

Datasets, DataFrames, and Spark SQL

```
%sh pwd

/data/test_spark/zeppelin-0.8.1-bin-netinst

// Input files
z.put("flight-data-2018", "/home/ubuntu/test_spark/flight/flightdata2018.json")

%sh

ls -l {flight-data-2018}

-rw-rw-r-- 1 ubuntu ubuntu 66074026 May 17 16:53 /home/ubuntu/test_spark/flight/flightdata2018.json

%sh

head {flight-data-2018}

{"id":"ATL_BOS_2018-01-01_DL_104","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":9,"crsdeptime":850,"depdelay":0.0,"crsarrrtime":1116,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_1200","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":11,"crsdeptime":1122,"depdelay":8.0,"crsarrrtime":1349,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_1500","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":14,"crsdeptime":1356,"depdelay":9.0,"crsarrrtime":1623,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_1800","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":16,"crsdeptime":1620,"depdelay":0.0,"crsarrrtime":1851,"arrdelay":3.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_2100","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":19,"crsdeptime":1940,"depdelay":6.0,"crsarrrtime":2210,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_2792","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":12,"crsdeptime":1248,"depdelay":0.0,"crsarrrtime":1513,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_903","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":22,"crsdeptime":2215,"depdelay":0.0,"crsarrrtime":39,"arrdelay":0.0,"crselaps":{"id":"ATL_BOS_2018-01-01_DL_980","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"DL","src":"ATL","dst":"BOS","crsdephour":15,"crsdeptime":1500,"depdelay":21.0,"crsarrrtime":1734,"arrdelay":33.0,"crselaps":{"id":"ATL_BOS_2018-01-01_WN_1633","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"WN","src":"ATL","dst":"BOS","crsdephour":15,"crsdeptime":1500,"depdelay":198.0,"crsarrrtime":1725,"arrdelay":208.0,"crselaps":{"id":"ATL_BOS_2018-01-01_WN_170","fdate":"2018-01-01","month":1,"dofw":1,"carrier":"WN","src":"ATL","dst":"BOS","crsdephour":21,"crsdeptime":2055,"depdelay":14.0,"crsarrrtime":2330,"arrdelay":0.0,"crselaps
```

The flight data is in JSON files, with each flight having the following information:

- id: ID composed of carrier, date, origin, destination, flight number
- dofw: day of week (1=Monday, 7=Sunday)
- carrier: carrier code
- origin: origin airport code
- dest: destination airport code
- crsdephour: scheduled departure hour
- crsdeptime: scheduled departure time
- depdelay: departure delay in minutes
- csarrrtime: scheduled arrival time
- arrdelay: arrival delay minutes
- crselapsedtime: elapsed time
- dist: distance

```
// Read JSON as DF (auto infer schema)
val flightDF = spark.read.format("json").load(z.get("flight-data-2018").toString)
// flightDF_autoinfer.printSchema

flightDF_autoinfer: org.apache.spark.sql.DataFrame = [arrdelay: double, carrier: string ... 12 more fields]

flightDF_autoinfer.printSchema()

root
 |-- arrdelay: double (nullable = true)
 |-- carrier: string (nullable = true)
 |-- csarrrtime: long (nullable = true)
 |-- crsdephour: long (nullable = true)
 |-- crsdeptime: long (nullable = true)
 |-- crselapsedtime: double (nullable = true)
 |-- depdelay: double (nullable = true)
 |-- dist: double (nullable = true)
 |-- dofw: long (nullable = true)
 |-- dst: string (nullable = true)
 |-- fdate: string (nullable = true)
 |-- id: string (nullable = true)
 |-- month: long (nullable = true)
 |-- src: string (nullable = true)
```

```
flightDF_autoinfer.filter("crsdephour == 10").show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|arrdelay|carrier|csarrrtime|crsdephour|crsdeptime|crselapsedtime|depdelay|dist|dofw|dst|fdate|id|month|src|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|191.0|WN|1250|10|1015|155.0|215.0|946.0|1|BOS|2018-01-01|ATL_BOS_2018-01-0...|1|ATL|
|0.0|DL|1124|10|1008|76.0|0.0|226.0|1|CLT|2018-01-01|ATL_CLT_2018-01-0...|1|ATL|
|7.0|AA|1131|10|1000|151.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|37.0|DL|1120|10|950|150.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|67.0|DL|1216|10|1010|126.0|0.0|746.0|1|EWR|2018-01-01|ATL_EWR_2018-01-0...|1|ATL|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

only showing top 5 rows
```

```
flightDF_autoinfer.filter(col("crsdephour") === 10).show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|arrdelay|carrier|csarrrtime|crsdephour|crsdeptime|crselapsedtime|depdelay|dist|dofw|dst|fdate|id|month|src|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|191.0|WN|1250|10|1015|155.0|215.0|946.0|1|BOS|2018-01-01|ATL_BOS_2018-01-0...|1|ATL|
|0.0|DL|1124|10|1008|76.0|0.0|226.0|1|CLT|2018-01-01|ATL_CLT_2018-01-0...|1|ATL|
|7.0|AA|1131|10|1000|151.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|37.0|DL|1120|10|950|150.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|67.0|DL|1216|10|1010|126.0|0.0|746.0|1|EWR|2018-01-01|ATL_EWR_2018-01-0...|1|ATL|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

only showing top 5 rows
```

```
flightDF_autoinfer.filter($"crsdephour" === 10).show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|arrdelay|carrier|csarrrtime|crsdephour|crsdeptime|crselapsedtime|depdelay|dist|dofw|dst|fdate|id|month|src|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|191.0|WN|1250|10|1015|155.0|215.0|946.0|1|BOS|2018-01-01|ATL_BOS_2018-01-0...|1|ATL|
|0.0|DL|1124|10|1008|76.0|0.0|226.0|1|CLT|2018-01-01|ATL_CLT_2018-01-0...|1|ATL|
|7.0|AA|1131|10|1000|151.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|37.0|DL|1120|10|950|150.0|0.0|731.0|1|DFW|2018-01-01|ATL_DFW_2018-01-0...|1|ATL|
|67.0|DL|1216|10|1010|126.0|0.0|746.0|1|EWR|2018-01-01|ATL_EWR_2018-01-0...|1|ATL|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

only showing top 5 rows
```

```
val c1 = col("crsdephour")

c1: org.apache.spark.sql.Column = crsdephour

val c2 = $"crsdephour"

c2: org.apache.spark.sql.ColumnName = crsdephour

// Read JSON as DF (Provide schema manually)

val schemaStr = """id String, fdate Date, month Integer, dofw Integer, carrier String, src String,
|dst String, crsdephour Integer, crsdeptime Integer, depdelay Double, csarrrtime Integer,
|arrdelay Double, crselapsedtime Double, dist Double""".stripMargin.replaceAll("\n", " ")

val flightDF = spark.read.format("json")
    .schema(schemaStr)
    .load(z.get("flight-data-2018").toString)

flightDF: org.apache.spark.sql.DataFrame = [id: string, fdate: date ... 12 more fields]

flightDF.printSchema

root
 |-- id: string (nullable = true)
 |-- fdate: date (nullable = true)
 |-- month: integer (nullable = true)
 |-- dofw: integer (nullable = true)
 |-- carrier: string (nullable = true)
```

```
-- src: string (nullable = true)
-- dst: string (nullable = true)
-- crsdephour: integer (nullable = true)
-- crsdeptime: integer (nullable = true)
-- depdelay: double (nullable = true)
-- crsarrrtime: integer (nullable = true)
-- arrdelay: double (nullable = true)
-- crselapsedtime: double (nullable = true)
-- dist: double (nullable = true)
```

