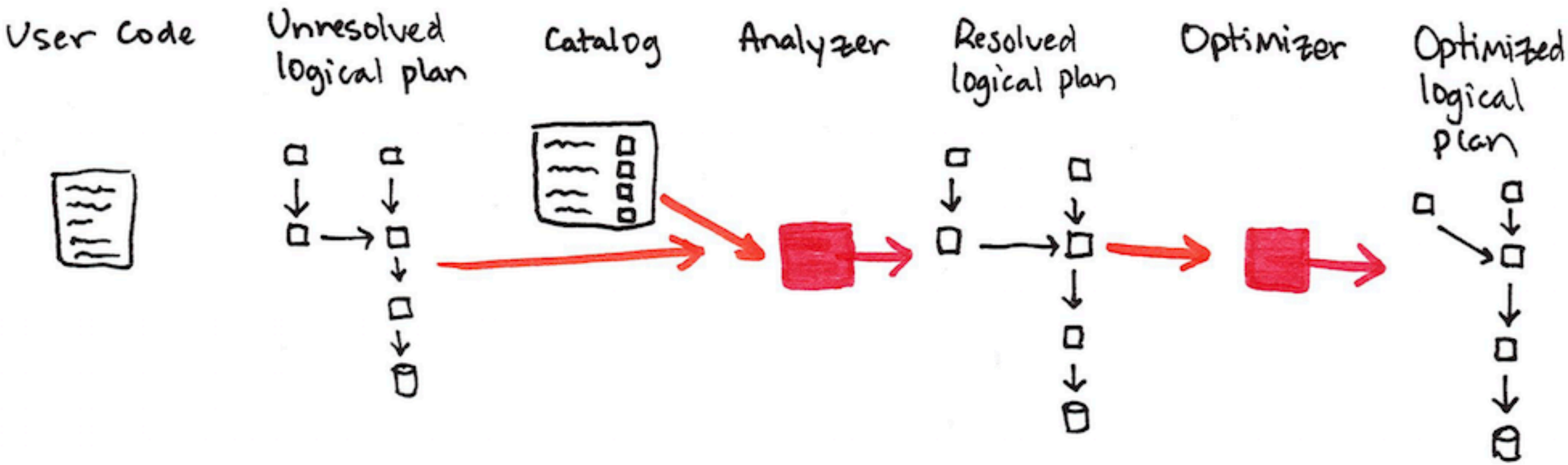
CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2017
Doc 9 Spark Transformations & Actions
Modified Oct 3, 2017

Questions

How many have Apache Toree kernel working - native or in Docker?

How many have tried to install it but failed?

Planning


User Code
Unresolved logical plan
Catalog
Analyzer
Resolved logical plan
Optimizer
Optimized logical plan

Schema

famousPeople.json

```
{"first": "Roger", "last":"Rabbit","age":29}
{ "last":"Modi","age":67}
{"first": "Sachin", "last":"Tendulkar","age":44}
```

```
val people = spark.read.format("json").load("famousPeople.json")
people.show
```

```
+---+------+---------+
|age| first|     last|
+---+------+---------+
| 29| Roger|   Rabbit|
| 67|  null|     Modi|
| 44|Sachin|Tendulkar|
+---+------+---------+
```

Schema

```
{"first": "Roger", "last":"Rabbit","age":29}
{ "last":"Modi","age":67}
{"first": "Sachin", "last":"Tendulkar","age":44}
```

```
val people = spark.read.format("json").load("famousPeople.json")
people.schema
```

```
StructType(
    StructField(age,LongType,true),
    StructField(first,StringType,true),
    StructField(last,StringType,true))
```

Schema

```
{"first": "Roger", "last":"Rabbit","age":29}
{ "last":"Modi","age":67}
{"first": "Sachin", "last":"Tendulkar","age":44}
```

```
import org.apache.spark.sql.types.{StructField, StructType, StringType, IntegerType}

val manualSchema = new StructType(Array(
  new StructField("first", StringType, true),
  new StructField("last", StringType, false),
  new StructField("age", IntegerType, false)
))

val people2 = spark.read.format("json").schema(manualSchema).load("famousPeople.json")
people2.show
```

```
+------+---------+---+
| first|     last|age|
+------+---------+---+
| Roger|   Rabbit| 29|
|  null|     Modi| 67|
|Sachin|Tendulkar| 44|
+------+---------+---+
```

Schema

```
{"first": "Roger", "last":"Rabbit","age":29}
{ "last":"Modi","age":67}
{"first": "Sachin", "last":"Tendulkar","age":44}
```

```
val manualSchema2 = new StructType(Array(
  new StructField("first", StringType, true),
  new StructField("last", StringType, false),
  new StructField("address", StringType, false)
))

val people3 = spark.read.format("json").schema(manualSchema2).load("famousPeople.json")
people3.show
```

```
+------+----------+-------+
| first|      last|address|
+------+----------+-------+
| Roger|    Rabbit|   null|
|  null|      Modi|   null|
|Sachin|Tendulkar|   null|
+------+----------+-------+
```

