbhanwala/nyse

>>> nyseRDD = sc.textFile("/user/saurabhbhanwala/nyse/nyse")

>>> for i in nyseRDD.take(10): print(i)

>>> from pyspark.sql import Row

>>> nyseRDDMap = nyseRDD.map(lambda m: m.split(","))

>>> for i in nyseRDDMap.take(10): print(i)

...

[u'AA', u'19970101', u'47.82', u'47.82', u'47.82', u'47.82', u'0']

[u'ABC', u'19970101', u'6.03', u'6.03', u'6.03', u'6.03', u'0']

[u'ABM', u'19970101', u'9.25', u'9.25', u'9.25', u'9.25', u'0']

[u'ABT', u'19970101', u'25.37', u'25.37', u'25.37', u'25.37', u'0']

[u'ABX', u'19970101', u'28.75', u'28.75', u'28.75', u'28.75', u'0']

[u'ACP', u'19970101', u'9.12', u'9.12', u'9.12', u'9.12', u'0']

[u'ACV', u'19970101', u'16', u'16', u'16', u'16', u'0']

[u'ADC', u'19970101', u'21.37', u'21.37', u'21.37', u'21.37', u'0']

[u'ADM', u'19970101', u'17.24', u'17.24', u'17.24', u'17.24', u'0']

[u'ADX', u'19970101', u'13.16', u'13.16', u'13.16', u'13.16', u'0']

<https://stackoverflow.com/questions/29753496/how-to-split-rows-of-a-spark-rdd-by-deliminator>

>>> nyseDF = spark.createDataFrame(nyseRDDMap)

>>> nyseDF.show()

+---+--------+------+------+------+------+---+

| \_1| \_2| \_3| \_4| \_5| \_6| \_7|

+---+--------+------+------+------+------+---+

| AA|19970101| 47.82| 47.82| 47.82| 47.82| 0|

|ABC|19970101| 6.03| 6.03| 6.03| 6.03| 0|

|ABM|19970101| 9.25| 9.25| 9.25| 9.25| 0|

|ABT|19970101| 25.37| 25.37| 25.37| 25.37| 0|

|ABX|19970101| 28.75| 28.75| 28.75| 28.75| 0|

|ACP|19970101| 9.12| 9.12| 9.12| 9.12| 0|

|ACV|19970101| 16| 16| 16| 16| 0|

|ADC|19970101| 21.37| 21.37| 21.37| 21.37| 0|

|ADM|19970101| 17.24| 17.24| 17.24| 17.24| 0|

|ADX|19970101| 13.16| 13.16| 13.16| 13.16| 0|

|AED|19970101| 31.5| 31.5| 31.5| 31.5| 0|

|AEE|19970101| 38.5| 38.5| 38.5| 38.5| 0|

|AEG|19970101| 15.2| 15.2| 15.2| 15.2| 0|

|AEM|19970101| 14| 14| 14| 14| 0|

|AEP|19970101| 41.12| 41.12| 41.12| 41.12| 0|

|AES|19970101| 11.62| 11.62| 11.62| 11.62| 0|

| AF|19970101| 12.29| 12.29| 12.29| 12.29| 0|

|AFG|19970101|25.179|25.179|25.179|25.179| 0|

|AFL|19970101| 10.69| 10.69| 10.69| 10.69| 0|

| AG|19970101| 28.62| 28.62| 28.62| 28.62| 0|

+---+--------+------+------+------+------+---+

only showing top 20 rows

user\_schema = "stockticker:string, transactionDate:string, openprice:float, highprice:float, lowprice: float, closeprice:float, volume:bigint"

>>> nyseDF.printSchema()

root

|-- \_1: string (nullable = true)

|-- \_2: string (nullable = true)

|-- \_3: string (nullable = true)

|-- \_4: string (nullable = true)

|-- \_5: string (nullable = true)

|-- \_6: string (nullable = true)

|-- \_7: string (nullable = true)

<https://stackoverflow.com/questions/35684018/how-to-convert-dataframe-columns-from-string-to-float-double-in-pyspark-1-6>

<https://stackoverflow.com/questions/53426685/does-spark-support-biginteger-type>

>>> nyseDF\_columns = nyseDF.select(nyseDF.\_1.alias('stockTicker'), nyseDF.\_2.alias('transactionDate'), nyseDF.\_3.cast(DoubleType()).alias('openPrice'), nyseDF.\_4.cast(DoubleType()).alias('highPrice'), nyseDF.\_5.cast(DoubleType()).alias('lowPrice'), nyseDF.\_6.cast(DoubleType()).alias('closePrice'), nyseDF.\_7.cast(LongType()).alias('Volume'))

>>> nyseDF\_columns.show()