flightDF.show

	id	fldate	month	dofW	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	9	850	0.0	1116	0.0	146.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	11	1122	8.0	1349	0.0	147.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	14	1356	9.0	1623	0.0	147.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	16	1620	0.0	1851	3.0	151.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	19	1940	6.0	2210	0.0	150.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	12	1248	0.0	1513	0.0	145.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	22	2215	0.0	39	0.0	144.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	DL	ATL	BOS	15	1500	21.0	1734	33.0	154.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	WN	ATL	BOS	15	1500	198.0	1725	208.0	145.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	WN	ATL	BOS	21	2055	14.0	2330	0.0	155.0	946.0	
ATL_BOS_2018-01-0...	[2018-01-01]	1	1	WN	ATL	BOS	10	1015	215.0	1250	191.0	155.0	946.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	11	1114	0.0	1238	0.0	84.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	8	845	0.0	1011	0.0	86.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	15	1548	0.0	1710	0.0	82.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	7	705	0.0	821	0.0	76.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	12	1226	0.0	1347	0.0	81.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	AA	ATL	CLT	22	2205	0.0	2319	1.0	74.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	DL	ATL	CLT	22	2210	11.0	2324	0.0	74.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	DL	ATL	CLT	15	1543	1.0	1659	0.0	76.0	226.0	
ATL_CLT_2018-01-0...	[2018-01-01]	1	1	DL	ATL	CLT	10	1008	0.0	1124	0.0	76.0	226.0	

only showing top 20 rows

```
// take() method returns Array of Row objects
```

```
flightDF.take(2)
```

```
res15: Array[org.apache.spark.sql.Row] = Array([ATL_BOS_2018-01-01_DL_104,2018-01-01,1,1,DL,ATL,BOS,9,850,0.0,1116,0.0,146.0,946.0], [ATL_BOS_2018-01-01_DL_1200,2018-01-01,1,1,DL,ATL,BOS,11,1122,8.0,1349,0.0,147.0,946.0])
```