Time

```
val dwellFile = "/Users/whitney/Courses/696/Fall17/datasets/data/dwell-times.tsv"

val reader = spark.read
reader.option("header", true)
reader.option("inferSchema",true)
reader.option("sep", "   ")

val dwellDf = reader.csv("smallDwell.tsv")
dwellDf.printSchema
dwellDf.show(3)
```

dwell-times.tsv

date	dwell-time
2015-01-01T00:03:43Z	74

```
root
 |-- date: timestamp (nullable = true)
 |-- dwell-time: integer (nullable = true)

+--------------------+-----------+
|                date|dwell-time|
+--------------------+-----------+
|2014-12-31 16:03:43|         74|
|2014-12-31 16:32:12|        109|
|2014-12-31 17:52:18|         88|
+--------------------+-----------+
```

RDD Transformations & Actions

Transformations

```
val rdd = sc.parallelize(List(1,2,3,3))
```

```
val newRDD = rdd.map(x => x + 1)
newRDD.collect
```

```
Array(2, 3, 4, 4)
```

rdd.map(x -> x + 1).collect	Array(2, 3, 4, 4)
rdd.distinct().collect	Array(1, 2, 3)
rdd.sample(false,0.5).collect	varies

Transformations

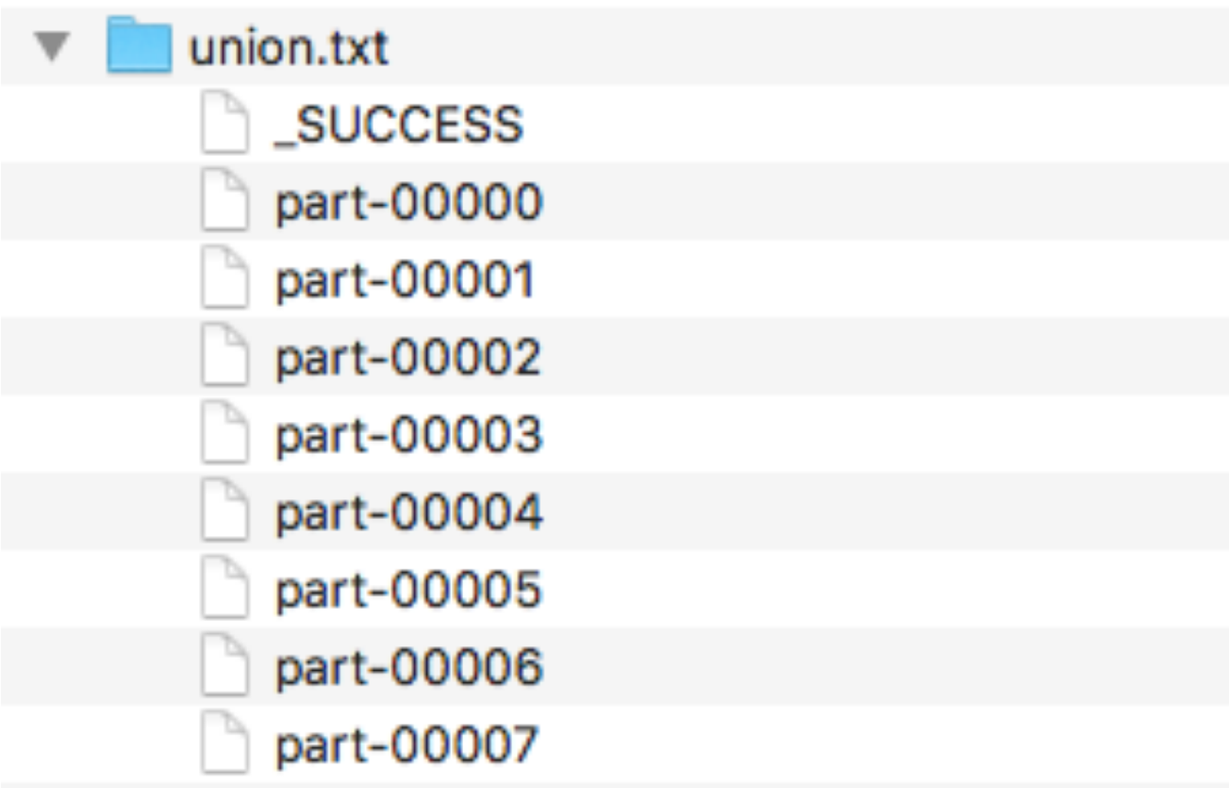
```
val rdd = sc.parallelize(List(1, 2, 3))
val other = sc.parallelize(List(3, 4, 5))
```

rdd.union(other).collect	Array(1, 2, 3, 3, 4, 5)
rdd.intersection(other).collect	Array(3)
rdd.subtract(other).collect	Array(1, 2)