+-----------+---------------+---------+---------+--------+----------+------+

|stockTicker|transactionDate|openPrice|highPrice|lowPrice|closePrice|Volume|

+-----------+---------------+---------+---------+--------+----------+------+

| AA| 19970101| 47.82| 47.82| 47.82| 47.82| 0|

| ABC| 19970101| 6.03| 6.03| 6.03| 6.03| 0|

| ABM| 19970101| 9.25| 9.25| 9.25| 9.25| 0|

| ABT| 19970101| 25.37| 25.37| 25.37| 25.37| 0|

| ABX| 19970101| 28.75| 28.75| 28.75| 28.75| 0|

| ACP| 19970101| 9.12| 9.12| 9.12| 9.12| 0|

| ACV| 19970101| 16.0| 16.0| 16.0| 16.0| 0|

| ADC| 19970101| 21.37| 21.37| 21.37| 21.37| 0|

| ADM| 19970101| 17.24| 17.24| 17.24| 17.24| 0|

| ADX| 19970101| 13.16| 13.16| 13.16| 13.16| 0|

| AED| 19970101| 31.5| 31.5| 31.5| 31.5| 0|

| AEE| 19970101| 38.5| 38.5| 38.5| 38.5| 0|

| AEG| 19970101| 15.2| 15.2| 15.2| 15.2| 0|

| AEM| 19970101| 14.0| 14.0| 14.0| 14.0| 0|

| AEP| 19970101| 41.12| 41.12| 41.12| 41.12| 0|

| AES| 19970101| 11.62| 11.62| 11.62| 11.62| 0|

| AF| 19970101| 12.29| 12.29| 12.29| 12.29| 0|

| AFG| 19970101| 25.179| 25.179| 25.179| 25.179| 0|

| AFL| 19970101| 10.69| 10.69| 10.69| 10.69| 0|

| AG| 19970101| 28.62| 28.62| 28.62| 28.62| 0|

+-----------+---------------+---------+---------+--------+----------+------+

only showing top 20 rows

>>> nyseDF\_columns.printSchema()

root

|-- stockTicker: string (nullable = true)

|-- transactionDate: string (nullable = true)

|-- openPrice: double (nullable = true)

|-- highPrice: double (nullable = true)

|-- lowPrice: double (nullable = true)

|-- closePrice: double (nullable = true)

|-- Volume: long (nullable = true)

>>>

<https://medium.com/@ivan.georgiev_19530/create-dataframe-from-python-objects-in-pyspark-bd8e191b9ebd>

>>> nyseDF\_columns.coalesce(1).write.format("parquet").save("/user/saurabhbhanwala/nyse\_parquet")

>>> sqlContext.load("/user/saurabhbhanwala/nyse\_parquet").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SQLContext' object has no attribute 'load'

>>> sqlContext.read.load("/user/saurabhbhanwala/nyse\_parquet").show()

+-----------+---------------+---------+---------+--------+----------+------+

|stockTicker|transactionDate|openPrice|highPrice|lowPrice|closePrice|Volume|

+-----------+---------------+---------+---------+--------+----------+------+

| AA| 19970101| 47.82| 47.82| 47.82| 47.82| 0|

| ABC| 19970101| 6.03| 6.03| 6.03| 6.03| 0|

| ABM| 19970101| 9.25| 9.25| 9.25| 9.25| 0|

| ABT| 19970101| 25.37| 25.37| 25.37| 25.37| 0|

| ABX| 19970101| 28.75| 28.75| 28.75| 28.75| 0|

| ACP| 19970101| 9.12| 9.12| 9.12| 9.12| 0|

| ACV| 19970101| 16.0| 16.0| 16.0| 16.0| 0|

| ADC| 19970101| 21.37| 21.37| 21.37| 21.37| 0|

| ADM| 19970101| 17.24| 17.24| 17.24| 17.24| 0|

| ADX| 19970101| 13.16| 13.16| 13.16| 13.16| 0|

| AED| 19970101| 31.5| 31.5| 31.5| 31.5| 0|

| AEE| 19970101| 38.5| 38.5| 38.5| 38.5| 0|

| AEG| 19970101| 15.2| 15.2| 15.2| 15.2| 0|

| AEM| 19970101| 14.0| 14.0| 14.0| 14.0| 0|

| AEP| 19970101| 41.12| 41.12| 41.12| 41.12| 0|

| AES| 19970101| 11.62| 11.62| 11.62| 11.62| 0|

| AF| 19970101| 12.29| 12.29| 12.29| 12.29| 0|

| AFG| 19970101| 25.179| 25.179| 25.179| 25.179| 0|

| AFL| 19970101| 10.69| 10.69| 10.69| 10.69| 0|

| AG| 19970101| 28.62| 28.62| 28.62| 28.62| 0|

+-----------+---------------+---------+---------+--------+----------+------+

only showing top 20 rows