Loading data into *DataSet* of typed objects

```
// Create case class
import java.sql.Date
import org.apache.spark.sql.types._
import spark.implicits._
case class Flight(id: String, fldate: Date, month: Integer, dofW: Integer, carrier: String, src: String,
  dst: String, crsdephour: Integer, crsdeptime: Integer, depdelay: Double, crsarrrtime: Integer,
  arrdelay: Double, crselapsedtime: Double, dist: Double) extends Serializable
```

```
val schemaStr1 = """id String, fldate Date, month Integer, dofW Integer, carrier String, src String,
  [dst String, crsdephour Integer, crsdeptime Integer, depdelay Double, crsarrrtime Integer,
  [arrdelay Double, crselapsedtime Double, dist Double""".stripMargin.replaceAll("\n", " ")
```

```
val flightDS = spark.read.format("json")
  .schema(schemaStr1)
  .load(z.get("flight-data-2018").toString)
  .as[Flight]
```

```
import java.sql.Date
import org.apache.spark.sql.types._
import spark.implicits._
defined class Flight
flightDS: org.apache.spark.sql.Dataset[Flight] = [id: string, fldate: date ... 12 more fields]
```

flightDF_autoinfer.printSchema

```
root
|-- arrdelay: double (nullable = true)
|-- carrier: string (nullable = true)
|-- crsarrrtime: long (nullable = true)
|-- crsdephour: long (nullable = true)
|-- crsdeptime: long (nullable = true)
|-- crselapsedtime: double (nullable = true)
|-- depdelay: double (nullable = true)
|-- dist: double (nullable = true)
|-- dofW: long (nullable = true)
|-- dst: string (nullable = true)
|-- fldate: string (nullable = true)
|-- id: string (nullable = true)
|-- month: long (nullable = true)
|-- src: string (nullable = true)
```

```
// Another way, using auto infer schema
```

```
// NOTE: In this case make sure that auto inferred data types are matching the case class
```

```
import java.sql.Date
import spark.implicits._
case class Flight1(id: String, fldate: Date, month: Long, dofW: Long, carrier: String, src: String,
  dst: String, crsdephour: Long, crsdeptime: Long, depdelay: Double, crsarrrtime: Long,
  arrdelay: Double, crselapsedtime: Double, dist: Double) extends Serializable
```

```
// val flightDS_autoinfer = spark.read.format("json").load(z.get("flight-data-2018").toString).as[Flight1]
// same as below
```

```
val flightDS_autoinfer = flightDF_autoinfer.as[Flight1]
```

```
import java.sql.Date
import spark.implicits._
defined class Flight1
flightDS_autoinfer: org.apache.spark.sql.Dataset[Flight1] = [arrdelay: double, carrier: string ... 12 more fields]
```

```
flightDS.take(2)
```

```
res35: Array[Flight] = Array(Flight(ATL_BOS_2018-01-01_DL_104,2018-01-01,1,1,DL,ATL,BOS,9,850,0.0,1116,0.0,146.0,946.0), Flight(ATL_BOS_2018-01-01_DL_1200,2018-01-01,1,1,DL,ATL,BOS,11,1122,8.0,1349,0.0,147.0,946.0))
```

```
flightDS_autoinfer.take(2)
```

```
res36: Array[Flight1] = Array(Flight1(ATL_BOS_2018-01-01_DL_104,2018-01-01,1,1,DL,ATL,BOS,9,850,0.0,1116,0.0,146.0,946.0), Flight1(ATL_BOS_2018-01-01_DL_1200,2018-01-01,1,1,DL,ATL,BOS,11,1122,8.0,1349,0.0,147.0,946.0))
```

flightDS.printSchema

```
root
|-- id: string (nullable = true)
|-- fldate: date (nullable = true)
|-- month: integer (nullable = true)
|-- dofW: integer (nullable = true)
|-- carrier: string (nullable = true)
|-- src: string (nullable = true)
|-- dst: string (nullable = true)
|-- crsdephour: integer (nullable = true)
|-- crsdeptime: integer (nullable = true)
|-- depdelay: double (nullable = true)
|-- crsarrrtime: integer (nullable = true)
|-- arrdelay: double (nullable = true)
|-- crselapsedtime: double (nullable = true)
|-- dist: double (nullable = true)
```

```
flightDS.show(5)
```

	id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS		9	850	0.0	1116	0.0	146.0	946.0
ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS		11	1122	8.0	1349	0.0	147.0	946.0
ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS		14	1356	9.0	1623	0.0	147.0	946.0
ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS		16	1620	0.0	1851	3.0	151.0	946.0
ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS		19	1940	6.0	2210	0.0	150.0	946.0

only showing top 5 rows

Common Dataset typed transformations “Dataset[T]”