Transformations & Output

```
val rdd = sc.parallelize(List(1, 2, 3))
val other = sc.parallelize(List(3, 4, 5))

val result = rdd.union(other)
result.saveAsTextFile("union.txt")
```



Actions & Output

```
val rdd = sc.parallelize(List(1, 2, 3))
val other = sc.parallelize(List(3, 4, 5))

val result = rdd.union(other)
val onMaster: Array[Int] = result.collect
```

Array is Scala type

Need to save as you would normal in Scala

Actions

```
val rdd =  sc.parallelize(List(3,1,2,3))
```

rdd.collect()	Array(3,1, 2, 3)
rdd.reduce((a,b) -> a + b)	9
rdd.first	3
rdd.top(2)	Array(3, 1)
rdd.take(2)	Array(3, 1)
rdd.takeOrdered(2)	Array(1, 2)
rdd.count()	3
rdd.countByValue()	Map(1 -> 1, 2 -> 1, 3 -> 2)

Statistical Actions

```
val rdd =  sc.parallelize(List(5.0,1.0,2.0,3.0,4.0,5.0))
```

rdd.mean()
rdd.min()
rdd.sum()
rdd.variance()
rdd.stdev()
rdd.stats()
rdd.histogram(bucketCount)

```
rdd.stats

(count: 6, mean: 3.333333, stdev: 1.490712,
  max: 5.000000, min: 1.000000)
```

```
rdd.histogram(3)

(Array(1.0, 2.3333333, 3.66666666665, 5.0),
  Array(2, 1, 3))
```

In Java you need DoubleRDD to call these methods

RDD of Pairs

```
val a = sc.parallelize(List(1, 1, 3))
val b = sc.parallelize(List(2, 4, 8))
val pairs = a.zip(b)
pairs.collect
```

pairs.collect	Array((1,2), (1,4), (3,8))
pairs.reduceByKey((x, y) => x + y).collect	Array((1,6), (3,8))
pairs.groupByKey().collect	Array((1,CompactBuffer(2, 4)), (3,CompactBuffer(8)))
pairs.mapValues(x => x + 1).collect	Array((1,3), (1,5), (3,9))
pairs.top(2)	Array((3,8), (1,4))
pairs.keys.collect	Array(1, 1, 3)
pairs.values.collect	Array(2, 4, 8)
pairs.sortByKey().collect	Array((1,2), (1,4), (3,8))

Understand Where Data is Located

```
val a = sc.parallelize(List(1, 2, 3, 4, 5, 6))
var sum = 0
a.foreach(x => sum + x)
sum
```

Result
0

```
a.reduce((a,b) => a+b)
```

21

DataFrame & DataSet Transformations

Add rows or columns

Remove rows or columns

Change row into column and column into row

Change order of rows

Where is the API Docs for DataFrame?

Data For Example

United States Bureau of Transportation statistics

The Definitive Guide, Zaharia & Chambers, O'Reilly Media

```
val df = spark.read.json("2015-summary.json")
```

2015-summary.json


```
{"ORIGIN_COUNTRY_NAME":"Romania","DEST_COUNTRY_NAME":"United States","count":15}
{"ORIGIN_COUNTRY_NAME":"Croatia","DEST_COUNTRY_NAME":"United States","count":1}
{"ORIGIN_COUNTRY_NAME":"Ireland","DEST_COUNTRY_NAME":"United States","count":344}
{"ORIGIN_COUNTRY_NAME":"United States","DEST_COUNTRY_NAME":"Egypt","count":15}
{"ORIGIN_COUNTRY_NAME":"India","DEST_COUNTRY_NAME":"United States","count":62}
{"ORIGIN_COUNTRY_NAME":"Singapore","DEST_COUNTRY_NAME":"United States","count":1}
{"ORIGIN_COUNTRY_NAME":"Grenada","DEST_COUNTRY_NAME":"United States","count":62}
{"ORIGIN_COUNTRY_NAME":"United States","DEST_COUNTRY_NAME":"Costa Rica","count":588}
{"ORIGIN_COUNTRY_NAME":"United States","DEST_COUNTRY_NAME":"Senegal","count":40}
{"ORIGIN_COUNTRY_NAME":"United States","DEST_COUNTRY_NAME":"Moldova","count":1}
```

Select & SelectExpr

SELECT * FROM dataFrameTable
SELECT columnName FROM dataFrameTable
SELECT columnName * 10, otherColumn, someOtherCol as c FROM dataFrameTable

import org.apache.spark.sql.functions.{**expr, col, column**}

```
val newDf = df.select(
  "DEST_COUNTRY_NAME",
  "ORIGIN_COUNTRY_NAME")
newDf.show(2)
```

```
+-----------------+-------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|
+-----------------+-------------------+
|    United States|            Romania|
|    United States|            Croatia|
+-----------------+-------------------+
```

Different Syntax

```
import org.apache.spark.sql.functions.{expr, col, column}

df.select(
  df.col("DEST_COUNTRY_NAME"),
  col("DEST_COUNTRY_NAME"),
  column("DEST_COUNTRY_NAME"),
  'DEST_COUNTRY_NAME,
  $"DEST_COUNTRY_NAME",
  expr("DEST_COUNTRY_NAME")
)
```

Columns & expo

```
col("someCol") - 5                    expr("someCol - 5")

(((col("someCol") + 5) * 200) - 6) < col("otherCol")

expr("(((someCol + 5) * 200) - 6) < otherCol")
```

Col example

```
import org.apache.spark.sql.functions._
val newDf = df.select(
    col(“DEST_COUNTRY_NAME"),
    col(“ORIGIN_COUNTRY_NAME"),
    col("count")*2 - sin("count"))
newDf.show(2)
```

```
+-----------------+-------------------+---------------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|((count * 2) - SIN(count))|
+-----------------+-------------------+---------------------------+
|    United States|            Romania|        29.349712159842884|
|    United States|            Croatia|        1.1585290151921035|
+-----------------+-------------------+---------------------------+
```

expo

```
import org.apache.spark.sql.functions._
val newDf = df.select(
    expr(“DEST_COUNTRY_NAME"),
    expr(“ORIGIN_COUNTRY_NAME"),
    expr("count *2 - sin(count)").alias("Goofy"))
newDf.show(2)
```

```
+-------------------+--------------------+-------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|              Goofy|
+-------------------+--------------------+-------------------+
|    United States|            Romania|29.349712159842884|
|    United States|            Croatia|1.1585290151921035|
+-------------------+--------------------+-------------------+
```

select + expr

```
import org.apache.spark.sql.functions._
val newDf = df.selectExpr("DEST_COUNTRY_NAME",
                          "ORIGIN_COUNTRY_NAME",
                          “count *2 - sin(count) as Goofier")
newDf.show(2)
```

```
+-------------------+---------------------+-------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|           Goofier|
+-------------------+---------------------+-------------------+
|    United States|             Romania|29.349712159842884|
|    United States|             Croatia|1.1585290151921035|
+-------------------+---------------------+-------------------+
```

Adding to Existing

```
df.selectExpr(
  "*",
  "(DEST_COUNTRY_NAME = ORIGIN_COUNTRY_NAME) as withinCountry"
).show(2)
```

```
+-----------------+-------------------+-----+--------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|withinCountry|
+-----------------+-------------------+-----+--------------+
|    United States|            Romania|   15|         false|
|    United States|            Croatia|    1|         false|
+-----------------+-------------------+-----+--------------+
```

Aggregate functions

```
import org.apache.spark.sql.functions._
val newDf = df.selectExpr("sum(count) as `Total Flights`",
                          "count(DEST_COUNTRY_NAME) as `Country Pairs`",
                          "count(Distinct(DEST_COUNTRY_NAME)) as Destinations",
                          "count(Distinct(ORIGIN_COUNTRY_NAME)) as Origins"
)
newDf.show
```

```
+--------------+--------------+-------------+-------+
|Total Flights|Country Pairs|Destinations|Origins|
+--------------+--------------+-------------+-------+
|       453316|           256|          132|    125|
+--------------+--------------+-------------+-------+
```

So What Functions can we use?

See org.apache.spark.sql.functions

org.apache.spark.sql.DataFrameStatFunctions

org.apache.spark.sql.DataFrameNaFunctions

Adding Columns with withColumn

df.**withColumn**("**numberOne**", **lit(1)**).show(2)

```
+-----------------+-------------------+-----+----------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|numberOne|
+-----------------+-------------------+-----+----------+
|    United States|            Romania|   15|         1|
|    United States|            Croatia|    1|         1|
+-----------------+-------------------+-----+----------+
```

df.withColumn("Random", **rand**()).show(2)

```
+-----------------+-------------------+-----+------------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|            Random|
+-----------------+-------------------+-----+------------------+
|    United States|            Romania|   15|0.6922143025538695|
|    United States|            Croatia|    1|0.4291601163044023|
+-----------------+-------------------+-----+------------------+
```