- filter
- map
- groupByKey

```
flightDS.filter(flight => flight.crsdephour == 10).take(3)
```

```
res3: Array[Flight] = Array(Flight(ATL_BOS_2018-01-01_WN_1726,2018-01-01,1,1,WN,ATL,BOS,10,1015,215.0,1250,191.0,155.0,946.0), Flight(ATL_CLT_2018-01-01_DL_2251,2018-01-01,1,1,DL,ATL,CLT,10,1008,0.0,1124
```

```
flightDS.filter(_ .crsdephour == 10).take(3)
```

```
res4: Array[Flight] = Array(Flight(ATL_BOS_2018-01-01_WN_1726,2018-01-01,1,1,WN,ATL,BOS,10,1015,215.0,1250,191.0,155.0,946.0), Flight(ATL_CLT_2018-01-01_DL_2251,2018-01-01,1,1,DL,ATL,CLT,10,1008,0.0,1124
```

```
flightDS.filter($"crsdephour" === 10).show(3)
```

	id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
ATL_BOS_2018-01-0...	2018-01-01	1	1	WN	ATL	BOS		10	1015	215.0	1250	191.0	155.0	946.0
ATL_CLT_2018-01-0...	2018-01-01	1	1	DL	ATL	CLT		10	1008	0.0	1124	0.0	76.0	226.0
ATL_DFW_2018-01-0...	2018-01-01	1	1	AA	ATL	DFW		10	1000	0.0	1131	7.0	151.0	731.0

only showing top 3 rows

```
flightDS.where("crsdephour = 10").show(3)
```

	id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
ATL_BOS_2018-01-0...	2018-01-01	1	1	WN	ATL	BOS		10	1015	215.0	1250	191.0	155.0	946.0
ATL_CLT_2018-01-0...	2018-01-01	1	1	DL	ATL	CLT		10	1008	0.0	1124	0.0	76.0	226.0
ATL_DFW_2018-01-0...	2018-01-01	1	1	AA	ATL	DFW		10	1000	0.0	1131	7.0	151.0	731.0

only showing top 3 rows

```
flightDS.explain(true)
```

```
== Parsed Logical Plan ==
```

```
Relation[id#2082,fldate#2083,month#2084,dofw#2085,carrier#2086,src#2087,dst#2088,crsdephour#2089,crsdeptime#2090,depdelay#2091,crsarrrtime#2092,arrdelay#2093,crselapsedtime#2094,dist#2095] json
```

```
== Analyzed Logical Plan ==
```

```
id: string, fldate: date, month: int, dofw: int, carrier: string, src: string, dst: string, crsdephour: int, crsdeptime: int, depdelay: double, csarrrtime: int, arrdelay: double, crselapsedtime: double, Relation[id#2082,fldate#2083,month#2084,dofw#2085,carrier#2086,src#2087,dst#2088,crsdephour#2089,crsdeptime#2090,depdelay#2091,crsarrrtime#2092,arrdelay#2093,crselapsedtime#2094,dist#2095] json
```

```
== Optimized Logical Plan ==
```

```
InMemoryRelation [id#2082, fldate#2083, month#2084, dofw#2085, carrier#2086, src#2087, dst#2088, crsdephour#2089, crsdeptime#2090, depdelay#2091, csarrrtime#2092, arrdelay#2093, crselapsedtime#2094, dist#2095] FileScan json [id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] Batched: false, Format: JSON, Location: (null)
```

```
== Physical Plan ==
```

```
InMemoryTableScan [id#2082, fldate#2083, month#2084, dofw#2085, carrier#2086, src#2087, dst#2088, crsdephour#2089, crsdeptime#2090, depdelay#2091, csarrrtime#2092, arrdelay#2093, crselapsedtime#2094, dist#2095] InMemoryRelation [id#2082, fldate#2083, month#2084, dofw#2085, carrier#2086, src#2087, dst#2088, crsdephour#2089, crsdeptime#2090, depdelay#2091, csarrrtime#2092, arrdelay#2093, crselapsedtime#2094, dist#2095] FileScan json [id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] Batched: false, Format: JSON, Location: (null)
```

Common Dataset untyped transformations “Dataframe = Dataset[Row]”

- select
- join
- groupBy

```
flightDS.groupBy(col("carrier")).count().show()
```

carrier	count
UA	98978
AA	109427
DL	58195
WN	16028

```
flightDS.groupBy("carrier").count().show()
```

carrier	count
UA	98978
AA	109427
DL	58195
WN	16028

```
flightDF.groupBy("carrier").count().show()
```

carrier	count
UA	98978
AA	109427
DL	58195
WN	16028

```
flightDF.explain(true)
```

```
== Parsed Logical Plan ==
```

```
Relation[id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] json
```

```
== Analyzed Logical Plan ==
```

```
id: string, fldate: date, month: int, dofw: int, carrier: string, src: string, dst: string, crsdephour: int, crsdeptime: int, depdelay: double, csarrrtime: int, arrdelay: double, crselapsedtime: double, Relation[id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] json
```

```
== Optimized Logical Plan ==
```

```
InMemoryRelation [id#0, fldate#1, month#2, dofw#3, carrier#4, src#5, dst#6, crsdephour#7, crsdeptime#8, depdelay#9, csarrrtime#10, arrdelay#11, crselapsedtime#12, dist#13], StorageLevel(disk, memory, deserialized) FileScan json [id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] Batched: false, Format: JSON, Location: (null)
```

```
== Physical Plan ==
```

```
InMemoryTableScan [id#0, fldate#1, month#2, dofw#3, carrier#4, src#5, dst#6, crsdephour#7, crsdeptime#8, depdelay#9, csarrrtime#10, arrdelay#11, crselapsedtime#12, dist#13] InMemoryRelation [id#0, fldate#1, month#2, dofw#3, carrier#4, src#5, dst#6, crsdephour#7, crsdeptime#8, depdelay#9, csarrrtime#10, arrdelay#11, crselapsedtime#12, dist#13], StorageLevel(disk, memory, deserialized) FileScan json [id#0,fldate#1,month#2,dofw#3,carrier#4,src#5,dst#6,crsdephour#7,crsdeptime#8,depdelay#9,csarrrtime#10,arrdelay#11,crselapsedtime#12,dist#13] Batched: false, Format: JSON, Location: (null)
```