WithColumn & Expr

```
val inCountry = df.withColumn(
  "withinCountry",
  expr("ORIGIN_COUNTRY_NAME == DEST_COUNTRY_NAME")
)

inCountry.show(2)
```

```
+-----------------+-------------------+-----+--------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|withinCountry|
+-----------------+-------------------+-----+--------------+
|    United States|            Romania|   15|         false|
|    United States|            Croatia|    1|         false|
+-----------------+-------------------+-----+--------------+
```

Rename

```
df.withColumnRenamed("DEST_COUNTRY_NAME", "dest").columns

    Array(dest, ORIGIN_COUNTRY_NAME, count)
```

to_date

```
val dwellDf = reader.csv("smallDwell.tsv")
dwellDf.show(2)
```

```
+--------------------+-----------+
|                date|dwell-time|
+--------------------+-----------+
|2014-12-31 16:03:43|         74|
|2014-12-31 16:32:12|        109|
+--------------------+-----------+
```

```
dwellDf.withColumn("Date", to_date(col("date"))).show(2)
```

```
+-----------+-----------+
|      Date|dwell-time|
+-----------+-----------+
|2014-12-31|         74|
|2014-12-31|        109|
+-----------+-----------+
```

Date, Hour, format

```
val withHour = dwellDf.withColumnRenamed("date", "TimeStamp").
                        withColumn("Date", to_date(col("TimeStamp"))).
                        withColumn("Hour", hour(col("TimeStamp"))).
                        withColumn("Month", date_format(col("Date"),"MMMM"))
withHour.show(2)
```

```
+--------------------+-----------+-----------+----+---------+
|           TimeStamp|dwell-time|       Date|Hour|    Month|
+--------------------+-----------+-----------+----+---------+
|2014-12-31 16:03:43|         74|2014-12-31|  16|December|
|2014-12-31 16:32:12|        109|2014-12-31|  16|December|
+--------------------+-----------+-----------+----+---------+
```

Single Pass

```
val withHour = dwellDf.withColumnRenamed("date", "TimeStamp").
                        withColumn("Date", to_date(col("TimeStamp"))).
                        withColumn("Hour", hour(col("TimeStamp"))).
                        withColumn("Month", date_format(col("Date"),"MMMM"))
withHour.explain()
```

```
== Physical Plan ==
*Project [date#1116 AS TimeStamp#1656, dwell-time#1117,
   to_date(cast(date#1116 as date)) AS Date#1660,
   hour(date#1116, Some(America/Los_Angeles)) AS Hour#1665,
   date_format(cast(to_date(cast(date#1116 as date)) as timestamp), MMMM, Some(America/
   Los_Angeles)) AS Month#1671]
+- *FileScan csv [date#1116,dwell-time#1117]
         Batched: false, Format: CSV,
         Location: InMemoryFileIndex[file:/Users/whitney/Courses/696/Fall17/notebookExamples/
smallDwell.tsv],
         PartitionFilters: [], PushedFilters: [], ReadSchema: struct<date:timestamp,dwell-time:int>
```

Dropping Columns

```
df.drop("ORIGIN_COUNTRY_NAME").columns

    Array(DEST_COUNTRY_NAME, count)
```

Selecting Rows filter = where

```
val colCondition = df.filter(col("count") < 2).take(2)
val conditional = df.where("count < 2").take(2)
```

```
df.where(col("count") < 2)
.where(col("ORIGIN_COUNTRY_NAME") =!= "Croatia")
.show(2)
```

Another where

```
val inCountry = df.withColumn(
                    “withinCountry",
                    expr("ORIGIN_COUNTRY_NAME == DEST_COUNTRY_NAME”))
              .where(col("withinCountry"))

inCountry.show(2)
```

```
+-----------------+-------------------+------+--------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME| count|withinCountry|
+-----------------+-------------------+------+--------------+
|    United States|      United States|370002|          true|
+-----------------+-------------------+------+--------------+
```

where’s are anded

```
import org.apache.spark.sql.functions.{col, column}

val frequent = col("count") > 100
val toUSA = col("DEST_COUNTRY_NAME").contains("United States”)

val frequentToUSA = df.where(frequent).where(toUSA)
frequentToUSA.show(5)

+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Ireland|  344|
|    United States|       Sint Maarten|  325|
|    United States|             Russia|  161|
|    United States|        Netherlands|  660|
|    United States|            Ecuador|  300|
+-----------------+-------------------+-----+
```

or

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

val frequent = col("count") > 100
val toUSA = col("DEST_COUNTRY_NAME").contains("United States”)

val frequentToUSA = df.where(frequent.or(toUSA))
frequentToUSA.show(7)
```

```
+-----------------+--------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+--------------------+-----+
|    United States|             Romania|   15|
|    United States|             Croatia|    1|
|    United States|             Ireland|  344|
|    United States|               India|   62|
|    United States|           Singapore|    1|
|    United States|             Grenada|   62|
|       Costa Rica|       United States|  588|
+-----------------+--------------------+-----+
```

!, not

```
val inCountry = df.withColumn(
  "withinCountry",
  expr("ORIGIN_COUNTRY_NAME == DEST_COUNTRY_NAME")
).where(!col("withinCountry"))

inCountry.show(5)
```

```
+-----------------+-------------------+-----+--------------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|withinCountry|
+-----------------+-------------------+-----+--------------+
|    United States|            Romania|   15|         false|
|    United States|            Croatia|    1|         false|
|    United States|            Ireland|  344|         false|
|            Egypt|      United States|   15|         false|
|    United States|              India|   62|         false|
+-----------------+-------------------+-----+--------------+
```

```
  ).where(not(col("withinCountry")))
```

Distinct Rows - distinct

```
df.select("ORIGIN_COUNTRY_NAME").
    distinct().
    sort(col("ORIGIN_COUNTRY_NAME")).
    take(10)
```

```
Array([Angola],
    [Anguilla],
    [Antigua and Barbuda],
    [Argentina],
    [Aruba],
    [Australia],
    [Austria],
    [Azerbaijan],
    [Bahrain],
    [Barbados])
```

Sort === orderBy

```
df.sort("count").show(5)
df.orderBy("count", "DEST_COUNTRY_NAME").show(5)
df.orderBy(col("count"), col("DEST_COUNTRY_NAME")).show(5)
```

```
import org.apache.spark.sql.functions.{desc, asc}
```

~~df.orderBy(expr("count desc")).show(2)~~

```
df.orderBy(desc("count"), asc("DEST_COUNTRY_NAME")).show(2)
```

WTF?

val topTenDF = df.orderBy(expr("count **asc**"))
topTenDF.show(3)

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|          Moldova|      United States|    1|
|    United States|          Singapore|    1|
|    United States|            Croatia|    1|
+-----------------+-------------------+-----+
```

val topTenDF = df.orderBy(expr("count **desc**"))
topTenDF.show(3)

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|          Moldova|      United States|    1|
|    United States|          Singapore|    1|
|    United States|            Croatia|    1|
+-----------------+-------------------+-----+
```

Limit

```
import org.apache.spark.sql.functions.{desc, asc}
val topTenDF = df.orderBy(desc("count")).limit(10)
topTenDF.show
topTenDF.count
```

```
+-----------------+-------------------+------+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME| count|
+-----------------+-------------------+------+
|    United States|      United States|370002|
|    United States|             Canada|  8483|
|           Canada|      United States|  8399|
|    United States|             Mexico|  7187|
|           Mexico|      United States|  7140|
|   United Kingdom|      United States|  2025|
|    United States|     United Kingdom|  1970|
|            Japan|      United States|  1548|
|    United States|              Japan|  1496|
|          Germany|      United States|  1468|
+-----------------+-------------------+------+
```

10

Take, Collect

Return DF to master node as Array

take(n)

collect returns all

Appending Rows to a DataFrame

```
import org.apache.spark.sql.Row
import org.apache.spark.sql.functions.col

val schema = df.schema

val newRows = Seq(
  Row("New Country", "Other Country", 5L),
  Row("New Country 2", "Other Country 3", 1L)
)
val parallelizedRows = spark.sparkContext.parallelize(newRows)

val newDF = spark.createDataFrame(parallelizedRows, schema)

val added = df.union(newDF)

added.where("count = 1").
  where(col("ORIGIN_COUNTRY_NAME") =!= "United States").
  show()
```

union add DataFrame to end

Schemas of the two DataFrames must match

Output

```
+-----------------+--------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+--------------------+-----+
|    United States|            Croatia|    1|
|    United States|          Singapore|    1|
|    United States|          Gibraltar|    1|
|    United States|             Cyprus|    1|
|    United States|            Estonia|    1|
|    United States|          Lithuania|    1|
|    United States|           Bulgaria|    1|
|    United States|            Georgia|    1|
|    United States|            Bahrain|    1|
|    United States|   Papua New Guinea|    1|
|    United States|         Montenegro|    1|
|    United States|            Namibia|    1|
|    New Country 2|    Other Country 3|    1|
+-----------------+--------------------+-----+
```

Random Samples

```
val tenDF = df.limit(10)
tenDF.show
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|    United States|          Singapore|    1|
|    United States|            Grenada|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+
```

```
val seed = 5
val withReplacement = false
val fraction = 0.5

val smallSample = tenDF.sample(withReplacement, fraction, seed)
smallSample.show
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
+-----------------+-------------------+-----+
```

Random Splits

Split DF into two disjoint parts randomly

One dataframe for training
One for validation

randomSplit