Commonly used Dataset actions

- show(n)
- take(n)
- count

```
flightDS.select("id", "crsdephour").where($"crsdephour" === 10).show(3)
```

id	crsdephour
ATL_BOS_2018-01-0...	10
ATL_CLT_2018-01-0...	10

```
|ATL_DFW_2018-01-0...| 10|
+-----+
only showing top 3 rows

flightDS.filter(_._depdelay > 40).groupBy("dst").count().orderBy($"count".desc).show(3)

+-----+
|dst|count|
+-----+
|ORD| 3465|
|SFO| 2946|
|EWR| 2796|
+-----+
only showing top 3 rows

flightDS.filter(_._depdelay > 40).groupBy("dst").count().orderBy(desc("count")).show(3)
```

```
+-----+
|dst|count|
+-----+
|ORD| 3465|
|SFO| 2946|
|EWR| 2796|
+-----+
only showing top 3 rows

flightDS.filter(_._depdelay > 40).groupBy("dst").count()

res31: org.apache.spark.sql.DataFrame = [dst: string, count: bigint]

desc("count")

res28: org.apache.spark.sql.Column = count DESC NULLS LAST

$"count".desc

res29: org.apache.spark.sql.Column = count DESC NULLS LAST
```

Spark SQL

```
// Cache flight dataframe
flightDF.cache

res2: flightDF.type = [id: string, fldate: date ... 12 more fields]

// create table view of DF
flightDF.createOrReplaceTempView("flights")

// cache flights table in memory
spark.catalog.cacheTable("flights")

// Spark SQL
spark.sql("select * from flights limit 4").show
```

	id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
	ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS	9	850	0.0	1116	0.0	146.0	946.0
	ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS	11	1122	8.0	1349	0.0	147.0	946.0
	ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS	14	1356	9.0	1623	0.0	147.0	946.0
	ATL_BOS_2018-01-0...	2018-01-01	1	1	DL	ATL	BOS	16	1620	0.0	1851	3.0	151.0	946.0

```
%sql select carrier, avg(depdelay) from flights group by carrier
```

carrier	avg(depdelay)
UA	14.442583200307139
AA	14.129090626627798
DL	14.26168914855228
WN	17.70183428999251

```
flightDS.select($"carrier", $"src", $"dst", $"depdelay", $"crsdephour", $"crsdephour").filter($"depdelay" > 40).orderBy(desc("depdelay")).show(5)
```

	carrier	src	dst	depdelay	crsdephour	crsdephour
	AA	IAH	MIA	1559.0	11	11
	AA	IAH	DFW	1445.0	18	18
	AA	BOS	DFW	1345.0	10	10
	UA	BOS	ORD	1334.0	9	9
	AA	LAX	MIA	1283.0	9	9

```
only showing top 5 rows
```

```
%sql
select dofw, count(depdelay)
from flights where depdelay > 40
group by dofw
```

dofw	count(depdelay)
1	4680
6	2598
3	3986
5	4677
4	4809
7	4163
2	4406

```
flightDS.select($"dofw", $"depdelay").filter($"depdelay" > 40).groupBy($"dofw").count().show()
```

	dofw	count
	1	4680
	6	2598
	3	3986
	5	4677
	4	4809
	7	4163
	2	4406

```
%sql
select crsdephour, count(depdelay)
from flights where depdelay > 40
group by crsdephour order by crsdephour
```

crsdephour	count(depdelay)
0	106
1	51
5	175
6	518
7	1068
8	1067
9	1112
10	1472
11	1321
12	1803
13	1728
14	2041
15	1969
16	2309
17	2434
18	2029
19	2198
20	2260
21	1180
22	1066
23	364

```
24      58
flightD5.select($"crsdephour", $"depdelay").filter($"depdelay" > 40).groupBy($"crsdephour").count().orderBy($"crsdephour").show()
```

+-----+-----+	
crsdephour count	
+-----+-----+	
	0 106
	1 51
	5 175
	6 518
	7 1068
	8 1067
	9 1112
	10 1472
	11 1321
	12 1893
	13 1728
	14 2041
	15 1969
	16 2309
	17 2434
	18 2929
	19 2198
	20 2260
	21 1180
	22 1066

only showing top 20 rows

```
%sql
select src, count(depdelay)
from flights where depdelay > 40
group by src
ORDER BY count(depdelay) desc
```

src	count(depdelay)
ORD	4033
ATL	3106
DFW	2782
EWI	2328
DEN	2304
MIA	2294
SFO	2261
LAX	2126
LGA	1944
IAH	1829
CLT	1755
BOS	1711
SEA	846

```
flightD5.select($"src", $"depdelay").filter($"depdelay" > 40).groupBy($"src").count().withColumnRenamed("count", "total").orderBy($"total".desc).show()
```