```
// 5 = seed
val twoDF = tenDF.randomSplit(Array(0.25, 0.75), 5)
twoDF(0).show
twoDF(1).show
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|   15|
|    United States|            Croatia|    1|
|    United States|            Ireland|  344|
|            Egypt|      United States|   15|
|    United States|              India|   62|
|    United States|          Singapore|    1|
|    United States|            Grenada|   62|
|       Costa Rica|      United States|  588|
|          Senegal|      United States|   40|
|          Moldova|      United States|    1|
+-----------------+-------------------+-----+
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|       Costa Rica|      United States|  588|
|    United States|            Ireland|  344|
|    United States|            Romania|   15|
+-----------------+-------------------+-----+
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|            Egypt|      United States|   15|
|          Moldova|      United States|    1|
|          Senegal|      United States|   40|
|    United States|            Croatia|    1|
|    United States|            Grenada|   62|
|    United States|              India|   62|
|    United States|          Singapore|    1|
+-----------------+-------------------+-----+
```

Weights Normalized to 1

```
val twoDF = tenDF.randomSplit(Array(0.25, 0.50))
twoDF(0).show
twoDF(1).show
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|       Costa Rica|      United States|  588|
|    United States|            Ireland|  344|
|    United States|            Romania|   15|
+-----------------+-------------------+-----+

+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|            Egypt|      United States|   15|
|          Moldova|      United States|    1|
|          Senegal|      United States|   40|
|    United States|            Croatia|    1|
|    United States|            Grenada|   62|
|    United States|              India|   62|
|    United States|          Singapore|    1|
+-----------------+-------------------+-----+
```

User Defined Functions on DataFrames

```
def add100(n:Long):Long = { n + 100 }

import org.apache.spark.sql.functions.udf

val add100udf = udf(add100(_:Long):Long)

val added = df.withColumn("100", add100udf(col("count")))
added.show(3)
```

```
+-----------------+-------------------+-----+---+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|100|
+-----------------+-------------------+-----+---+
|    United States|            Romania|   15|115|
|    United States|            Croatia|    1|101|
|    United States|            Ireland|  344|444|
+-----------------+-------------------+-----+---+
```

Type Mismatch

```
def add100(n:Long):Long = { n + 100 }

import org.apache.spark.sql.functions.udf

val add100udf = udf(add100(_:Long):Long)

val added = df.withColumn("100", add100udf(col("DEST_COUNTRY_NAME")))
added.show(3)
```

```
+-----------------+-------------------+-----+----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count| 100|
+-----------------+-------------------+-----+----+
|    United States|            Romania|   15|null|
|    United States|            Croatia|    1|null|
|    United States|            Ireland|  344|null|
+-----------------+-------------------+-----+----+
```

Repartition & Coalesce

Can change the number of partitions

repartition
- Causes full shuffle
- Increase or decrease number of partition

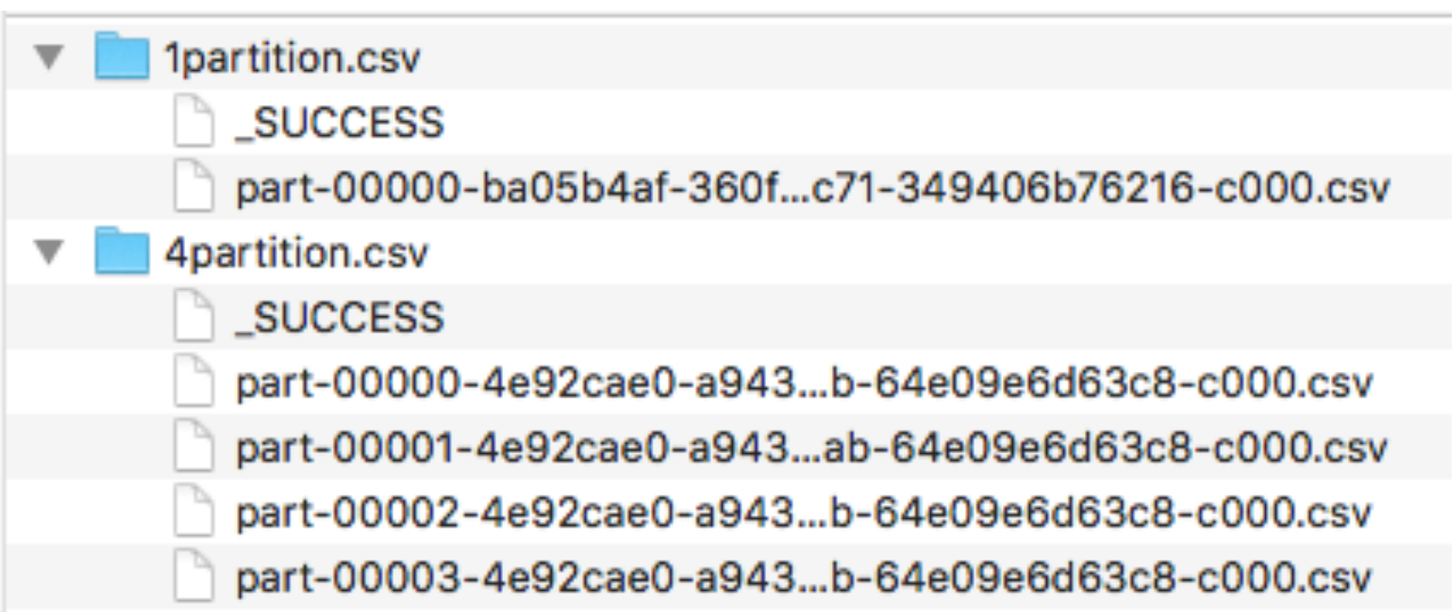
coalesce
- No shuffle
- Combines partitions only

Example

```
df.rdd.getNumPartitions
    1

dfPart4.rdd.getNumPartitions
val dfPart4 = df.repartition(4)
     4


df.write.format("csv").save("1partition.csv")
dfPart4.write.format("csv").save("4partition.csv")
```



Partition on a Column

```
val byColumn = df.repartition(col("DEST_COUNTRY_NAME"))

var byColumnWithSize = df.repartition(5, col("DEST_COUNTRY_NAME"))
```

Aggregations

Summarize

groupBy

roll up

cube

window

Aggregation Functions

count
countDistinct
approx_count_distinct
first, last
min, max
sum
sumDistinct
avg, mean
variance, var_samp, var_pop
stddev, stddev_samp, stddev_pop
skewness, kurtosis

Covariance & Correlation
    corr, covar_samp, covar_pop

Example

```
import org.apache.spark.sql.functions._
val newDf = df.selectExpr("sum(count) as Sum",
                          "mean(count) as Mean",
                          "max(count) as Max",
                          "stddev_samp(count) as `Sample StdDev`",
                          "stddev_pop(count) as `Pop StdDev`",
                          "count(DEST_COUNTRY_NAME) as Count")
newDf.show
```

```
+------+-----------+------+------------------+-----------------+-----+
|   Sum|       Mean|   Max|     Sample StdDev|       Pop StdDev|Count|
+------+-----------+------+------------------+-----------------+-----+
|453316|1770.765625|370002|23126.516918551915|23081.30374350104|  256|
+------+-----------+------+------------------+-----------------+-----+
```

Counting

```
import org.apache.spark.sql.functions._
val newDf = df.select(countDistinct(col("DEST_COUNTRY_NAME")).alias("Distinct Dest"),
                countDistinct(col("ORIGIN_COUNTRY_NAME")).alias("Distinct Origin"),
                countDistinct(
                        col("DEST_COUNTRY_NAME"),
                        col("ORIGIN_COUNTRY_NAME")).alias("Distinct Pair"),
                countDistinct(
                        col("DEST_COUNTRY_NAME"),
                        col(“ORIGIN_COUNTRY_NAME"),
                        col("count")).alias("Distinct Rows"),
                approx_count_distinct(col("ORIGIN_COUNTRY_NAME")).alias("Approx Origin"))
newDf.show
```

```
+-------------+---------------+-------------+-------------+-------------+
|Distinct Dest|Distinct Origin|Distinct Pair|Distinct Rows|Approx Origin|
+-------------+---------------+-------------+-------------+-------------+
|          132|            125|          256|          256|          116|
+-------------+---------------+-------------+-------------+-------------+
```

These functions did not work in selectExpr

Group By Data

```
val reader = spark.read
reader.option("header",true).option("inferSchema",true)
val ordersDF = reader.csv("orders.csv")
ordersDF.show
```

```
+--------+------+
|customer|amount|
+--------+------+
|       a|     2|
|       b|     8|
|       a|     3|
|       c|     9|
|       a|     4|
|       b|    16|
|       c|    11|
|       b|    24|
|       c|    30|
+--------+------+
```

groupBy

```
val amountGrouped = ordersDF.groupBy("customer").
      agg(
        sum("amount").alias("Total"),
        mean("amount").alias("Average"),
        count("amount").alias("Number of Orders"))
amountGrouped.sort("customer").show
```

```
+--------+-----+------------------+----------------+
|customer|Total|           Average|Number of Orders|
+--------+-----+------------------+----------------+
|       a|    9|               3.0|               3|
|       b|   48|              16.0|               3|
|       c|   50|16.666666666666668|               3|
+--------+-----+------------------+----------------+
```