+---+-----+	
src total	
+---+-----+	
ORD	4033
ATL	3106
DFW	2782
EWI	2328
DEN	2304
MIA	2294
SFO	2261
LAX	2126
LGA	1944
IAH	1829
CLT	1755
BOS	1711
SEA	846

```
%sql
select src, dst, count(depdelay)
from flights where depdelay > 40
group by src, dst
ORDER BY count(depdelay) desc
```

src	dst	count(depdelay)
ORD	LGA	588
LAX	SFO	578
ATL	EWI	561
LGA	ORD	532
ORD	SFO	470
SFO	LAX	463
ATL	LGA	445
ORD	LAX	425
ORD	DFW	423
DEN	SFO	388
EWI	ATL	381
DFW	ORD	377
MIA	LGA	355
ORD	EWI	347
LGA	ATL	344
SFO	ORD	343
DFW	ATL	328
ORD	ATL	326
ORD	BOS	318
DEN	LAX	311
LAX	DEN	303
ORD	DEN	302
SFO	DEN	299
MIA	ATL	299
LAX	ORD	298
EWI	ORD	297
ATL	DFW	293
BOS	LGA	279
LGA	BOS	276
DEN	EWI	275
ATL	LAX	275
ATL	ORD	273
BOS	EWI	269
DFW	SFO	269
IAH	EWI	269
BOS	ORD	267
DEN	ORD	265
IAH	DFW	264
MIA	ORD	262
MIA	DFW	259
EWI	SFO	257
ATL	MIA	255
CLT	EWI	255
EWI	BOS	255
DFW	IAH	255
LGA	MIA	248
ATL	BOS	243
ORD	SEA	232
DFW	DEN	222
ORD	IAH	217
DFW	MIA	217
DFW	LAX	216
DFW	LGA	214
SFO	EWI	214
CLT	ORD	213
ORD	MIA	212
EWI	LAX	202
SFO	DFW	202

```
MIA EWR 201
LAX ATL 201
DFW EWR 200
IAH SFO 199
BOS ATL 198
MIA LAX 198
DEN DFW 198
SFO SEA 196
ATL DEN 192
SEA SFO 191
IAH ORD 189
EWR MIA 188
CLT ATL 186
DFW CLT 185
ATL SFO 185
DEN LGA 184
LGA DEN 182
CLT BOS 181
MIA BOS 178
EWR DEN 178
EWR IAH 177
LAX DFW 174
DFW SEA 174
MIA CLT 174
ORD CLT 173
ATL CLT 170
SFO BOS 165
DEN ATL 164
MIA SFO 164
EWR CLT 162
LGA DFW 162
EWR DFW 159
DEN SEA 155
CLT DFW 155
IAH LGA 151
SEA ORD 149
CLT LGA 147
CLT MIA 144
LAX EWR 140
IAH DEN 139
IAH LAX 136
IAH ATL 136
SFO ATL 128
SEA DEN 126
LAX MIA 126
DFW BOS 125
CLT SFO 125
ATL IAH 124
LGA CLT 124
MIA IAH 123
SFO IAH 122
DEN IAH 120
BOS SFO 120
BOS CLT 116
LAX BOS 114
BOS LAX 113
DEN BOS 112
CLT DEN 106
SEA DFW 97
BOS MIA 97
IAH MIA 94
CLT LAX 93
IAH BOS 90
ATL SEA 90
IAH CLT 86
CLT IAH 86
BOS DFW 85
BOS DEN 81
IAH SEA 76
LGA IAH 76
EWR SEA 72
SFO MIA 70
LAX SEA 69
SEA ATL 68
DEN CLT 67
DEN MIA 65
SEA EWR 65
LAX CLT 65
CLT SEA 64
BOS IAH 63
SFO CLT 59
LAX IAH 58
SEA LAX 48
SEA IAH 47
MIA DEN 41
MIA SEA 40
SEA MIA 25
BOS SEA 23
SEA BOS 21
SEA CLT 9
```

```
flightDB.select($"src", $"dst", $"depdelay").filter($"depdelay" > 40).groupBy($"src", $"dst").count().orderBy($"count".desc).show()
```

UDF

<https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-udfs.html>

```
// List available standard and user defined functions

spark.catalog.listFunctions.show(false)
```

name	database	description	className	isTemporary
!	null	null	org.apache.spark.sql.catalyst.expressions.Not	true
%	null	null	org.apache.spark.sql.catalyst.expressions.Remainder	true
&	null	null	org.apache.spark.sql.catalyst.expressions.BitwiseAnd	true
*	null	null	org.apache.spark.sql.catalyst.expressions.Multiply	true

+	null	null	org.apache.spark.sql.catalyst.expressions.Add	true
-	null	null	org.apache.spark.sql.catalyst.expressions.Subtract	true
/	null	null	org.apache.spark.sql.catalyst.expressions.Divide	true
<	null	null	org.apache.spark.sql.catalyst.expressions.LessThan	true
<=	null	null	org.apache.spark.sql.catalyst.expressions.LessThanOrEqual	true
<=>	null	null	org.apache.spark.sql.catalyst.expressions.EqualNullSafe	true
=	null	null	org.apache.spark.sql.catalyst.expressions.EqualTo	true
==	null	null	org.apache.spark.sql.catalyst.expressions.EqualTo	true
>	null	null	org.apache.spark.sql.catalyst.expressions.GreaterThan	true
>=	null	null	org.apache.spark.sql.catalyst.expressions.GreaterThanOrEqual	true
^	null	null	org.apache.spark.sql.catalyst.expressions.BitwiseXor	true
abs	null	null	org.apache.spark.sql.catalyst.expressions.Abs	true
acos	null	null	org.apache.spark.sql.catalyst.expressions.Acos	true
add_months	null	null	org.apache.spark.sql.catalyst.expressions.AddMonths	true
aggregate	null	null	org.apache.spark.sql.catalyst.expressions.ArrayAggregate	true
and	null	null	org.apache.spark.sql.catalyst.expressions.And	true

only showing top 20 rows

flightDS.show(false)

id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist
ATL_BOS_2018-01-01_DL_104	2018-01-01	1	DL	ATL	BOS	9	850	0.0	1116	0.0	146.0	946.0	
ATL_BOS_2018-01-01_DL_1200	2018-01-01	1	DL	ATL	BOS	11	1122	8.0	1349	0.0	147.0	946.0	
ATL_BOS_2018-01-01_DL_1500	2018-01-01	1	DL	ATL	BOS	14	1356	9.0	1623	0.0	147.0	946.0	
ATL_BOS_2018-01-01_DL_1800	2018-01-01	1	DL	ATL	BOS	16	1620	0.0	1851	3.0	151.0	946.0	
ATL_BOS_2018-01-01_DL_2100	2018-01-01	1	DL	ATL	BOS	19	1940	6.0	2210	0.0	150.0	946.0	
ATL_BOS_2018-01-01_DL_2792	2018-01-01	1	DL	ATL	BOS	12	1248	0.0	1513	0.0	145.0	946.0	
ATL_BOS_2018-01-01_DL_903	2018-01-01	1	DL	ATL	BOS	22	2215	0.0	39	0.0	144.0	946.0	
ATL_BOS_2018-01-01_DL_980	2018-01-01	1	DL	ATL	BOS	15	1500	21.0	1734	33.0	154.0	946.0	
ATL_BOS_2018-01-01_WN_1633	2018-01-01	1	WN	ATL	BOS	15	1500	198.0	1725	208.0	145.0	946.0	
ATL_BOS_2018-01-01_WN_170	2018-01-01	1	WN	ATL	BOS	21	2055	14.0	2330	0.0	155.0	946.0	
ATL_BOS_2018-01-01_WN_1726	2018-01-01	1	WN	ATL	BOS	10	1015	215.0	1250	191.0	155.0	946.0	
ATL_CLT_2018-01-01_AA_1973	2018-01-01	1	AA	ATL	CLT	11	1114	0.0	1238	0.0	84.0	226.0	
ATL_CLT_2018-01-01_AA_2010	2018-01-01	1	AA	ATL	CLT	8	845	0.0	1011	0.0	86.0	226.0	
ATL_CLT_2018-01-01_AA_2018	2018-01-01	1	AA	ATL	CLT	15	1548	0.0	1710	0.0	82.0	226.0	
ATL_CLT_2018-01-01_AA_2095	2018-01-01	1	AA	ATL	CLT	7	705	0.0	821	0.0	76.0	226.0	
ATL_CLT_2018-01-01_AA_2472	2018-01-01	1	AA	ATL	CLT	12	1226	0.0	1347	0.0	81.0	226.0	
ATL_CLT_2018-01-01_AA_757	2018-01-01	1	AA	ATL	CLT	22	2205	0.0	2319	1.0	74.0	226.0	
ATL_CLT_2018-01-01_DL_2052	2018-01-01	1	DL	ATL	CLT	22	2210	11.0	2324	0.0	74.0	226.0	
ATL_CLT_2018-01-01_DL_2133	2018-01-01	1	DL	ATL	CLT	15	1543	1.0	1659	0.0	76.0	226.0	
ATL_CLT_2018-01-01_DL_2251	2018-01-01	1	DL	ATL	CLT	10	1008	0.0	1124	0.0	76.0	226.0	

only showing top 20 rows

// Using inbuild UDF (Preferred way)

import org.apache.spark.sql.functions._

flightDS.select(lower('carrier')).show

lower(carrier)
dl
dl
dl
dl
dl
dl
dl
wn
wn
wn
aa
aa
aa
aa
aa
aa
dl
dl
dl

only showing top 20 rows

import org.apache.spark.sql.functions._

// Create UDF

val myLower: String => String = _.toLowerCase

val lowerUDF = udf(myLower)

myLower: String => String = <function1>

lowerUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringType)))

flightDS.withColumn("lower_src", lowerUDF('src')).show

id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist	lower_src
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	9	850	0.0	1116	0.0	146.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	11	1122	8.0	1349	0.0	147.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	14	1356	9.0	1623	0.0	147.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	16	1620	0.0	1851	3.0	151.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	19	1940	6.0	2210	0.0	150.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	12	1248	0.0	1513	0.0	145.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	22	2215	0.0	39	0.0	144.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	15	1500	21.0	1734	33.0	154.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	WN	ATL	BOS	15	1500	198.0	1725	208.0	145.0	946.0	atl	wn
ATL_BOS_2018-01-0...	2018-01-01	1	WN	ATL	BOS	21	2055	14.0	2330	0.0	155.0	946.0	atl	wn
ATL_BOS_2018-01-0...	2018-01-01	1	WN	ATL	BOS	10	1015	215.0	1250	191.0	155.0	946.0	atl	wn
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	11	1114	0.0	1238	0.0	84.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	8	845	0.0	1011	0.0	86.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	15	1548	0.0	1710	0.0	82.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	7	705	0.0	821	0.0	76.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	12	1226	0.0	1347	0.0	81.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	AA	ATL	CLT	22	2205	0.0	2319	1.0	74.0	226.0	atl	aa
ATL_CLT_2018-01-0...	2018-01-01	1	DL	ATL	CLT	22	2210	11.0	2324	0.0	74.0	226.0	atl	dl
ATL_CLT_2018-01-0...	2018-01-01	1	DL	ATL	CLT	15	1543	1.0	1659	0.0	76.0	226.0	atl	dl
ATL_CLT_2018-01-0...	2018-01-01	1	DL	ATL	CLT	10	1008	0.0	1124	0.0	76.0	226.0	atl	dl

only showing top 20 rows

```
def lowerNullCheck(s: String): Option[String] = {  
  val str = Option(s).getOrElse(return None)  
  Some(str.toLowerCase().replaceAll("\\s+", ""))  
}
```

val lowerWithNullCheck = udf[Option[String], String](lowerNullCheck)

lowerNullCheck: (s: String)Option[String]

lowerWithNullCheck: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringType)))

flightDS.withColumn("lower_src", lowerWithNullCheck('src')).show

id	fldate	month	dofw	carrier	src	dst	crsdephour	crsdeptime	depdelay	crsarrrtime	arrdelay	crselapsedtime	dist	lower_src
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	9	850	0.0	1116	0.0	146.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	11	1122	8.0	1349	0.0	147.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	14	1356	9.0	1623	0.0	147.0	946.0	atl	dl
ATL_BOS_2018-01-0...	2018-01-01	1	DL	ATL	BOS	16	1620	0.0	1851	3.0	151.0	946.0	atl	dl

ATL_BOS_2018-01-0... 2018-01-01	1	1	DL ATL BOS	19	1940	6.0	2210	0.0	150.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	DL ATL BOS	12	1248	0.0	1513	0.0	145.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	DL ATL BOS	22	2215	0.0	39	0.0	144.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	DL ATL BOS	15	1500	21.0	1734	33.0	154.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	WN ATL BOS	15	1500	198.0	1725	208.0	145.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	WN ATL BOS	21	2055	14.0	2330	0.0	155.0 946.0	atl
ATL_BOS_2018-01-0... 2018-01-01	1	1	WN ATL BOS	10	1015	215.0	1250	191.0	155.0 946.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	11	1114	0.0	1238	0.0	84.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	8	845	0.0	1011	0.0	86.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	15	1548	0.0	1710	0.0	82.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	7	705	0.0	821	0.0	76.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	12	1226	0.0	1347	0.0	81.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	AA ATL CLT	22	2205	0.0	2319	1.0	74.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	DL ATL CLT	22	2210	11.0	2324	0.0	74.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	DL ATL CLT	15	1543	1.0	1659	0.0	76.0 226.0	atl
ATL_CLT_2018-01-0... 2018-01-01	1	1	DL ATL CLT	10	1008	0.0	1124	0.0	76.0 226.0	atl

only showing top 20 rows

Since UDFs are not efficient, create methods that accept Column as parameter

Advantages

- More efficient as it can be optimised
- Not a blackbox
- No need to register as UDF
- No need to handle NULL (if you are only using inbuild sql functions)

```
import org.apache.spark.sql.functions._

def lowerColumnNullCheck(c: Column): Column = {
    org.apache.spark.sql.functions.lower(regex_replace(c, "\\s+", ""))
}

import org.apache.spark.sql.functions._
lowerColumnNullCheck: (c: org.apache.spark.sql.Column)org.apache.spark.sql.Column

flightDS.select(lowerColumnNullCheck('src').as("lower_src")).show
```

+-----+
lower_src
+-----+
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
atl
+-----+

only showing top 20 rows